EVALUATION OF FOSS VIDEO GAMES IN COMPARISON TO THEIR COMMERCIAL COUNTERPARTS

A Thesis in Media Studies

By Jesse A. Clark

© 2008 Jesse A. Clark

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Arts

August 2008
The thesis of Jesse A. Clark was reviewed and approved* by the following::

John Nichols  
Professor of Communications  
Associate Dean for Graduate Studies and Research

Matt Jackson  
Associate Professor of Communications  
Head of Department of Telecommunications  
Thesis Advisor

Robert Frieden  
Professor; Pioneers Chair in Telecommunications

Ronald Bettig  
Associate Professor of Communications

*Signatures are on file in the Graduate School.
Abstract

The topic of copyrights and copyright law is a crucial component in understanding today's media landscape. The purpose for having a copyright system as outlined in the U.S. Constitution is to provide content creators with an incentive to create. The copyright system allows revenue to be generated through sales of copies of works; thus allowing for works to be created which otherwise would not be created. Yet it is entirely possible that not all large creative projects require the same legal framework as an incentive. The so called “copyleft” movement (which will be defined and explained in depth later) offers an alternative to the industrial mode of cultural production. Superficially, “copylefted” works can be divided into two broad categories: artistic/creative works (which are often protected by “Creative Commons” licenses), and Free/Open Source Software. This thesis evaluates how open source video games compare to their commercial counterparts and discusses the reasons for any difference in overall quality. From this evaluation, this thesis attempts to evaluate whether copyright protection is necessary for high quality video games to be developed. This study finds that, in term of technical merit, FOSS games vary widely. The most sophisticated of these games, however, seem to be only a few years behind their commercial counterparts. A partial explanation for this disparity can be found in the fact that commercial games have significantly larger user bases, and these users contribute a large amount of value to commercial games in the form of free user modifications. This study suggests that FOSS games could be greatly benefited if they were able to induce more users to take part in developing modifications and levels for these games rather than for commercial games. The implication for copyright law is that, at least in some cases, strong copyright protection may not be necessary for the production of high quality video games.
# Table of Contents

List of Tables ........................................................................................................................................ v
List of Figures ......................................................................................................................................... vi
Introduction ......................................................................................................................................... 1
Chapter 1: Overview of Copyrights and the “Copyleft” Alternative .................................................... 5
  Introduction to Copyright Law and Criticisms Thereof: ................................................................. 5
  Case Law Relating to Video Games and Copyrights: ................................................................. 10
  Introduction to the “Copyleft” Movement: .................................................................................... 14
    Why I Must Write GNU ............................................................................................................... 16
Chapter 2: Economic Perspectives on FOSS Production, Consumption, and Policy ......................... 25
  Articles dealing with the supply side: ............................................................................................ 28
  Articles dealing with the demand side: .......................................................................................... 35
  Discussion: .................................................................................................................................. 38
Chapter 3: Video Games and the Video Game Industry ....................................................................... 40
  History: ....................................................................................................................................... 40
  Structure: .................................................................................................................................... 41
  The Role of User Generated Content: .......................................................................................... 47
  Literature Relating to FOSS Games: ............................................................................................. 52
  Sales Statistics: ............................................................................................................................. 54
  Conclusion: .................................................................................................................................. 56
  Restatement of Research Questions: ............................................................................................ 57
Chapter 4: Methodology ..................................................................................................................... 58
Chapter 5: Results ............................................................................................................................... 77
  StepMania: .................................................................................................................................. 77
  Battle for Wesnoth: ....................................................................................................................... 81
  Sauerbraten Game & Engine: ......................................................................................................... 84
  Scorched 3D: ................................................................................................................................ 88
  Nexuiz: ......................................................................................................................................... 90
  Armagetron Advanced: .................................................................................................................... 96
  FreeCiv: ......................................................................................................................................... 100
  Secret Maryo Chronicles: .............................................................................................................. 103
  Table Summarizing Results: ........................................................................................................ 105
Chapter 6: Conclusions, Limitations, and Future Research ................................................................. 108
  Conclusions and Discussion: ......................................................................................................... 108
  Limitations and Future Research: ................................................................................................. 115
Appendix A: Inside the Development of a Simple Game ..................................................................... 120
Appendix B: Screenshots .................................................................................................................... 130
References .......................................................................................................................................... 145
List of Tables

Table 1 Consoles (Including Handhelds) Sales Data for 2007 ................................................................. 55
Table 2 Top Ten Console (Including Handhelds) Games for 2007 .......................................................... 55
Table 3 Top Ten PC Games of 2007 ........................................................................................................ 56
Table 4 Top Ten FOSS Games on SourceForge ....................................................................................... 56
Table 5 Games in Sample ......................................................................................................................... 76
Table 6 Summary of Results ................................................................................................................... 107
**List of Figures**

Figure 1 Illustration of Scrolling on Both an x-axis and y-axis With a Mini-Map Drawn On-Screen ... 61
Figure 2 Nexuiz 2.4 .................................................................................................................................. 130
Figure 3 Nexuiz 2.4 (second screenshot) ................................................................................................ 131
Figure 4 Sauerbraten (December 2007 Version) .................................................................................. 132
Figure 5 Sauerbraten (December 2007 Version) .................................................................................. 133
Figure 6 Stepmania 4............................................................................................................................. 134
Figure 7 Battle for Wesnoth 1.4 ............................................................................................................ 135
Figure 8 Scorched 3d 41.3 .................................................................................................................. 136
Figure 9 Armagetron Advanced 3.0 .................................................................................................. 137
Figure 10 Freeciv 2.1.3.......................................................................................................................... 138
Figure 11 Secret Maryo Chronicles 1.4 ............................................................................................... 139
Figure 12 Civilization IV ...................................................................................................................... 140
Figure 13 Super Mario Bros 3 ............................................................................................................... 141
Figure 14 Crysis .................................................................................................................................... 142
Figure 15 Unreal Tournament III ........................................................................................................ 143
Figure 16 Unreal Tournament 2004 .................................................................................................... 144
Introduction

Copyright law is a crucial component in understanding today's media landscape. Media industries have developed business models which depend on strong copyright protections in order to generate profit. These same industries have played a significant role in the creation of copyright law as it exists today in order to protect and expand the profitability of existing business models. Beyond protecting the profitability of media industries, copyright law also plays a significant role in determining what types of media products are created, how they are created, how they are distributed, and how they can be used. For this reason, copyright law deserves the attention of researchers.

The purpose for having a copyright system as outlined in the U.S. Constitution is to provide content creators with an incentive to create. However, today's copyright system has become more focused on protecting industrial profits than providing an incentive to create. While the current system may be necessary to provide an incentive for the production of certain types of capital intensive works, the monopoly granted by copyright law also limits the circulation of works and the ability of others to improve upon existing works. Copyright law allows creative works to be sold far above marginal cost, thus limiting their availability. Furthermore, some scholars have argued that copyright law reduces human creativity to a consumer commodity rather than culture owned by all.

On the other hand, a compelling argument can be made that many of the creative works we enjoy today, such as motion pictures, computer software, and video games, appear to require intensive capital investment and human resources. In order for such large endeavors to take place, some mechanism must exist for funding the projects and organizing the human resources. The copyright system allows revenue to be generated through sales of copies of works; thus allowing for works to be created which otherwise would not be created. Content creators may create works for a variety of reasons. Such reasons may include prestige (as is often the case with academic works), a desire to make art, or to generate a profit. Even when generation of a profit is not a significant motivation,
however, some creators may require a way to recoup monetary costs of production. Yet it is entirely possible that not all large projects require the same legal framework in order to take place. This thesis will explore one such category of works: video games.

The so called “copyleft” movement (which will be defined and explained in depth later) offers an alternative to the industrial mode of cultural production. Essentially, the “copyleft” movement attempts to apply copyright law in such a way as to insure that works remain accessible to the public and can be freely circulated and modified. Superficially, “copylefted” works can be divided into two broad categories: artistic/creative works (which are often protected by “Creative Commons” licenses), and Free/Open Source Software. Much research could be done on either of these two broad categories of works, and generalized research on either of these two categories would be too large a subject for this Master’s thesis.

This thesis will focus on open source video games. While a large body of high quality open source business software, system software, and miscellaneous utility software exists, the purpose of this research is to evaluate the quality of open source video games. Unlike video games, much open source business and utility software has received significant corporate backing. For example, Sun has been heavily involved in the OpenOffice.org project (an open source office suite which competes with Microsoft Office). Viable business models, such as by providing services and support, exist for generating profit through the development of open source software. It seems less likely that open source video games will receive significant support from for-profit firms (although, as will be seen later, in certain cases commercial game developers do indeed contribute to the open source code base). For this reason, it is worthwhile to evaluate how open source games compare to their commercial counterparts.

Because of the apparent lack of viable business models relating to the production of open source games, understanding what differences exist between open source video games and their commercial
counterparts will provide insight into the question of what types of content can be produced absent commercial incentives. Yet, after we establish how open source games compare to commercial games, it is important that we attempt to understand the causes of observed similarities and differences. From this evaluation, we should be able to gain a better understanding of the role and utility of copyright law in relation to video games. Thus, the research questions are as follows:

**RC1:** How do open source videogames compare to their commercial counterparts?

**RC2:** How can any disparities in quality be explained?

**RC3:** What implications, if any, do these findings have for the utility of copyright law in regard to video games?

For this study, a formal analysis of open source video games across several genres will be performed; the games will be evaluated in terms of both technical quality and subjective entertainment (both evaluated on multiple dimensions) when compared to the standards set by contemporary commercial video games. Popularity and user ratings also will be used in the evaluation of the open source games.

Chapter 1 will provide a general discussion about copyrights and will address criticisms of the current copyright system. After this introduction to copyright law, this chapter will provide an introduction to the so called “copyleft” movement. It will discuss both the ideology of copyleft and various copyleft licenses. Toward the end of the first chapter, theories from political economy will be used to explain ideological differences in Free/Open Source Software (FOSS). Chapter 2 will provide an in depth look at the economics of FOSS. A major focus of this chapter will be on both supply side and demand side economics of open source software. Policy options based on an economic perspective will also be addressed in this chapter. Chapter 3 will explore both the history and the economics of the video game industry. This chapter will also explore the phenomenon of user generated modifications to commercial video games, and how this relates both to the video game industry and to FOSS video
games. Finally, this chapter will provide a literature review of the extremely limited research which has been done on FOSS games. Chapter 4 will provide a detailed description of the methodology both for the selection and the evaluation of the FOSS games. Chapter 5 will present the results of my study. Finally, Chapter 6 will present conclusions, limitations, and ideas for future research. Appendix “A” will provide an overview as to what goes into the development of a video game by walking through the development process of a simple game I made myself. Appendix “B” will consist of screenshots from both the FOSS games evaluated and from a select few commercial games for visual comparison.

This study finds that, in term of technical merit, FOSS games vary widely. The most sophisticated of these games, however, seem to be only a few years behind their commercial counterparts. A partial explanation for this disparity is that commercial games have significantly larger user bases, and these users contribute a large amount of value to commercial games in the form of free user modifications. This study suggests that FOSS games could greatly benefit if they were able to induce more users to take part in developing modifications and levels for these games rather than for commercial games. The implication for copyright law is that, at least in some cases strong copyright protection may not be necessary for the production of high quality video games.
Chapter 1: Overview of Copyrights and the “Copyleft” Alternative

Before discussing the copyleft movement and FOSS, it is necessary to take a look at the current copyright system. The fact that people see a need to produce software under FOSS licenses implies that, at least in some cases, the dominant copyright paradigm has significant shortcomings. This chapter will provide a brief overview of the copyright system, outline some criticisms of the current copyright climate and discuss why many contend that it often not only stifles content creation, but narrows the public domain and works against the public interest. This chapter will also provide a brief review of case law relating to the application of copyrights to video games. This is by no means meant to be a comprehensive critique of the current copyright system; rather, this chapter will point out some of the most serious tensions that the “copyleft” movement attempts to resolve.

After discussing the copyright system and its application to video games, this chapter will present an introduction to the “copyleft” movement which has begun to counteract and challenge the fundamental assumptions of intangible “property”. This chapter has several major goals. The so-called “copyleft” movement will be defined, and a brief history of its development will be discussed. A discussion of the various ideologies and goals of the movement will also be presented. In this discussion, it will be necessary to provide an overview of various licenses in this movement, as the technical terms of various copyleft licenses relates closely to the ideology. Toward the end of the chapter, more attention will be given specifically to the FOSS movement. A discussion of both the broader copyleft movement and the FOSS movement in particular will be presented using perspectives related to political economy.

Introduction to Copyright Law and Criticisms Thereof:

In Article 1, Section 8 of the United States Constitution, Congress is granted the right “To promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries”. This is the constitutional basis of
copyright law in the United States. But is the current copyright climate serving to further the goal of promoting the progress of science and useful arts? Or is it actually working against this goal and against the public interest in general? This study contends that, at least to some extent, the latter is the case.

The subject matter of copyrights, along with language limiting the scope of copyrights is outlined in U.S.C. Title 17, Chapter 1, Section 2:

(a) Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. Works of authorship include the following categories:
   (1) literary works;
   (2) musical works, including any accompanying words;
   (3) dramatic works, including any accompanying music;
   (4) pantomimes and choreographic works;
   (5) pictorial, graphic, and sculptural works;
   (8) architectural works.

(b) In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.

By rules established by the Copyright Office in 1988, computer programs (including video games) can be registered as either literary works or audiovisual works, but not both (the decision as to how software should be registered should be made in accordance to which element of the work dominates) (Docket No. 87-4, 53 Fed. Reg. 21817, June 10, 1988). It should be emphasized that under the statute, fixation in a tangible medium is a prerequisite to securing a copyright and that ideas, procedures, processes, systems, methods of operation, concepts, principles, and discoveries are not copyrightable (although patent law provides protection in some cases where copyright law does not).

The specific rights granted to copyright holders are defined in U.S.C. Title 17, Chapter 1, Section 106, which reads as follows:

Subject to sections 107 through 122, the owner of copyright under this title has the exclusive rights to do and to authorize any of the following:
(1) to reproduce the copyrighted work in copies or phonorecords;
(2) to prepare derivative works based upon the copyrighted work;
(3) to distribute copies or phonorecords of the copyrighted work to the public by sale or other transfer of ownership, or by rental, lease, or lending;
(4) in the case of literary, musical, dramatic, and choreographic works, pantomimes, and motion pictures and other audiovisual works, to perform the copyrighted work publicly;
(5) in the case of literary, musical, dramatic, and choreographic works, pantomimes, and pictorial, graphic, or sculptural works, including the individual images of a motion picture or other audiovisual work, to display the copyrighted work publicly; and
(6) in the case of sound recordings, to perform the copyrighted work publicly by means of a digital audio transmission.

Thus, copyright holders are legally granted the right to restrict copying and public display/performance of their works. In addition, they are granted the right to restrict the creation of derivative works. When exercised, these rights allow copyright holders to generate profit through the sale of copies of their work and through payment of royalties for public display/performance and creation of derivative works.

When the copyright of a work expires, the work is said to be in the public domain. When a work is in the public domain, it is essentially public property. Anyone can use a work in the public domain in any way she pleases. She is free to copy and redistribute the work with or without giving attribution to the original author. She is even free to modify the work and copyright the modified version. Classical works such as the writings of William Shakespeare are examples of works in the public domain. In 1790, when the first national copyright law was passed in the United States, a copyrighted work would pass into the public domain after a maximum of 28 years (copyrights lasted 14 years, and could optionally be extended an additional 14 years); today copyrights last an average of 95 years, so a work published today will not be in the public domain within our lifetime (Jensen, 2004). This extension of the term of copyright protection can be criticized as it is dubious as to whether the framers of the constitution would have considered 95 years compatible with the requirement that authors only receive exclusive rights “for a limited time”.

Bailey (2006) writes about another distinction between the state of copyright law in 1790 and
the present. In 1790, for a work to be protected by copyrighted, it needed to be registered; but very few works were registered at that time, so most fell into the public domain by default. Today, no registration is required for a work to be copyrighted, and unless the author explicitly dedicates a work to the public domain, it is automatically copyrighted the moment it is fixed in a tangible medium. A consequence of this, according to Bailey, is many copyrighted works become “orphan works”. When an author of a copyrighted work cannot be contacted, people are unable to obtain permission to republish or create a derivative work from the copyrighted material. Thus it is possible that some forms of creative expression will be needlessly lost. This has the effects of decreasing the amount of information the public has access to and stifling creative innovation.

Why has copyright law been extended so much? Bailey suggests it is the result of Congress turning the responsibility for drafting copyright legislation over to corporate copyright holders:

By 1901, copyright law had become fairly esoteric and complex, and drafting new copyright legislation had become increasingly difficult. Consequently, Congress adopted a new strategy: let those whose commercial interests were directly affected by copyright law deliberate and negotiate with each other about copyright law changes, and use the results of this process as the basis of new legislation. Over time, this increasingly became a dialog among representatives of entertainment, high-tech, information, and telecommunications companies: other parties, such as library associations: and rights holder groups (e.g., ASCAP). Since these parties often had competing interests the negotiations were frequently contentious and lengthy, and the resulting laws created a kind of crazy quilt of specific exceptions for the deals made during these sessions to the ever expanding control over intellectual works that copyright reform generally engendered. Since the public was not at the table, its highly diverse interests were not directly represented, and, since stakeholder industries lobby Congress and the public doesn't, the public's interests were often not served. (Bailey, 2006).

The fair use doctrine was codified by the Copyright Act of 1976. The purpose of this legislation was to balance the relationship between copyright holders and the public, and to protect the public from abuse of copyrights by the right holders. Another purpose of the fair use doctrine was to help assure that copyright law would not deviate from its constitutional purpose of “promoting the sciences and the arts” (Imfeld, 2003). The Copyright Act of 1976 establishes a four factor test to determine whether a
use of copyrighted content constitutes fair use. A court must weigh all four factors when determining
whether a specific use is protected by the fair use doctrine:

1) The purpose and character of use.
2) The nature of the work.
3) The amount and substantially of the portion used in proportion to the copyrighted work as a
whole.
4) The effect of the use upon the potential market for or value of the copyrighted work.
   (U.S.C. Title 17, Chapter 1, Section 107).

According to Fred Lohmann of the Electronic Frontier Foundation (EFF), there are three
reasons why fair use is important. First, many day to day activities would be illegal if it were not for
fair use. Whistling a copyrighted song in public, for example, would be in violation of copyright law
(as it would constitute a public performance). Without fair use, first amendment free speech rights
would be severely limited. Second, since fair use is ambiguous, it has to be evaluated on a case to case
basis. With each emerging technology, fair use must be renegotiated. If this were not the case,
copyright law would prevent many innovative new uses for technology from emerging. Third, fair use
promotes competition. When a new technology creates a new application for fair use, a competitive
market is opened and entrenched market powers are forced to compete (Lohmann, 2002).

The Digital Millennium Copyright Act (DMCA) was passed in 1998. One provision of this law
is the prohibition of the circumvention of copy protection mechanisms such as Digital Rights
Management (DRM) (See U.S.C. Title 17, Chapter 12, Section 1201). Unfortunately, this section of
the DMCA poses a major threat to fair use, restricts the free flow of information, and has the potential
to stifle both technological innovation and the creation of new content which makes fair use of existing
content. While the original stated purpose of copyright law was to “promote the sciences and the arts”,
the stated purpose of the DMCA was to “facilitate the robust development and world-wide expansion
of electronic commerce, communications, research, development, and education in the digital age”
(Imfeld, 2003). Coupled with DRM, the DMCA limits the access of creators to creative inputs. Landes
& Posner (1989) argue, however, that the cost of creative inputs needs to be taken into account in the evaluation of the economics of copyright law. While the DMCA succeeds in maintaining the commodity status of intellectual property, it does so at the expense of public access to information, fair competition, and fair use.

Digital Rights Management (DRM) is a set of technologies which control access to digital works. The following definition of DRM was given by H ter Doest et al. (2004):

DRM is the process of defining, managing and enforcing the usage rights of digital content. The function of DRM also includes that the legal rights are observed of and by all participants engaged in the electronic business and digital distribution of content and services.

Thus, this added layer of technological and legal protection seriously threatens fair use and changes the relationship between content producers and consumers.

Fred Lohmann argues that it is unlikely that any DRM system could fully protect fair use rights. While the content industry would argue that DRM is important for protecting content from piracy, it is important to answer the question of what the public has to gain from DRM. In other words, the public is being asked to give up some of their fair use rights so the content industry can protect their content. This both has the potential to limit the free flow of information by imposing technological restrictions on access and use, and is likely to ultimately work against public interests (Lohmann, 2002).

At this point we have explored the current legal framework of copyrights along with some discussion as to how the law got to its current state. We have also briefly discussed criticisms of copyright law. The letter of the law, however, is only part of the picture. In order to fully understand the current copyright environment, it is important to explore how courts have interpreted the law.

**Case Law Relating to Video Games and Copyrights:**

As video games have become prevalent, litigation involving video games has become increasingly common. Initially, it was unclear how copyrights would be applied to video games. Over
time, however, case law eliminated much of this ambiguity. While it is impossible to review all the case law on copyrights applied to video games, four cases in particular are worth noting, as they help establish the scope of copyrights as applied to computer programs in general and video games in particular; and help identify what constitutes fixation as required for copyrightability.

Williams Electronics, Inc. v. Artic International, Inc. (685 F.2d 870, 3d Cir. 1982) deals with the copyrightability of video games as audiovisual works as well as with the fixation requirement for copyrightability. Williams Electronics was the copyright holder of a coin-operated arcade game called “Defender”. As this case predates the Copyright Office’s decision (discussed above) that computer programs should be registered either as literary works or as audiovisual works (but not both), Williams Electronic had registered three copyrights pertaining to the game. One copyright was for the program itself (as a “literary work”), one for the audiovisual components of the “attract mode” (the sequence of sounds and images created for the purpose of enticing people to play the game), and one for the audiovisual components of the game’s “play mode” (in which the player interacted with the program). Artic International had produced a game virtually identical to Defender in all three regards (the program, the attract mode, and the play mode). The court held that Artic had infringed on all three of Williams’ copyrights. Artic had argued that object code (code which has been translated into machine language), as opposed to source code, is not copyrightable as it is not directly readable by humans. The court rejected this argument as the results of object code run on a machine are perceivable by humans. Artic also argued that as the audiovisual components of the game in “play mode” are in part a result of actions by the user and are likely to be different every time the game is played, the “play mode” is not copyrightable. The court rejected this argument as, although there would be differences between each game played, various audio and visual components would always be present, and certain patterns would be observed regardless of actions by the user. Artic also argued that, as both the program and the audiovisual components were embedded on ROM chips, which constitute uncopyrightable
“utilitarian objects”, there could be no copyright in the work. The court rejected this argument as Williams Electronics was claiming copyright interests in the data stored on the ROM chips, not the ROM chips themselves. An additional argument made by Artic and rejected by the court was that as game play depended on user input, and each game was transient, the “play mode” did not meet the fixation requirement for copyrightability. This argument was rejected because the court saw the “play mode” as an audiovisual work as a whole; and while play differed from game to game, enough remained the same as to meet the fixation requirement.

Although not about video games, Apple Computer, Inc. v. Franklin Computer Corp (714 F. 2d 1240, 3d Cir. 1983) is significant because it sheds more light on computer programs as literary works. Furthermore, it adds more strength to the precedents set in Williams v. Artic in regard to the copyrightability of object code and the fixation requirement for copyrightability. Franklin Computer was manufacturing clones of Apple’s Apple II computer. In order to achieve interoperability, Franklin Computer copied operating system software from the ROM chips of the Apple II. Franklin Computer argued that computer programs should not be considered literary works, and therefore are not copyrightable; that object code is not directly readable by humans, and therefore is not copyrightable; and that as the programs in question had been fixated on ROM chips, which Franklin Computer argued are non-copyrightable utilitarian objects, the copyrights on the programs were invalid. The court rejected all these arguments. In the opinion of the court, programs could indeed be classified as literary works. Furthermore, although object code is not directly readable by humans, it can be used by humans when mediated through a machine. The court also held that, while ROM chips are indeed utilitarian objects, it is the program embedded on the ROM chips, not the ROM chips themselves which is the subject matter of copyright. In fact, the act of embedding the programs on a medium such as ROM chips meets the fixation requirement for copyrightability.

Midway Manufacturing Co. v. Artic International, Inc. (704 F.2d 1009, 7th Cir. 1983) is an early
case dealing with derivative works as applied to video games. In this case, the defendant (Artic) created a modified circuit board which executed the instructions for the plaintiff’s (Midway) game Galaxian faster than Midway had originally designed the game to be run. It should be noted that Artic had not actually modified the code of the game, or the resource files located on the ROM chips; but rather was simply providing a modified circuit board used for executing the code. Midway argued that when arcade owners used the modified circuit board in coin operated machines, the output of the speeded up audio and visual components of the copyrighted game constituted a direct violation of the public performance rights to a derivative work. As Artic sold these circuit boards for the purpose of speeding up Galaxian, Artic’s activities constituted contributory infringement. Artic argued that speeding up a video game was analogous to playing a vinyl record at a higher speed than at which it was recorded. The court rejected this argument. While the court acknowledged that playback of a vinyl record at an increased speed probably would not constitute the performance of a derivative work; the court noted that there was not significant commercial demand for speeded up vinyl records, while commercial demand did exist for speeded up coin operated arcade games (as the games are more exciting, plus last for a shorter time as to allow arcade owners to generate increased revenues). The court agreed with Midway that Artic was guilty of contributory infringement for the public performance of a derivative work.

Lewis Galoob v. Nintendo (964 F.2d 965, 9th Cir. 1992) also addresses video games, but in this case, the 9th Circuit considered a similar fact pattern to what the 7th Circuit addressed in Midway v. Artic, but reached the opposite conclusion. This case was an appeal to the decision that Galoob’s product the “Game Genie” did not create an infringing derivative work of games for Nintendo’s NES (Nintendo Entertainment System). The Game Genie was a device which interfaced both with the NES and games for the NES and allowed users to enter codes which instructed the system to read user specified memory addresses on the ROM cartridge as having different values, thus altering gameplay
and/or the audiovisual output of the game. The effects of the device were temporary, and the alterations were lost once the system was turned off. Nintendo argued that the Game Genie created a derivative work of games licensed to play on the NES. The court disagreed with Nintendo’s assertion. In the view of the court, no new work was created by the Game Genie; rather, the Game Genie simply temporarily modified the operation of the games (analogous to a kaleidoscope temporarily modifying the appearance of a painting). Thus a derivative work was not created. Furthermore, the court argued that even if a derivative work were created, the temporary creation of this work in a home for the individual enjoyment of the user would constitute fair use. Thus, despite the obvious similarities between this case and Midway v. Artic, the court reached the opposite conclusion.

Now that we have established the nature of copyright law, common criticisms of the law, and how courts have applied the law to video games, we now shift our focus to attempts to remedy the situation. In particular, we discuss how the copyleft movement offers an alternative to the exercising of exclusive rights granted by the law.

**Introduction to the “Copyleft” Movement:**

The copyleft movement can be said to have its ideological roots in the early computer programming culture prior to 1965-1970. During this time period, software development was a remarkably open endeavor. Code was generally unprotected by copyrights, and programmers were free to share code and collaborate with programmers from other businesses and industries. It was this ideology which ultimately inspired hacker/developer Richard Stallman to develop his GNU (a recursive acronym which stands for GNU's Not Unix) code base and GNU GPL (General Public License). However, this open development environment didn't last long. From 1965-1970, software development began to become increasingly institutionalized as corporations realized that code was a valuable economic commodity. This change accelerated in the 1980s with the rapid diffusion of the personal computer (Bradley, 2005).
Chopra & Dexter (2007) outline one significant turning point in the transition of the computing culture from an open to a closed environment. In 1976, the Homebrew Computer Club was a vibrant community of hobbyists excited by the emerging personal computer scene. This year, Bill Gates (who founded Microsoft along with Paul Allen) sent an open letter to this club blasting computer hobbyists for their attitude that copyrighted software could be freely distributed. He bemoaned the fact that the BASIC language implementation for the Altair personal computer which he and his friend Paul Allen had developed was being widely copied without the payment of copyright royalties. Gates contended that without adequate copyright protection, high quality software would not be developed. Chopra & Dexter argue that this represents the beginning of an ideological schism within the hacker culture between those who saw software simply as a commodity to be sold and those who saw it as an emerging set of tools which would benefit all if freely exchanged.

The final stimuli which prompted Richard Stallman to start the modern copyleft movement with his GNU GPL occurred during the early 1980s. Stallman was working on developing a Lisp interpreter (a computer language specialized for artificial intelligence programming) at MIT AI Lab when there was a split in his development team. The breakaway group asked Stallman for access to the source code (which Stallman considered to be in the public domain) to improve upon it, and Stallman consented. When Stallman asked for access to the modified code, however, the breakaway group declined to give him access, and insisted that their modified code was copyright protected. In response, Stallman decided to develop a license which would mandate that derivatives of code made free to the public would also be published under similar terms (Dinkgrave, 2005). In 1983, Stallman introduced the GNU GPL (Stallman, 1983), and in 1985 he coined the term “copyleft” (a term meant to oppose copyrights and promote the ideology that information should be free to access and modify) and founded his non-profit Free Software Foundation (FSF) whose goal was to promote publication of software under his copyleft license (Dinkgrave, 2005).
Richard Stallman discusses his motivation for writing the GNU license in his GNU manifesto:

**Why I Must Write GNU**

I consider that the golden rule requires that if I like a program I must share it with other people who like it. Software sellers want to divide the users and conquer them, making each user agree not to share with others. I refuse to break solidarity with other users in this way. I cannot in good conscience sign a nondisclosure agreement or a software license agreement. For years I worked within the Artificial Intelligence Lab to resist such tendencies and other inhospitalities, but eventually they had gone too far: I could not remain in an institution where such things are done for me against my will.

So that I can continue to use computers without dishonor, I have decided to put together a sufficient body of free software so that I will be able to get along without any software that is not free. I have resigned from the AI lab to deny MIT any legal excuse to prevent me from giving GNU away. (Stallman, 1983).

In 1991, Open Source Software/Free Software (software published under copyleft licenses) moved from obscurity into the mainstream when Linus Torvalds released the first version of the Linux operating system. Linux is heavily backed by the much less idealistic and much more business friendly Open Source Initiative (OSI) lead by Eric Raymond (Bradley, 2005). The ideological differences between Stallman and Raymond will be discussed in much more detail shortly.

Inspired by the Open Source/Free Software movement, and disturbed by perceived imbalances caused by legislation such as the DMCA and overzealous implementation of DRM, intellectual property experts Lawrence Lessig and James Boyle, among others, set out to develop a set of licenses which would extend the ideals of the open source movement to other forms of creative works. The result was the Creative Commons movements, which consisted of a variety of licenses which gave publishers flexible control over what rights, if any, to reserve. This allowed publishers to essentially copyleft content and decide what terms of use and restrictions to attach to their works (Flew, 2005). Since the founding of the Creative Commons, a related effort, the International Commons, has sought to extend Creative Commons licenses into the legal frameworks of other countries. Since each country has its own copyright laws, changes must be made in the wordings of the licenses to fit into other legal frameworks. As of May 2008, the International Commons had developed licenses for forty-five
countries, licenses for nine additional countries are in the works, and plans exist for the development of licenses for many more countries (http://creativecommons.org/international/).

As there are about 60 different licenses which could be considered copyleft licenses (Gunzburger, 2005), I will not provide a comprehensive list of all copyleft licenses available. I will instead give an overview of some of the most important of these licenses.

The original copyleft license, and one of the most commonly used licenses is the GNU GPL license. This license allows for the modification and redistribution of source code under the stipulation that the modified code is distributed under the same license terms. Another important attribute of this license is that any additional source code in a project which uses GPL code must also be covered by the GPL. Therefore, if someone were to use just a little bit of GPL code in a large software project, the whole project would be covered by the GPL, thus making the license viral (Gunzburger, 2005).

The Lesser GPL, or LGPL license, has identical terms to the GPL, except it allows for the distribution of proprietary copyrighted code along with copylefted code. It is thus less viral (Gunzburger, 2005). The Berkeley Standard Distribution (BSD) license, which is largely supported by the more ideologically conservative OSI, is similar in terms to the LGPL and is therefore also non-viral (Chopra & Scott, 2007).

Also developed by Stallman's Free Software Foundation is the GNU Free Documentation License. This license was originally designed for covering documentation bundled with software, but far more relevant is the fact that this is the license Wikipedia (a free and open Internet encyclopedia which anyone can contribute to) uses. Wikipedia predates the Creative Commons initiative, and the GNU Free Documentation License was the best copyleft license available at the time of its conception. The two most relevant terms of this license are that content published under it should be free for modification and reproduction, and that information published under this license should be available in a format which facilitates modification (Flew, 2005).
While the previously mentioned licenses pertain mostly to software or software related projects (as is the case with the GNU Free Documentation License), the Creative Commons (CC) license is designed particularly to allow for copylefting of creative content (such as pictures, music, TV shows, books, etc.) in digital format. Some baseline features that all CC licenses share are that the public is free to copy, perform, and redistribute, or make a literal verbatim copy of the work in another format; that the CC license is irrevocable; that the CC work cannot be restricted by DRM; and that the original author must be given attribution. Optional features which differentiate CC licenses include the “Non-Commercial” feature, which stipulates that only the original author is allowed to make commercial use of the work or derivative works unless permission is granted; the “No derivative works” feature, which prevents others from producing derivative works without the permission of the original author; and the “Share alike” feature, which stipulates that derivative works can be produced only if they are distributed under an identical license (this feature is similar to the conditions of the GNU GPL) (Fitzgerald, 2005).

Another set of “licenses” which deserves attention is the Libre Commons initiative. These “licenses” are not true legally binding “licenses”, as the authors of these “licenses” completely reject the concepts of copyrights, law, and nation states. They are rather promises to waive all rights to a work and dedicate a work as common property of all humanity. There are two separate “licenses” available under the Libre Commons flag. The Libre Commons Res Communes License states “This license declares your work to a common that is shared between us as human beings. It is therefore owned in common with others.” The Libre Commons Res Divini Juris License states “This license declares your work to the realm of the gods. Where as a moment of clearing it contributes to a permanent state of exception rejecting state law and liberal conceptions of the nation state.” (Libre Society, 2006).

Just as there are a variety of copyleft licenses, there are a number of different parties with
different motivations and ideologies promoting copyleft. In order to explore the spectrum of ideological viewpoints, it is worthwhile to give an overview of a few of the major debates within the movement.

In some countries, in addition to economic rights to licenses, authors are granted the “moral rights” of attribution to and integrity of their works. Under some legal systems these right cannot be waived. However, not being able to waive moral rights can rule out the possibility for others to make derivative works or make “remixes” of works published under CC licenses. This seriously limits the extent to which CC licenses can be utilized in some countries (Fitzgerald, 2005).

At this point, it is necessary to make a distinction between “Open Source Software” and “Free Software”. From here on, when I discuss Free Software and Open Source Software together in a generic sense I will use the term “FOSS” (Free/Open Source Software).

A major ideological divide exists between Richard Stallman's (founder of the Free Software Foundation, or FSF) vision of copyleft and Eric Raymond's (founder of the OSI) vision. In fact, Eric Raymond probably would not even use the term copyleft. As is apparent from the names of the two organizations, Richard Stallman prefers to use the term “Free Software”, while the more business friendly Eric Raymond prefers the term “Open Source Software” (Bradley, 2005). It should be noted that this distinction is purely ideological, and isn't directly related to terms of FOSS licenses; although, as noted by Chopra & Dexter (2007), viral licenses such as the GNU GPL tend to be preferred by the Free Software movement while the Open Source movement is more willing to accept non-viral licenses such as the BSD license.

Dale Bradley (2005) describes the ideology of the Open Source/Free Software movement as “Anarcho-Utopian”, but explains that these terms mean very different things to the two main ideological leaders, Stallman and Raymond.

Richard Stallman identifies more strongly with the “utopian” aspect of FOSS than with anarchy.
For Stallman, it is a moral value that information should be free (he makes the distinction between “free as in speech”, rather than “free as in beer”, with “free as in speech” being mandatory for Free Software. In practice, however, most “Free Software” is free in both senses). Development of source code which can be freely distributed and modified is not just a means to an end, it is an end in itself. Stallman has stated that he would rather use a less advanced piece of software that can be freely modified and distributed than a more advanced piece of proprietary software (Bradley, 2005).

Eric Raymond emphasizes the “utopian” aspect of FOSS significantly less than Stallman. To him, it is only utopian in that the open source model allows for the development of technologically superior software in a more cost effective manner. He believes that by allowing source code to be freely modified, the end product will be superior. He also sees the open source model as compatible with business interests (Bradley, 2005).

Raymond focuses more on the “anarchic” aspect of FOSS than Stallman. The most important aspect of open source development, according to Raymond, is the decentralized and non-hierarchical organization of open source development communities. Raymond's conception of anarchy is highly individualistic, and focuses on maximizing personal liberty. Through this individualistic and anarchic mode of production, Raymond believes the result will be technologically superior software. He also sees this model as useful for generation of profit. People will engage in FOSS development out of self interest, and private profit is a legitimate ambition for developers under Raymond's model (Bradley, 2005).

Stallman's conception of anarchy is more social-democratic. Individual freedom, to some extent, must be sacrificed for the good of the whole development community. In Stallman's model, individual software developers may sacrifice some individual freedom for the promotion of the common cause of developing free software. While development communities in Stallman's model are still more decentralized and less hierarchical than proprietary development communities, some
hierarchies may exist to keep the development on focus. In Stallman's model, people engage in the
development of Free Software for the benefit of all software users, with the motivation of sharing and
benefiting society at large (Bradley, 2005).

Chopra & Dexter provide further insight into the ideological differences between “Open
Source” and “Free Software”. Rather than challenging the underlying assumptions of capitalism, the
“Open Source” movement simply believes that, in general, the use value of software takes priority over
its exchange value. Profits can be generated through the provision of services and customizations
rather than through sale of software. In the view of the “Open Source” movement, allowing for freer
exchange of knowledge simply allows for greater efficiency in software production. Furthermore, it is
acceptable to sacrifice some amount of “freedom” and “openness” (such as by using non-viral licenses)
in order to gain corporate acceptance of what the Open Source movement sees as a superior mode of
production of value. For the Free Software movement, on the other hand, such compromises are seen
as opening the door both to exploitation of labor and co-optation of the movement to promote freedom
of access to information. In the view of the Free Software movement, many firms who use non-viral
licenses are essentially exploiting free labor of hobbyist coders in order to improve their own
businesses without giving anything back to the movement. Furthermore, to the Free Software
movement, the fundamental ideology is that users should be free to modify and share all code. Another
principle of the Free Software movement is that property rights should not be applied to intangibles
such as source code. Property rights are a utilitarian tool to allow greater efficiency in the presence of
scarcity, not a fundamental right. Proprietary software licenses actually create scarcity, both by
excluding people from using software, and by preventing the creation of new software from existing
source code; ultimately leading to monopoly (Chopra & Dexter, 2007).

Currently Raymond's model seems to be dominating the FOSS movement (Bradley, 2005). There is much corporate interest in projects such as the Linux operating system, and Sun's
As noted earlier, Lessig was inspired to start the Creative Commons movement by perceived imbalances in the intellectual property system. He believes that trends such as DRM and the DMCA shifted the balance too far toward private interests and away from public interests. Creative Commons licenses allow authors to decide which, if any, rights to reserve.

David Berry and Giles Moss (2006), the founders of the Libre Commons movement, harshly criticize the Creative Commons movement. First, they state that Lessig's CC initiative will not be able to stand up to the profit driven intellectual property system, and is likely to be co-opted by corporate intellectual property holders. They point to the fact that the CC initiative has received a $25,000 donation from Microsoft, and make the following statement:

For us, Lessig's particular understanding of the world, and his desire to strike a balanced bargain between the public and private that follows from this, appear naive and outmoded in the age of late capitalism. We follow the political economists instead, and trace the association they draw. Capital is continually rendering culture and communication private, subject to property rights and the horror of commercial exploitation and beautification. When immaterial labour is hegemonic, the relationship codified in intellectual property between the 'public' and 'private', between labour and capital, becomes a crucial locus of power and profit. And it is quite natural that private interests would want to protect and extend this profit base at all costs. Their existence depends on it. If libre culture or the Creative Commons threatens this profit base in any way, wars of manoeuvre and position will ensure, where corporations and the state will set out either to crush or co-opt (Berry & Moss, 2006).

Another criticism Berry and Moss (2006) make of CC is that it is not a true commons. It treats culture as a resource, and allows for private rights to be retained. They argue that while CC licenses may allow sharing of culture, it is not a true commons. In a true commons, culture is commonly owned by all.

Yet another major criticism of the CC initiative is that it is based on law; specifically the law of a capitalist state. According to Berry and Moss, the nature of the struggle is not merely legal and economic, but is deeply political. In their view, attempting to divorce the issue of intellectual property
from political struggle is counterproductive. Rather than acting within the existing legal framework, political struggle is essential for challenging the current IP paradigm, and the capitalist model at large. For this reason the Libre Commons licenses, rather than seeking protection through the capitalist legal system, challenge its very existence:

Let's be clear: these “licenses” are political-democratic devices. We do not claim that they have legal authority. Indeed, our non-legal usage of the term licence (sic) has upset some lawyers and the like. They have lectured to us that our use of the term “licence” is 'wrong', 'incorrect' and 'contradictory'. It is not surprising that those, who retain power and status by claiming to speak 'correctly' and with 'rectitude' on other's behalf, would fear polysemy and flat-out deny our capacity to think or speak otherwise. It is not surprising that those who move in anti-political worlds of straight lines would want to deny our political capacity to contest and multiply meanings (Berry & Moss, 2006).

This leads to the question of how much these ideological differences matter to the larger copyleft movement and to FOSS in particular. My contention is that, on the one hand, these ideological differences will determine the direction the movement takes. A radical rejection of intellectual property and corporate commodification of software and culture will take the movement in a significantly different direction than coexistence or even cooperation with corporate forces. On the other hand, even if corporations or profiteers decide to use copyleft licenses as part of a business model for the purpose of generating profit, the information published will still be more accessible to the public and of more value to the public than traditional copyrighted expression. Furthermore, it is unknown how much impact the ideological copyleft movement can have on its own; non-ideological copyleft may be able to generate a significantly larger body of copylefted material than the ideological copyleft. Furthermore, while the ideology of radical copyleft (such as the ideology of the Libre Commons), is appealing, without legal protection it is unlikely to have a significant impact on the availability of copylefted information.

In my view, at least until copyright law is reformed, the corporate friendly (or even corporate backed) elements of the copyleft movement, the moderate factions of the copyleft movement, and the
radical copyleft can and should coexist. Corporate produced copylefted material is still of greater value to the public than corporate produced copyrighted material; and to the extent that the copyleft movement can influence how information is published, even by capitalist players, it is succeeding. The ideological copyleft should continue to oppose the trends of commodification of culture, and must, to the extent possible, resist co-optation.

While the supporters of copyrights argue that copyrights are necessary for capital and labor intensive works to be created, the success of phenomena such as Wikipedia, Linux, Mozilla Firefox, and many other large scale projects demonstrate that this is not always the case. Benkler (2006) notes that people collaborate for reasons other than making money, and, with the aid of technology, large scale collaborative efforts can lead to significant productivity.

The ideological perspectives on copyleft and FOSS presented in this chapter provide an important background for our discussion of FOSS video games. Since the dominant business model for video games focuses on generating profit through sales, the ideological perspectives of the Free Software movement may be more important for explaining motivations for FOSS games than the ideologies of the Open Source movement (the list of games in my sample at the end of Chapter 4 includes the license for each game). Nonetheless, traditional economic models also provide valuable insight into the workings of FOSS development and distribution. This is the topic of the next chapter.
Chapter 2: Economic Perspectives on FOSS Production, Consumption, and Policy

Thus far we have evaluated ideological perspectives on FOSS development. While such an approach provides part of the picture of why people participate in FOSS development and of the value of FOSS to the public, insight can also be gained by looking at FOSS through the lens of classical economics. This chapter will provide a literature review of research which has been conducted on the economics of FOSS. Perspectives on both the supply side and the demand side will be presented; and policy considerations for both will be discussed.

Before going into the literature on the economics of FOSS, it is worthwhile to point out that FOSS has become fairly prevalent, and thus cannot be ignored in relation to the economics of the larger IT industry. Wikipedia (www.wikipedia.org), a free online encyclopedia which has been written and edited entirely by user contributions, although not software, is an extremely significant example of a collaborative copylefted effort. As of July 4th, 2008, Wikipedia has more than 2.4 million English language articles along with smaller numbers of articles in many other languages. The success of Wikipedia demonstrates that through technologically mediated collaboration, a wealth of information far in excess to that found in traditional encyclopedias can be generated, edited, discussed, and verified.

Perhaps the best known and most significant piece of open source software is the Linux operating system. According to Tim Ansell (2008), “Linux revenues” (he failed to define this term) were $35 Billion in 2007, accounting for 10% of global software revenues for the year. The Linux operating system, according to Venture Development Corporation, a technology market research organization, has made significant inroads in the world of embedded computing and is used on devices ranging from cell phones to automobiles (Rommel, C. et. al., 2008). Furthermore, Steven J. Vaughan-Nichols (2007) of Linux-Watch notes that Linux plays a significant role in the server market. Another significant example of the highly successful FOSS is Mozilla Firefox (a popular open source web browser) which, in February 2008, Cnet reported had been downloaded half a billion times.
While this does not indicate that it is being used by half a billion people (it is likely that many people downloaded it multiple times), this is nonetheless significant. SourceForge, which refers to itself as “the world’s largest Open Source software development web site”, had 177,533 registered projects as of May 20, 2008. As of May 20, 2008, the five most downloaded programs alone had been downloaded a combined total of 837,613,796 times (interestingly, all five of the most downloaded programs on SourceForge are Peer-To-Peer file sharing applications). It should be noted that the actual number of times these programs have been downloaded may be significantly higher, as the cited figure only counts downloads from SourceForge, and not from other sites on which they may be hosted (although the SourceForge numbers for each project include all download from the project’s SourceForge page, and include downloads of source code in addition to binaries).

In addition to these impressive figures, it is worthwhile to note some additional examples where FOSS is having an impact. BOINC (an acronym for Berkeley Open Infrastructure for Network Computing) is an open source utility which allows users to donate spare computational resources to a wide variety of scientific endeavors including, but not limited to, the Search for Extra-Terrestrial Intelligence (SETI) and multiple projects working to better scientific understand protein folding. As of May 2008, the combined contributions of BOINC users are averaging more than a PetaFLOP (a quadrillion, which is a thousand trillion, floating point operations per second), thus making this computational effort equivalent to an extremely powerful supercomputer (http://boinc.berkeley.edu/). The Open Handset Alliance, a consortium of more than thirty technology companies (including major players such as Google and Intel), has developed a FOSS platform for mobile phones. This platform is called Android, and is based on Linux. Phones using this platform are expected to become available in the second half of 2008 (http://www.openhandsetalliance.com/). The statistics and projects cited above do not provide a comprehensive picture of the importance of FOSS. Rather, they simply serve to
demonstrate how significant FOSS has become.

Before diving into a review of journal articles dealing with the economics of FOSS, a discussion of research done by Yochai Benkler (2006) is appropriate, as his book *The Wealth of Networks* presents concepts which largely exist on the border of ideological and economic analysis. Benkler argues that society may be undergoing a transition from what he calls an “industrial information economy” to what he calls a “networked information economy”. Information goods (including, but not limited to, software), rather than being produced by firms, are beginning to be produced by networks of individuals. The key enabling factor for this transition is the emergence of digital communications technology which allows people to effectively collaborate over wide geographic areas. Benkler also discusses factors other than technology which influence the ability of decentralized networks of people to produce information goods. One significant factor noted by Benkler is that individuals often have non-monetary incentives to contribute to culture. At times, people will behave altruistically in order to feel good about themselves. In such cases, monetary incentives can actually dissuade people from participating in cultural production. Thus, while the exclusive rights granted by copyright law may have promoted information production in the industrial information economy, these same exclusive rights may actually discourage individuals from collaborating in a networked information economy. Furthermore, Benkler notes that in order to effectively contribute creativity to a collaborative project, creative inputs are required. Individuals need to be exposed to the works of others in order to create new works. This again implies exclusive copyrights may be counterproductive in a networked information economy. Also of importance, Benkler introduces the ideas of modularity and granularity (modularity will be discussed at multiple points in this thesis) as important factors in predicting which types of information goods can be produced by decentralized networks. Modularity is the property of how easily a project can be broken down into smaller tasks (thus in a non-modular task, division of labor is impossible). Granularity, as
defined by Benkler, is the size of individual modules. Benkler argues that a high degree of modularity accompanied by high granularity (meaning small modules) is highly conducive to networked information production. In addition to these properties, Benkler also argues that what he calls “institutional ecology” is crucial in determining what types of social production will take place. Institutional ecology, as defined by Benkler, refers to the legal, social, and economic environment in which cultural production takes place. In order for a successful networked information economy to emerge, steps must be taken to create a favorable institutional ecology.

**Articles dealing with the supply side:**

Lerner & Tirole (2000) attempt to apply a variety of economic theories to FOSS development. They evaluate what motivates individuals to participate in FOSS projects, how these incentives influence the quality of FOSS projects, why for-profit firms invest in FOSS development, and what unsolved economic questions remain about FOSS development.

Lerner & Tirole begin by asserting that although altruistic motivations for participating in FOSS development exist, altruism alone cannot explain why programmers participate in this process. Broadly, they argue that people will only engage in open source development if the benefits of participating outweigh the costs. The primary cost considered is opportunity cost (meaning that while developers are working on FOSS, they are likely to be unable to simultaneously participate in other endeavors). The benefits can be divided into immediate and long term. The immediate benefits identified are any monetary compensation received (if the developer is working for a firm), and the payoff related to the use value of their work to themselves (creating a useful tool or fixing a bug in an existing tool). Lerner & Tirole characterize future benefits as “the signaling incentive”, which they define as the ability to signal your skill to perspective employers (the “career concern incentive”) and/or to other developers (the “ego gratification incentive”).

According to Lerner & Tirole, signaling incentives can be used not only to broadly explain why
people participate in FOSS development, but can also explain various phenomena within the development process. They explain that the signaling incentive is greatest for prominent players (generally the people who started working on a project at the earliest date) and for participating in products individuals believe will become highly visible. Depending on the project, this can result in either a bandwagon effect or failure to reach critical mass. Thus this model explains patterns in which developers join or abandon a project. Another aspect of FOSS projects which Lerner & Tirole explain with the concept of signaling incentives has to do with the modularity of large projects. In an open source development environment, large projects must be broken down into manageable components called modules, which individual geographically dispersed developers can work on. Technologically sophisticated modules provide a more significant signaling incentive than do mundane tasks. This may explain why FOSS applications are often lacking in areas such as user interface and documentation while they are technologically sophisticated under the hood. (The question of whether having to break projects down into small modules would limit the ability of the FOSS model to produce certain types of applications was one of the unanswered economic questions presented by the authors.)

Lerner & Tirole also use economics to explain why for-profit firms sometimes invest in FOSS development or release their proprietary software under FOSS licenses. Firms may invest in FOSS development if they offer products or services which the open source model cannot provide (thus, FOSS increases demand for their products and services). They may also invest in FOSS development if they lack the ability to market their own proprietary solution, but wish to diminish the market share of a competitor. However, the article also suggests that corporate investment in FOSS may be limited by the ability of other firms to “free ride” on their investment.

Lerner & Tirole conclude that while economic theories can help explain observed phenomena in FOSS development, existing economic models lack the ability to adequately predict the future of FOSS.
While Lerner & Tirole provide insight on some of the economic incentives for FOSS development, the fact that they admit their models cannot adequately predict future trends in FOSS limits their usefulness. Application of the concept of modularity to explain why certain aspects of FOSS (such as documentation and user interfaces) are underdeveloped is a valuable insight. As we shall see, the concept of modularity has also been applied by others when exploring various aspects of open source development.

Baldwin & Clark (2006) apply game theory to discuss how code architecture (the way a software project is broken down into components and how the components interact) impacts labor distribution and efficient outcomes in FOSS development. Specifically, they look at how modularity (discussed above) and option value impact free riding within an open source development environment.

The primary concepts in Baldwin & Clark's argument are modularity, option value, free riding, and involuntary altruism. It is not necessary for me to explain modularity again, but the other three concepts require some explication. In the model presented in their article, option value is closely related to modularity. It refers to opportunities (options) to improve a project. Option value is said to be high if there are many options for contributing to a software project. Individual modules in a project can be implemented in a variety of ways; and not all implementations are equal. Thus a developer has the option of improving a software project by writing a new module or by writing a new (possibly superior) implementation of an existing module (thus, option value is generally positively correlated with modularity, although modularity is not the only factor in determining option value; the presence of multiple ways to implement individual modules also increases option value).

The concept of involuntary altruism refers to the non-rival nature of open source code. The usage of code by one person does not decrease the value of the code to another person. When a developer writes code, she not only creates value for herself, but also for others. This leads to the concept of free riding. In the model presented by Baldwin & Clark, anyone can reap the benefit of
code by choosing to join an open source project. However, individuals who have joined the project will reap the value of code which is written whether or not they contribute significantly to the code base. Contributing to the code base constitutes a cost to the individual in this model, so the net value reaped by a participant can be defined as the value of all the code developed within the project minus personal cost of contributing to the code base.

Baldwin & Clark argue that increasing modularity and option value within a project will change the strategies employed by developers in the “development game” in such a way as to decrease free riding and to produce both more net value and a more equitable distribution of labor. In a project with only one module and only one way to implement the module (low modularity and low option value), and two developers; each developer will be better off if the code is written than if it is not written; but each developer knows that she would be better off if the other developer writes the code. The only economically efficient outcome is if one and only one developer writes the code. If both developers write the code more labor will have been expended than necessary without any additional value being created. If neither developer writes the code, the value is not created. However, in the efficient case, the developer who does not write the code is better off than the developer who wrote the code. In a scenario in which there is more than one module, however, there is more opportunity for efficient and equitable distribution of labor. And if option value is high for each module, it is not inefficient for more than one developer to work on the same module, as whichever implementation works better will be included in the final distribution of the product (thus, the more implementations of the same module, the more likely it will be that a highly efficient implementation will be included in the final distribution).

In a nutshell, Baldwin & Clark present an analytical framework for evaluating how the code architecture will impact labor distribution, net value, and equity within an open source development project, therefore, when deciding on a code architecture for an open source project, it is necessary to
consider not only the technical merits of various architectures, but also how architecture will affect the ability to efficiently divide labor.

This model is valuable in that it can help project founders predict which code architecture will generate the most participation and least free riding in development. It could also potentially be applied in evaluating what type of FOSS projects are likely to succeed and which are not (to the extent that modularity and option value in code architecture are inherent traits for certain types of software). There are, however, serious limitations to this model. First, it ignores the signaling incentives and other incentives to participate in FOSS development. While this omission may have been necessary to keep the model reasonably simple, it should be kept in mind that signaling incentives and other incentives will somewhat skew the results in this model. Furthermore, the model only considers people capable of developing software. The layperson is unable to participate in development, but can still make use of FOSS (and thus there is an additional type of free riding not considered by the article; people who use the software but are not involved in the development). Also, the model only evaluates labor distribution within a project. It does not address the question on when a developer which choose to start a new FOSS project or choose between contributing one of multiple related program (although, by applying the basic ideas of the model, it might be reasonable to assume the developer will consider modularity and option value, along with the current state of each of the projects, in making this decision).

A parallel can be drawn between Baldwin & Clark and Lerner & Tirole. Lerner & Tirole noted that developers are likely to shy away from working on modules with low signaling incentives. It is fair to assume that, at least in some cases, high option value will also lead to more significant signaling incentives. Complicated code which requires a high degree of technical expertise will likely be able to be implemented in multiple ways, therefore possessing higher option value. However, high option value will not always predict strong signaling incentives. Important, yet relatively mundane modules,
such as documentation and user interface, can very often be improved on; and not all implementations are equal. Despite the relatively high option value of these modules, they continue to possess poor signaling incentives.

Twidale & Nichols (2004) note that poor user interfaces are a major problem for many open source projects. They also note, however, that because the FOSS development model is highly transparent, it is easy to study how discussions of usability and bug reporting are carried out between FOSS developers. In their study, they qualitatively evaluate examples of user interface discussion and bug reporting in various FOSS projects. The purpose of their study is simply to shed some light on the bug reporting process. They conclude, however, that while they have provided a description of the process, further research must be done to suggest improvements to the process. While this study has limited significance in the larger discussion of organizational structures in FOSS development, it does make the important point that the open nature of both the source code and the development process makes studying the process easier. It also makes it clear that fairly complicated and formalized social interaction protocols are in place for addressing issues of usability (and likely in all aspects of development).

While not discussed by Twidale & Nichols, the fact that FOSS development is highly transparent could lend weight to the theory that signaling incentives play an important role in motivating people to participate in projects (as the transparency is likely to make their contributions more visible and thus increase the incentives). This same transparency may also highlight where option value exists (programmers can see where improvement is required, and thus direct their efforts to fixing the bugs).

Bessen (2005) observes that a high percentage of open source development takes place within for profit firms. He observes that not all this development can be accounted for by traditional models involving using FOSS to increase the value of complementary goods and services. He develops a
model in which firms develop open source software to reduce costs and to allow for maximum customizability of software.

Bessen bases his model on the fact that firms require highly specialized software. Firms have the option of purchasing prepackaged software (sometimes bundled with APIs, Application Programming Interfaces), contracting a third party to develop the specialized software, or modifying existing open source software in house to meet the specialized needs. As prepackaged software is meant to meet the demands of a wide base of users, it is often not sufficiently customized to meet the specialized needs of firms. This can be partially addressed by including APIs (libraries for adding customized features), but these libraries still may not meet all the customization needs of a firm. The weakness of contracting with third parties for specialized software development has to do with the limited ability to adequately customize contract terms for the development of software. It is often impossible to include all the specifications of a piece of software in a contract, and frequently the buyer has to customize the contracted software after purchasing it. This leaves open source software. Open source software allows firms to customize preexisting software packages free of charge. Furthermore, because of the viral licensing of much open source software, the cumulative contributions of multiple firms to a FOSS project leads to a more complete and specialized code base for future firms to build on. Therefore, it is often most efficient for firms to customize (thus contribute to) FOSS rather than buying prepackaged software or contracting third party developers for specialized software.

Bessen's model is significant because it explains a massive amount of open source development which other models fail to address. Furthermore, this model can be used to explain which types of projects are most likely to be contributed to by firms. Projects which have high utilitarian value in for profit firms are likely to receive more development labor than other categories of software (such as video games).

Valimaki (2003) presents an open source business model not discussed in other literature on
FOSS. He argues that dual licensing is a viable business model in open source development. In a dual licensing arrangement, the firm developing the software can allow people to use the software under a FOSS license free of charge. However, under the FOSS license, any modifications made to the software must also be distributed under the same FOSS license. If, however, a firm wishes to release a proprietary software package based on the dual licensed software, they can purchase a license allowing them to sell their product as proprietary software. Valimaki presents three case studies of firms which use dual licensing as a business model. It is apparent that dual licensing is a viable business model when the product being produced can be modified or embedded in another application of commercial value. Therefore, dual licensing allows for any network effect inherent to the software be manifested because of the large number of users using the software under a FOSS license. Valimaki argues that the majority of people using the FOSS version are people who wouldn't be willing to pay for the proprietary license anyway, and therefore do not constitute lost sales. When network effects are present, the number of people using the FOSS version increases the value of the proprietary license.

**Articles dealing with the demand side:**

Comino & Manenti apply the “Hotelling model” to evaluate how different demand side government policies would impact social welfare. They first analyze policy implications when network effects are not present, then add network effects to their model.

In this model, users are categorized as either “informed user” or “uninformed user”. An informed user is defined as a computer user who knows about FOSS, while an uninformed user is defined as a computer user who is not aware of FOSS. For construction of the model, we assume there are two software products equivalent in their functionality. The only difference between the two is that one is free “as in beer” and open source, while the other is proprietary and costs money. Each software user has a certain valuation for each of the two software products (for this model, while users valuations are normally distributed, the valuations for each of the two products is, on average, equal.
Thus, all else being equal, users will be indifferent between using the open source applications and receiving the proprietary application for free. However, in making a decision on which software to use, both the valuation of the software and the cost of adopting the software are considered. The total cost of adopting one of the two software packages is defined as the cost of the product (in the case of the proprietary software some monetary fee, and in the case of the open source software zero) plus “transportation costs” (how much it costs to move to each of the products; for example if there is a greater learning curve associated with the FOSS product as opposed to the proprietary product, the transportation costs for adopting FOSS will be more than for adopting the proprietary software.) Furthermore, as noted earlier, users are defined as either informed or uninformed. Uninformed users cannot adopt FOSS even if their value – the net value (value minus cost) of FOSS is greater than for the proprietary software, they are only able to adopt the proprietary software because that is all they are aware of. Informed users must weigh the net benefit of choosing each piece of software, taking into account both value and costs (including transportation costs) for each of the two products.

Having established this framework, Comino & Manenti identify three distinct policy options. They evaluate the impact each of these situations will have on consumer welfare and market performance. Using the categorization of consumers as informed or uninformed, they discuss the concept of “second best” and “best” outcome. The second best outcome is the market equilibrium which results in a market composed of a mixture of informed and uninformed users. The best outcome is the market equilibrium which occurs with the idealized situation of all users being informed users. Therefore, any policy which addresses only behavior, without moving users from the uninformed to the informed group will at best only achieve a second best outcome.

The first policy option considered is to mandate that certain segments of society (such as government and educational institutions) switch to open source software. An effect of this switch is that uninformed users become informed users. This will only lead to an increase in consumer welfare
if a high enough proportion of those forced to switch had been uninformed users (it must outweigh the disadvantage to informed users whose net value for the proprietary software was higher than for the open source software).

The second policy option considered is through an educational campaign. The educational campaign is assumed to be costless for the purpose of the model. The effect of this policy is increasing awareness of the existence of open source software, thus changing some of the uninformed users into informed users. This will always produce a positive outcome, as it allows the user to choose which program to use by comparing net valuations.

The final policy option considered is subsidization of open source software (as this model deals with demand side economics, this involves paying users to adopt open source software, rather than funding the development of FOSS). The effects of this form of subsidization are always negative (assuming they are publicly funded) as they cost the public money while simultaneously moving market equilibrium away from the socially optimal (where the sum of supplier and consumer surplus is greatest).

After introducing each of these policy options when network effects are absent, mathematical simulations were run to determine the impact the presence of network effects (in which each of the two products has greater value when it has a larger user base) will have on the implications of each of the policy options. Their simulations conclude that when network effects are present, any outcome which would have been positive in the absence of network effects will be even more positive when network effects are present. Also, network effects will mitigate the negative effects when a policy would produce a negative result in the absence of network effects.

While Comino & Manenti provide a model elegant in its simplicity and useful in evaluating various policy options, the unrealistic assumptions made along with the absence of a wider set of considerations imposes limitations on this model as a tool for determining policy. Specifically, the
article only applies short term considerations. While the transportation cost of adopting FOSS may be high initially, in the long term they are likely to reduce dramatically (users will become more familiar with the software, thus the learning curve will be reduced. In some cases, however, differences in levels of customer support will persist). This will change the optimal equilibrium. Furthermore, the article does not discuss the fact that private firms may engage in anticompetitive behavior and rent seeking behavior. Future research should evaluate how decrease in transportation costs over time and anticompetitive behavior impact policy decisions in this model.

Ghosh (2005) presents some policy considerations when dealing with standards. The topic of open standards is closely related to FOSS. Ghosh defines open standards as standards which are computable with open source applications and which neither directly nor indirectly give any firm a market advantage. Gosh makes two policy arguments about open standards and FOSS. He argues that it should be mandatory for eGovernment services to use both open standards and FOSS. Part of his rationale is that governments have an obligation to promote, or at least not to harm competition. Using a proprietary standard would force users of government services to adopt proprietary software; thus giving the firm owning the standard an unfair market advantage and subjecting the public to rent seeking behavior. Furthermore, Ghosh argues that government has an obligation to save money over the long run. He argues governments should consider the long term costs of using proprietary software and not just the short term cost of transitioning to FOSS. Use of proprietary software can lead to being locked into a proprietary standard and being subjected to rent seeking behavior. Therefore, it makes sense policy wise to mandate eGovernment services to adopt open standards and FOSS.

The contribution of Ghosh to our understanding of FOSS is that he provides another dimension to policy considerations. Essentially, in the public sector using proprietary standards can harm competition in the long run.

Discussion:
FOSS has established a significant presence in the software market. This chapter provides an overview on a number of economic perspectives on the significance and implications of FOSS. In the concluding chapter of this thesis, both ideological and political economy perspectives discussed in the previous chapter and classical economic perspectives from this chapter will be used to explain the state of FOSS video games. These two distinct approaches, when taken together, provide a more complete picture of the dynamics of FOSS. It is questionable to what extent the policy implications discussed in this chapter can be applied to FOSS video games, as games, by their nature, are generally meant to provide entertainment rather than utilitarian value. Such policy decisions, however, relate to the broader question of how policy relates to creation of cultural products.
Chapter 3: Video Games and the Video Game Industry

Having already discussed in broad terms the copyleft movement and FOSS, the focus will now shift to a discussion of video games. This chapter will provide insight into the history, structure, and economics of the video game industry. For the purpose of this study, an important aspect of the economics of video games is the practice of exploiting user modifications to commercial games. The relevance of user modifications will be discussed in some depth in this chapter. Next, a brief discussion of the limited research on FOSS video games will be presented. At the end of the chapter, some sales statistics for both hardware (consoles) and software (games) will be presented.

History:

In order to understand the current video game industry, it is worthwhile to take a brief look at its history. Williams (2002) describes how video games evolved from experimentation by hobbyists to a global industry. Two of the earliest pioneers in video game development were Wally Higginbotham and Steve Russell. In 1958, Higginbotham created a simple tennis game which ran on the equipment at a government nuclear research facility. In 1962 Russel, an MIT graduate student, wrote the game “Spacewar” for his school's PDP-1 computer. This game was improved upon by students at other universities. In the 1970s, an industry for video games began to emerge. In 1972, a coin operated game called Pong became a commercial success, and in 1974, Atari began to achieve commercial success selling a home version of Pong which interacted with televisions. Throughout the later 1970s and 1980s a market for home video game consoles which could run third party software began to emerge. Atari initially dominated this market, but in the mid 1980s, Atari's sales plummeted due to mismanagement and poor quality control. The entire industry for home video games appeared to have collapsed. Shortly after this failure, however, a new generation of hardware was released by market entrants Nintendo and Sega, and the industry was reborn. From this time on, the video game industry has undergone cycles of decline and rebirth as cycles of hardware have emerged, matured, and
eventually replaced by a new generation. Over the years video games developed into a multi-billion dollar industry. Parallel to the successive generations of consoles from the 1970s onward, home computer hardware have also continually evolved and matured during this time. As PC hardware became more advanced, home computers became capable of supporting increasingly sophisticated games; thus an industry for video games developed on both dedicated hardware and on personal computers (Williams, 2002).

Structure:

Having provided a brief history of video games, focus will now shift to the structure and economics of the industry as it stands today. Despite the fact that this study focuses on FOSS games on a Windows platform, it is important to take an in depth look at the topography of the industry, including proprietary consoles. This discussion will cover three broad topics: the individual segments of the home video game industry along with the unique characteristics of each; the economic and of the various stages of production and distribution of both hardware and software, and finally (and perhaps most relevant for this study) the increasing significance of user generated modifications to commercial games.

Williams (2002) explains that the home video game industry can be divided into three broad segments: the console segment, the PC segment, and the handheld segment. While these segments are interrelated, each one has unique characteristics.

Williams explains that the console segment is the largest segment of the home video game industry, due largely to its high profit margins. Johns (2006) suggests an additional reason the console segment of the industry dominates is because the console manufacturers play an active role in marketing software (even third party software). Williams describes the console segment as oligopolistic yet fiercely competitive. Competition within the console segment remains fierce because of the non-interoperability of hardware. Each console manufacturer attempts to draw consumers to
their console at the expense of the other console manufacturers. Furthermore, the console manufacturers generate profits through software sales rather than hardware sales; thus they often sell consoles at or below marginal costs. The console manufacturers generate profit from third party software titles through licensing agreements with developers and publishers (normally a fee is charged per unit). The console manufacturers also manufacture the physical medium containing third party software at their own plants.

Williams goes on to describe the handheld segment of the home game industry. At the time Williams conducted his research, the only significant player in this segment was Nintendo; which Williams attributed largely to high barriers to entry. Since that time, however, Sony has also emerged as a major player in the handheld market; thus, because of the competition between Nintendo and Sony, it could be argued that the handheld segment now more closely resembles the console segment.

Finally, Williams describes the unique characteristics of the PC segment of the video game industry. An obvious characteristic is that software developers and publishers need not pay royalties to hardware manufacturers, as the PC is a relatively open platform. Because of the lack of licensing fees, the cost of developing games for the PC is significantly lower than for consoles. Because of these lower costs, fewer copies need to be sold for a firm to break even. Williams notes that, largely because of these lower barriers to entry, a large number of smaller firms develop games for the PC. Due to this large number of developers, both a larger quantity of games and a more diverse body of games are produced for the PC than for any other platform. Despite this large number of games, however, the number of games sold is lower than for consoles. According to the Entertainment Software Association’s 2007 “Essential Facts” report, in 2006 approximately 201.3 million console games were sold for a gross of 6.46 billion dollars; while only 39.4 million computer games were sold for a gross of 0.97 billion dollars (ESA, 2007).

In regard to the assertion that the PC segment of the videogame industry is only a fraction of the
size of the console segment (and in regard to the numbers cited above), however, it should be noted that ESA uses data from NPD Group (which will be discussed in more detail later) to estimate sales. Independent game developer and blogger Jay Barnson (2007), however, suggests that NPD Group’s figures for PC games, both in terms of units and in terms of dollars, are hugely underestimated. Barnson notes that NPD group only tracks retail sales, and tracks neither legal digital downloads of games or revenue generated through subscription fees. Bernson asserts that online subscription based games such as World of Warcraft are among the most popular games for the PC platform. Also, “casual games” (games popular with people who enjoy lighter gaming) are becoming an increasingly important segment of the PC gaming market, and are disproportionately distributed online rather than through retail (Barnson, 2007).

Both Williams (2002) and Johns (2006) discuss vertical integration in the home video game industry. As Williams' description of the vertical stages of the industry and the integration between them is simpler than Johns', Williams' framework will be discussed first, then the insights of Johns will be presented to complement this framework.

Williams describes the industry as consisting of five vertical stages. These stages are development, publishing, manufacturing, distribution, and retail. The development stage is where the game is designed and programmed. Publishers responsible for the marketing of the games, and generally own the copyrights to the games. In the console segment of the industry, the console manufacturers typically manufacture the games (they put the works on the physical medium). The distribution stage is the process of storing and delivering games to retail. Finally, retail is the point of sale to consumers. As the industry has matured, these stages, to varying degrees, have become vertically integrated. Publishers, for example, frequently own multiple development firms. Console manufacturers often develop and publish their own in house titles. Publishers often also control the distribution of their products to retail chains. While publishers don't typically own retail chains,
publishers will often pay retail chains to place their products at favorable places in the stores, or to put up promotional posters for software titles. The relevance of the importance of marketing for commercial games (and lack of marketing budgets for FOSS games) will become more apparent when the concept of user generated content is discussed. Without marketing budgets, it is likely that not only will fewer people know about (and thus play) FOSS games; without a large user base, it seems likely that FOSS games will have a more difficult time attracting skilled modders (people who modify existing games, either by creating new levels or altering other components) who could increase the value of the games.

While Johns (2006) focuses almost exclusively on the console segment of the industry, he nonetheless provides significant insight into the business of video game production and distribution. The purpose of his study is to apply the framework of global production networks to both the hardware and software sides of the industry. Johns provides a more in depth look into the entire production process of video games along with information on how different players in the process capture value and the power relationships between these parties.

While Williams (2002) identified a five stage process for the production and distribution of video games, Johns identifies seven stages, beginning with financing and ending in consumption. He also notes that all seven of these stages exist for both hardware and for software, although they take different forms. For the purpose of this thesis, only the software side is relevant. Rather than individually discussing each of these stages, they only require explanation at points in which they differ from Williams' conception. Johns notes a duel role for publishers: they both publish and finance game development, thus publishers play a role at the beginning of the chain as well as near the end. Also, while Williams describes development as a single stage, Johns breaks development down into two distinct stages; development and production. What Johns refers to as development comes before any code is written or any visual or auditory artwork is produced. This stage involves planning budgets and
designing the game. After this stage has been completed, production begins. In addition to programming and debugging, Johns notes that increasingly the film and music industries are playing a part in the production stage of video game development (which has implications for FOSS games, as inclusion of copyrighted music and video clips would require payment of royalties, and would be incompatible with FOSS ideology).

Johns goes on to provide information on how much of the retail value of games each player in the production chain is able to claim. On the console side, 20% of retail value goes to the manufacturer, together the developer and publisher are able to capture 40% of the value, 10% goes to the distributor, and 30% to the retailer. On the PC side, because of the absence of licensing fees the distribution is different. Retailers capture 30% of the value, distributors capture 10%, and developers and publishers collectively capture 60% of the value. Johns notes that arrangements between publishers and developers are typically quite opaque, as the specific terms of their agreements are generally not publicly available; it is therefore impossible to know exactly how much value each of these two parties captures.

The reason for this distribution of value capture is related to the power relationships of the actors in the production chain. Naturally, each player would like to capture a larger share of the retail value, and will make use of whatever leverage they have over other players in order to capture more value. The power relations discussed by Johns include the relationships between console manufacturers (who also, as noted above, manufacture the physical medium containing the software) and publishers and developers as well as the relationship between console manufacturers, publishers, and retailers.

Developers are dependent on publishers for financing; thus they are in a weaker position to negotiate, which allows publishers to control the property rights to games. However, developers are able to improve their bargaining power by establishing a strong track record for producing good games.
Through interviews, Johns notes an interesting aspect of the dynamic between publishers and developers: The developers see themselves as being the true creative force and the publishers simply as business people. In fact, one anonymous developer interviewed by Johns went as far as to say that developers do what they do because they love making games, and rarely achieve large financial rewards; and it is the publishers who are able to make large profits from their works. This dynamic also has implications for FOSS development in that it demonstrates the motivation for developing games is not purely a desire to profit.

Console manufacturers are also in a fairly strong negotiating position. Whether they deal with publishers or directly with developers, Johns notes that the console manufacturers ultimately have the authority to decide whether the games will actually be manufactured. Each of the console manufacturers has a different policy as to what degree they exert their veto power over game production. Sony has a relatively hands off approach, enforcing only technical specifications; while Microsoft exerts more control over what games can be produced. Furthermore, at the beginning of a console's life cycle, when console manufacturers are trying to establish a library of games to add value to their console the publishers and developers have more negotiating power; while later in a console's life cycle power shifts to the console manufacturer.

As developers, publishers, and console manufacturers ultimately depend on retailers to reach the consumers, this gives the retailers a degree of negotiating power and thus allows them to capture a significant portion of value. The degree to which a game succeeds largely depends on how actively retailers promote a game; and retailers will more actively promote a game when they are able to capture a significant portion of its value.

The literature of Williams and Johns, in addition to simply describing how games are produced and distributed, also implicitly provides some insight into the viability of FOSS games. While the development of games is indeed expensive, a large portion of the money generated by the industry goes
to providing profits to a large number of players. As FOSS games generally are not advertised through traditional channels and can be downloaded directly from either the developer’s web site or from web sites such as SourceForge; there is no need for publishers, manufacturers, or retailers. Furthermore, the development of commercial games often has the additional expensive of royalties. Royalties will add significant costs to sports games which feature real teams and players and games based on movies. Sound tracks can also add additional expenses. This has at least two implications for FOSS games: on the one hand, it will limit what types of FOSS games will be produced as licensing fees for proprietary content is incompatible with FOSS (unless only the source code, but not the content, is open source); but on the other hand this will also significantly cut the costs of FOSS development. The flip side, however, is that FOSS games are less likely to attract large audiences due to the lack of advertising. As will be seen later, a lack of audience can hinder further development of a game because fewer people will create modifications and expansions for an unpopular game.

The Role of User Generated Content:

While some light can be shed on FOSS games from a general discussion of the video game industry, special attention should be given to the importance of user generated modifications and content to commercial games. This topic is of relevance both for the purpose of understanding the business model of the video game industry and for understanding innovation in video game development when capital incentives are absent. Arakji & Lang (2007) explain how the video game industry attempts to profit from user generated content while mitigating risks of outsourcing labor to consumers. Postigo (2003) discusses in more detail why consumers participate in unpaid development and makes a case that the outsourcing of production to consumers amounts to exploitation.

Arakji & Lang (2007) approach the issue of user modifications in terms of their value to the video game industry. For the most part, however, they ignore the question of what users get out of the act of modifying games. Likewise, they fail to critically evaluate the morality of the commodification
of unpaid labor. According to Arakji & Lang, in today's media environment it is costly and difficult for media firms to identify and adapt products to the ever changing and heterogeneous needs of consumers. At the same time, a large number of video game consumers have the technical ability and motivation to modify commercial games. This is beneficial to the industry both because it adds value to products and because it allows the products to be better adapted to ever changing and heterogeneous consumer needs. Arakji & Lang explain that user modifications became a significant part of the business model of the industry in the mid 1990's with the release of Id Software's game Doom. Id Software realized that the modifications hackers were making to the game significantly increased the value of their product; this prompted them to release the source code for the game under the GNU GPL license in order to prompt additional modifications. Due to the success of user modifications of Doom, when id Software released their next game Quake, in addition to making it open source, they included development tools with the game in order to further promote modifications (as will be discussed later, the FOSS Quake engine has been heavily modified and re-branded as the “DarkPlaces” engine, and over the years has developed to the point where it provides FOSS developers a valuable tool for creating commercial quality games). Later, Valve Software, developer of the game Half-Life, also decided to take advantage of user modifications. While they did not make the game completely open source, they made a large portion of the source code available for modification. Because of the modifiability Half-Life, modders were able to develop a completely new game based on Half-Life called Counter-Strike. Valve software proceeded to pay these modders a relatively modest sum for their work and went on to repackage this modification as a stand-alone software title. This was highly profitable for Valve for a couple of reasons. First, as the modification was already fairly popular before they purchased it, there was little risk that their repackaged version would end up being a flop. Also, the money they paid to the modders who made Counter-Strike was significantly less than the cost of developing the software themselves would have been (Arakji & Lang, 2007).
Despite the fact that user-generated modifications have the potential to add significant value to commercial games, Arakji & Lang note that there are also risks associated making source code available to consumers. An alternative to making source code generally available to the public would be to share the code with a single third party and contract with the third party to develop add-ons and modifications. The advantage to this approach from the perspective of the game developer would be a reduced risk of competitors including portions of the code in their products; or others piggy-backing on the firm’s source code and selling modifications for profit. However, while contracting with a third party rather than making code generally available may possess fewer risks, it also lacks the advantage of having a large number of enthusiastic end users providing free labor to increase the value of the software. Firms have developed different strategies to mitigate the risks involved in promoting user modifications. Id Software, for example, released their code under the GNU GPL, so any source code modifications would also have to be released under the same license, thus limiting the ability of third parties from profiting from the work of Id software (although it may ultimately turn out that FOSS derivatives of this code, such as the DarkPlaces engine, may provide significant competition to commercial software). Valve, on the other hand, only released limited portions of their source code for Half-Life, so users would have to purchase the original game in order to make use of modifications. Firms also make use of End User License Agreements (EULAs) to mitigate risks.

Postigo (2003) takes a different approach to evaluating the phenomenon of user modifications to commercial games. He focuses more on the motivations of users for engaging in this form of unpaid labor, and evaluates the relationship between firms and modders, analyzing to what extent this relationship amounts to exploitation. He places the broad phenomenon of user modifications within the framework of a “Post-Industrial economy”, in which much of production is no longer centralized, but distributed throughout society at large. He argues that while the hobbyists who engage in modding can be seen as participating in a “gift-economy”, firms go on to commodify the fruits of their labor.
Postigo identifies several reasons why people participate in unpaid modding work. While the motivations for participating in modding can be both altruistic and self-serving, Postigo notes that, in general, these people have little interest in controlling the intellectual property rights to their works. One possible motivation presented by Postigo is simply the sense of community and social gratification gained from the collaborative process common in development of mods. Furthermore, producing these modifications is an excellent way to derive experience and technical skills. Also, in general modders simply enjoy making games; and Postigo argues that this reflects a sense of ownership of the process and non-alienation from their creative endeavors. In addition to these motivations, Postigo notes that “signaling incentives” for future employment may often be a significant motivation for some modders. He notes, however, that the very act of modding may ironically reduce employment opportunities: firms will be less likely to hire workers when a significant portion of the development process can be outsourced to unpaid workers.

Postigo notes that from a labor standpoint, modding adds a significant amount of value to commercial games. Many user generated modifications are highly complex and represent a large amount of labor; labor which would be expensive and time consuming to perform in-house. Furthermore, commercial game developers have even been known to repackage and sell user-generated modifications as stand-alone software (as exemplified by Valve's marketing of Counterstrike, which they paid significantly less for than what it would have cost them to develop it in-house). Repackaging mods as stand-alone products not only allows the commercial firms to more fully capture the additional value of the creative work, but also greatly reduces risks, as the firms have the luxury of only packaging already successful mods as stand-alone products, thus reducing the probability of the product being a flop. The exploitation of this free labor, however, goes beyond appropriating value from unpaid labor. Postigo notes that Valve software actually requires modders to pay royalties for the development engine in order to produce modifications. Furthermore, in many massive multiplayer
online games, users create value by building the virtual environment through their creative efforts, but have to pay for the opportunity to add value. Taken together, it would seem that despite the fact users are voluntarily giving their time to the improvement of commercial games through modification, the developers are often simply exploiting free labor while giving little back to those who aid them.

Looking collectively at the article by Arakji & Lang and the article by Postigo, some valuable insights can be drawn. First, not all firms can be seen as equally exploitive. For example, while Id Software did indeed benefit from modifications to Doom and Quake, they also gave back to the community by releasing their source code under the GNU GPL. Even in the case of Id Software, however, the motivation for opening the source could be seen as a risk mitigation strategy rather than true altruism. Valve, on the other hand, continued to control the rights to the Half-Life source code and did not make the entire code visible to the public. The fact that Id's source code was opened up has had significant implications for the FOSS community, as will be seen later when the FOSS game Nexuiz (which uses the DarkPlaces engine) is discussed. Perhaps more significantly, the large number of mods available and the large number of skilled modders demonstrates there is a significant amount of game development talent in society, and many people are willing to contribute this talent even if they will not profit monetarily. If these modders were contributing their talent to enhance FOSS games, perhaps FOSS games would possess a more dominant position in the video game landscape. It seems likely that a major reason why these modders participate more in enhancing commercial games than participating in the development of FOSS games is simply because commercial games have larger user bases (if we assume a certain percentage of users have the technical ability and desire to modify games, a larger user base will translate into a larger group of potential modders). As will be seen later, however, many of the most popular FOSS games also have large user bases which actively participate in development. A normative statement can be made at this point. Commercial developers are harvesting the fruits of unpaid labor without fully returning the benefits to society at large. Thus, from
a labor standpoint, modification of FOSS games has a clear advantage. The fruits of the labor of modders, rather than being privately appropriated by a for-profit firm, essentially become public property. Modders of FOSS games do not have to watch their voluntary contribution be converted into a commodity by a for-profit firm.

**Literature Relating to FOSS Games:**

Thus far this chapter has discussed the topography and economics of the video game industry. The significance of user modifications to the industry has also been discussed. A discussion of the literature regarding FOSS games is now warranted. However, very little has been written about FOSS games, making the topic of this thesis fairly novel. In fact, in search for scholarly literature on the topic of FOSS games, only one highly relevant article was found. In addition to this article, however, Tim Ansell (2008) gave a talk at the Google campus, which was put online in the “Google Tech Talks” archive, about FOSS games. Both these works will be reviewed.

Scacchi (2004) attempts to evaluate how development practices in FOSS communities differ from the software engineering practices in commercial software development. This article is of special interest because Scacchi focuses specifically on practices in FOSS game development. He notes that the practices in FOSS game development are present in most FOSS development; but uses game development as a context for evaluating these practices. This research was conducted by in-depth study of four distinct FOSS game development communities. As FOSS development is often globally dispersed and decentralized, FOSS projects require different development practices than proprietary software. Scacchi identifies five areas in which FOSS development practices can be said to be unique.

First, rather than having a formal process for defining the specifications and requirements for a piece of software, these decisions are implicitly made through discussion and negotiation on development forums; thus no one person or entity has complete control over these decisions. Second, FOSS development communities have developed a unique method for managing release versions and
incremental updates of software. Commonly, FOSS developers make use of the Concurrent Version System (CVS), a FOSS system for organizing current source code for daily builds. This system allows individual developers to contribute modified modules, and allows either a project administrator or the community as a whole to decide which of these modules will be included in various builds; thus the best suited modules are maintained and the less well suited ones are replaced. Third is the issue of software maintenance and evolution. In proprietary software development a piece of software is said to be past its useful life once the code is no longer maintained or supported. In FOSS projects, so long as there is a sufficient amount of interest, a piece of software can be maintained indefinitely. Scacchi argues that in FOSS development this maintenance process is fundamentally different from the maintenance process in proprietary software development as, over time, it allows the software to be essentially reinvented into something which was not envisioned in earlier versions. Because of this characteristic of FOSS development, Scacchi notes that the life cycle of proprietary software can be dramatically extended by re-licensing the source code under a FOSS license (the DarkPlaces engine mentioned above is a good example of this). Fourth, FOSS projects tend to be managed through what Scacchi refers to as an “interlinked layered meritocracy”. Through seniority, competency, and leadership, developers can become leaders in the development of various components of a project. At the top, individuals who have taken on leadership positions share decision making responsibilities and ultimately decide the direction of the project. Finally, what Scacchi refers to as “technology transfer” is significant to FOSS development. Various FOSS communities make use of tools developed by other communities (and may also go back to contribute to the tools they use). This has both the effect of enhancing the tools available for FOSS development, and the effect of building an interconnected FOSS community with coherent social norms and values.

Tim Ansell (2008) gave a talk entitled “Gaming for freedom” relating to FOSS games on the Google campus early in 2008. Much of his talk was simply advocating the development and usage of
FOSS games rather than presenting factual information, but he made significant points as to why FOSS game development is important. One reason given was the closed nature of video game consoles. His assertion is that Microsoft would like nothing more than turn Windows into a console-like platform for which developers must pay licensing fees simply for the privilege of having their programs run. For this reason, Ansell argues quality FOSS games could challenge the console business model prevalent in the video game industry; thus preventing future lock-downs. Ansell also argues that the reputation of the Linux operating system as being behind Windows as a viable gaming platform is hindering uptake of Linux as a mainstream desktop operating system. He goes on to give examples (including Battle for Wesnoth, which is one of the games evaluated in this study) of successful FOSS games (although he does not analyze these games in the same depth as this study does), and encourages people who care about FOSS in general to get involved in game development.

Sales Statistics:

One of the most frustrating elements of video game research is the lack of information available about sales. NPD Group (a market research firm) gathers data about the videogame industry, however the majority of the data they gather is proprietary in nature and not available to the general public (nor to the academic community, even through university libraries). A small subset of this data, however, is made freely available to various publications. It is this freely available information which videogame magazines use in their monthly “charts” of best selling games. Even the Entertainment Software Association depends on this data, and cites this freely available data in their annual “Essential Facts” reports. As discussed above, however, it should be noted that information on sales of PC games may be significantly flawed.

NPD Group reports numbers of units sold for console (including handheld systems) on a monthly basis, but only releases number of units sold for PC games on an annual basis. NPD Group also does not release sales figures for specialized PC gaming hardware. This section thus presents sales
information from 2007 (number of console systems sold, top ten console games of 2007, and top ten PC games of 2007). In addition, the top ten most downloaded games from SourceForge are also listed (number of downloads from project pages as of July 4\textsuperscript{th}, 2008). It must be noted, however, that the SourceForge number may not be truly representative of the popularity of the FOSS games listed (see discussion of this topic in Chapter 4). The sales data for all charts (except for FOSS games) use NPD Group’s freely available data, while the prices listed in the price column are the listed prices on Amazon.com as of July 10, 2008 (shipping not included). Also, the data for the first three charts is limited to the U.S. market, while the SourceForge data is global.

\textit{Table 1 Consoles (Including Handhelds) Sales Data for 2007}

<table>
<thead>
<tr>
<th>Rank</th>
<th>System</th>
<th>Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Nintendo DS</td>
<td>8.5M</td>
</tr>
<tr>
<td>2</td>
<td>Nintendo Wii</td>
<td>6.29M</td>
</tr>
<tr>
<td>3</td>
<td>Xbox 360</td>
<td>4.62M</td>
</tr>
<tr>
<td>4</td>
<td>PlayStation 2</td>
<td>3.97M</td>
</tr>
<tr>
<td>5</td>
<td>PlayStation Portable</td>
<td>3.82M</td>
</tr>
<tr>
<td>6</td>
<td>PlayStation 3</td>
<td>2.56M</td>
</tr>
</tbody>
</table>

NPD data retrieved from \url{http://metue.com/01-17-2008/npd-december-2007-annual-video-game-sales/}

\textit{Table 2 Top Ten Console (Including Handhelds) Games for 2007}

<table>
<thead>
<tr>
<th>Rank</th>
<th>Title</th>
<th>Platform</th>
<th>Sales</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Halo 3</td>
<td>Xbox 360</td>
<td>4.82M</td>
<td>$59.99</td>
</tr>
<tr>
<td>2</td>
<td>Wii Play with Wii Remote</td>
<td>Nintendo Wii</td>
<td>4.12M</td>
<td>$49.99</td>
</tr>
<tr>
<td>3</td>
<td>Call of Duty 4: Modern Warfare</td>
<td>Xbox 360</td>
<td>3.04M</td>
<td>$59.99 (Game of the Year edition)</td>
</tr>
<tr>
<td>4</td>
<td>Guitar Hero III: Legends of Rock</td>
<td>PlayStation 2</td>
<td>2.72M</td>
<td>$49.99</td>
</tr>
<tr>
<td>5</td>
<td>Super Mario Galaxy</td>
<td>Nintendo Wii</td>
<td>2.52M</td>
<td>$49.99</td>
</tr>
<tr>
<td>6</td>
<td>Pokemon Diamond</td>
<td>Nintendo DS</td>
<td>2.48M</td>
<td>$33.99</td>
</tr>
<tr>
<td>7</td>
<td>Madden NFL 08</td>
<td>PlayStation 2</td>
<td>1.9M</td>
<td>$11.92</td>
</tr>
<tr>
<td>8</td>
<td>Guitar Hero II</td>
<td>PlayStation 2</td>
<td>1.89M</td>
<td>$34.99</td>
</tr>
<tr>
<td>9</td>
<td>Assassin’s Creed</td>
<td>Xbox 360</td>
<td>1.87M</td>
<td>$23.99</td>
</tr>
<tr>
<td>10</td>
<td>Mario Party 8</td>
<td>Nintendo Wii</td>
<td>1.82M</td>
<td>$49.99</td>
</tr>
</tbody>
</table>

NPD data retrieved from \url{http://metue.com/01-17-2008/npd-december-2007-annual-video-game-sales/}
Prices from Amazon.com
Table 3 Top Ten PC Games of 2007

<table>
<thead>
<tr>
<th>Rank</th>
<th>Title</th>
<th>Sales</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>World of Warcraft: Burning Crusade</td>
<td>2,250,000</td>
<td>$29.99</td>
</tr>
<tr>
<td>2</td>
<td>World of Warcraft</td>
<td>914,000</td>
<td>$19.99</td>
</tr>
<tr>
<td>3</td>
<td>The Sims 2: Seasonal Expansion Pack</td>
<td>433,000</td>
<td>$29.99</td>
</tr>
<tr>
<td>4</td>
<td>Call of Duty 4: Modern Warfare</td>
<td>383,000</td>
<td>$54.99</td>
</tr>
<tr>
<td>5</td>
<td>Command &amp; Conquer 3: Tiberium Wars</td>
<td>343,000</td>
<td>$28.99</td>
</tr>
<tr>
<td>6</td>
<td>Sim City 4 Delux</td>
<td>284,000</td>
<td>$14.99</td>
</tr>
<tr>
<td>7</td>
<td>The Sims 2</td>
<td>281,000</td>
<td>$15.00</td>
</tr>
<tr>
<td>8</td>
<td>The Sims 2 Bon Voyage Expansion Pack</td>
<td>271,000</td>
<td>$28.99</td>
</tr>
<tr>
<td>9</td>
<td>Age of Empires III</td>
<td>259,000</td>
<td>$29.99</td>
</tr>
<tr>
<td>10</td>
<td>The Sims 2 Pets Expansion Pack</td>
<td>236,000</td>
<td>$26.99</td>
</tr>
</tbody>
</table>

NPD data retrieved from [http://www.shacknews.com/onearticle.x/50939](http://www.shacknews.com/onearticle.x/50939)
Price information from Amazon.com

Table 4 Top Ten FOSS Games on SourceForge

<table>
<thead>
<tr>
<th>Rank</th>
<th>Title</th>
<th>Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>StepMania</td>
<td>4,981,902</td>
</tr>
<tr>
<td>2</td>
<td>Frets on Fire</td>
<td>4,897,639</td>
</tr>
<tr>
<td>3</td>
<td>SpeedSim</td>
<td>4,682,791</td>
</tr>
<tr>
<td>4</td>
<td>Battle for Wesnoth</td>
<td>2,456,926</td>
</tr>
<tr>
<td>5</td>
<td>Tux Racer</td>
<td>2,404,024</td>
</tr>
<tr>
<td>6</td>
<td>BZFlag</td>
<td>1,816,950</td>
</tr>
<tr>
<td>7</td>
<td>Cube game &amp; 3D engine</td>
<td>1,700,676</td>
</tr>
<tr>
<td>8</td>
<td>Nexuiz</td>
<td>1,650,558</td>
</tr>
<tr>
<td>9</td>
<td>Sauerbraten game &amp; engine</td>
<td>1,500,436</td>
</tr>
<tr>
<td>10</td>
<td>DooM Legacy (multiplatform port of the old commercial game Doom)</td>
<td>1,433,504</td>
</tr>
</tbody>
</table>

Conclusion:

In this chapter, the focus has shifted from a general discussion of FOSS to a discussion of video games and the videogame industry. However, through analysis of the video game industry, the importance of user modifications has also been explored. The discussion of user modifications sheds light both on the practices of the industry and provides insight into why people participate in unpaid game development work; thus this discussion is extremely valuable for the purpose of this thesis.
Finally, this chapter noted that not much research has been conducted specifically on FOSS games; but nonetheless a study of the development practices of FOSS game communities along with a summary of a Google Tech Talk about FOSS games are presented. The discussions of this chapter and previous chapters have provided a background on topics relevant to this thesis. Having provided this background, it is now time to present the methodology used for evaluating FOSS games.

**Restatement of Research Questions:**

Up to this point we have explored American copyright law, ideological perspectives of the copyleft movement, the economics of FOSS development, and the economics of the video game industry. These topics provide a background for the research of this thesis. Thus, before we discuss the methodology and results, it is useful to review the research questions we will be addressing in the coming chapters.

As we have seen, a large body of high quality open source software exists. However, in many cases economic incentives exist for FOSS development. In general, FOSS games seem to lack many of the economic incentives discussed in Chapter 2. Thus the first research question is: “**How do open source video games compare to their commercial counterparts?**” Naturally, if a difference in quality is observed between commercial games and FOSS games, we are lead to the second research question: “**How can any disparities in quality be explained?**” Finally, we should remember why these questions are being asked. Copyrights are meant to provide an incentive to promote the creation of new content. But if high quality content can be generated without the exclusive rights granted by copyrights, what is the utility of copyrights? Thus, the final research question can be stated as: “**What implications, if any, do these findings have for the utility of copyright law in regard to video games?**”
Chapter 4: Methodology

A central question in this thesis is how FOSS video games compare to commercial video games. This comparison, however, will largely depend on how quality in video games is defined. At the most basic level, it is fair to say a game is high quality if players experience it as “fun”. For the purpose of this study, however, this definition of quality is insufficient. While a subjective examination of the entertainment value of the games studied will be a component of this research, some more objective criteria are necessary. Furthermore, video games have evolved tremendously over the decades. While games such as Pong, Invaders, and Pac-Man may have provided many hours of entertainment to players in their time; players have come to expect more complicated and graphically sophisticated game play. This is not to say these classic games are no longer enjoyable; and game vendors continue to capitalize on these titles (as is seen with the “Virtual Console” for the Nintendo Wii and the repackaging of retro games for other platforms). Modern games have, however, undeniably become much more sophisticated as is graphically (no pun intended) demonstrated by recent titles such as Crysis (2007). For this reason, the enjoyability of games can be seen as an evolving standard; and the enjoyability of FOSS games needs to be evaluated in comparison to contemporary commercial games.

In evaluating the open source games, four distinct methods of evaluation will be used. Each of these four methods will be discussed in more detail below. First, a modified version of the formal framework presented by Mark Wolf (2001) will be used. Second, the criteria used by contemporary game reviews (such as those used in game magazines like Game Informer) will be applied. Third, user ratings and download statistics will be used as a proxy measurement for popularity. Finally, my own subjective judgment will be applied. After providing a more detailed description of each of these criteria, a coding scheme will be presented.

In his 2001 book “The Medium of the Video Game”, Mark Wolf applies the methodology of formal analysis from film and television studies to video games. He divides his analysis into four
broad categories: Space, time, narrative, and genre. While each of these concepts will be useful for this study, they will, to differing degrees, have to be refined for this thesis.

Wolf divides video game space into two broad categories: on-screen space and off-screen space. He notes that off-screen space can be acknowledged and used in a variety of ways. He goes on to describe eleven distinct ways space is used in video games then describes different methods of rendering three dimensional space on a two dimensional screen.

The first spatial arrangement described is all text representations of the game world. In such games, scenes are described by text rather than being represented graphically. Wolf argues that in such games all space could be considered “off-screen space”; on the other hand, Wolf argues that the scene currently being described could be considered the on-screen space while the larger game world could be considered the off screen space. Other spatial arrangements described by Wolf include games in which the entire world is contained on a single screen (in such games off-screen space can be acknowledged to varying degrees); games which allow scrolling around a world on either one or two axes; games containing multiple connected but non-scrolling screens; games with multiple scrolling planes (which can create an illusion of depth); games which allow some movement in a third dimension while falling short of being fully interactive three dimensional worlds; games which display multiple nonadjacent viewpoints on a single screen; games which allow a player to move around freely in a three dimensional world; and finally games which make use of schematic representational maps to help the player navigate through the world.

Wolf describes the various techniques used to create an illusion of three dimensional space on a two dimensional screen. To a large extent, the methods used have correlated to available computational technologies. Earlier games often made use of simple scaling of size (distant objects appear smaller than close objects) or pre-drawn backgrounds to create an illusion of depth. Newer games, however, have the ability to render three-dimensional worlds from three dimensional models in the computer.
memory in real time.

Since the time of Wolf's research, computational technologies and representational techniques have continued to advance. One example of such change is the developments which have occurred in strategy games. A standard format in modern day strategy games (both turn based games such as Civilization IV (2005) and real time games such as Rise of Nations: Rise of Legends (2006)) are what Wolf describes as scrolling on two axes, but with the world to be rendered in three dimensions. Furthermore, such games frequently make use of “mini-maps” which combine Wolf's representational map concept with his concept of multiple nonadjacent viewpoints on a single screen. Furthermore, in both Civilization IV and Rise of Legends (along with many other strategy games), the minimaps are used actively for navigation within the game world. Furthermore, while such games typically use scrolling on two axes, they now also typically allow the player to zoom in and out (in order to either focus on a smaller portion of the game world or view a larger field of play). An extreme example of this can be seen in Supreme Commander (2007) in which the player is allowed to scroll all the way out and view an extremely large battlefield with units, buildings, resources, and terrain depicted in a highly schematic and representational form; or to zoom in quite close to see detailed three dimensional animation of action and terrain.
While Wolf's description of the usage of space in video games is useful for analyzing the development of games over time and describing emerging conventions, in order to use it as a criteria for evaluating quality of open source games, it needs to be modified; as, while these descriptions are useful for describing game play, their usefulness as defined by Wolf for evaluating quality is limited. For the purpose of this research, the evaluation will focus on how effectively games make use of space with respect to their genre. More specifically, this study will look at both the use of space in terms of compliance with contemporary standards in the respective genres, and innovative use of space which enhances the experience of the specific game. Furthermore, graphical representations have grown significantly more sophisticated since the time of Wolf's research. Some of these advanced rendering technologies (such as dynamic lighting) go beyond Wolf's description of use of space; and will be discussed later.

Wolf takes a similar approach to describing the use of time in games as to his approach to space. As noted above, when dealing with space there is a distinction between on-screen and off-screen space.
A parallel distinction in terms of time is “real time” versus “game time”; where real time is time passing in the real world while game time is the internal management of time in the game world. Wolf notes that, just as in film, the usage of time often plays a significant role in the mechanics of games. Unlike films, however, many games can offer tens of hours of game play; allowing time to be less compressed than in films. Wolf notes that there are various techniques which can be used by game developers to portray the passage of time. One method is simply to contrast stillness and motion. True stillness and lack of time passage can occur when a game is paused; but passage of time can be communicated during periods of relative inactivity through the use of ambient sound and movement. Another component of time in video games is the use of “cyclical” or “looped” time in which movement and/or sound is repeated. An example of this type of usage of time can be seen in the game which is described in the appendix, in that within each level the enemies repeatedly move back and forth on a platform until they are killed and the music for a level continues to loop until the level is completed. The game described in the appendix also provides an example of a combination of stillness and looped time; as the title screen remains static until a key is pressed, but the music continues to loop. Another example of looped time can be seen in many games when a player loads a game to return to a previous point of game play. Passage of time can also be presented through the usage of interludes (such as video cut scenes between levels) which simultaneously allow the player to take a short break from game play, develop the story, and represent the passage of time. The impact passage of time has on game play can take different forms. In some action games, timing of movements is essential for success. In some simulation games, such as the Sim City games and Civilization games, passage of time represents progress through history.

Not discussed in detail by Wolf was the usage of time in real time strategy (RTS) games. Just as in the above discussion of space, Supreme Commander (2007) provides an excellent example of innovative usage of time. As described above, the ability to zoom in and out on scrolling two
dimensional (rendered in three dimensions) maps is common in modern RTS games, but Supreme Commander takes it to the extreme by allowing the player to zoom all the way out or zoom in very close. Likewise, the ability to adjust the rate of time passage in RTS games has become a common feature, but Supreme Commander takes it a step farther by giving the player an extreme degree of control over the passage of time. Time can be slowed nearly to a halt or speeded up to allow what would take hours at default speed to occur within minutes. This control greatly enhances the game play experience. It is useful, for example, to slow time down nearly to a halt in order to add buildings, military units, and terrain enhancements to production queues, then speed the game up to allow resources to accumulate and production to be completed, then either slowed down or speeded up during battles depending on the degree of micromanagement sought by the player.

As with the usage of space, the usage of time cannot directly be said to correlate with the quality of a game. Therefore, for the purpose of my evaluation, the usage of time will be evaluated in terms of whether it complies with conventions of the respective genres of games and whether it is used in an innovative manner which improves the experience of game play. As with my evaluation of the usage of space, my evaluation of time will require some subjective judgment. Time, however, will play a significant role in my evaluation of a number of games.

The next element of formal analysis of video games discussed by Wolf is narrative. A large portion of Wolf's discussion of narrative has to do with the historical development of narrative in the medium. He explains how early hardware used for games was unable to render detailed characters, which prevented significant anthropomorphism of on screen characters. As representations became more realistic, narrative began to emerge as more important to games. Narrative can play dramatically different roles in different games. In highly abstract games, such as Tetris, narrative may not be required. In other games, narrative is only present as a background story meant to give structure to the game world. This is the case in the game described in the appendix; in this game, the background story
simply served to explain the game world and provide context and meaning to the game world and game play. In other games, explains Wolf, narrative can develop during game play. Stories can be pushed forward through the use of video cut scenes or information presented in game. One key concept presented by Wolf is the idea of interactivity in narrative; or to what extent the player can control the direction the story takes. In some games, the player has little or no control over the story, and simply advances through the pre-defined story as she progresses further through the game. In various strategy games, on the other hand, the player has control of the entire history of a society. In various offline role playing games such as The Elder Scrolls IV: Oblivion (2005), while there are many predefined quests a player can choose to undertake, the player has a large amount of freedom to choose affiliations and development of her character. Skills, character parameters, inventory items, owned property, reputation, and wealth are all influenced by the decisions of the player. In online role playing games, players are often given the ability to form relationships with other players and participate in the development and dynamics of the politics and economics of the game world.

Narrative is important to a varying extent in different games. In some games, it is more necessary than in others. Therefore, for the purpose of my research, narrative will be acknowledged when it is present, and it will also be noted if it is conspicuously lacking where it should be present. Innovative usage of narrative will also be noted.

Finally, Wolf addresses the concept of genre in video games. His conception of genres is based on the tasks which the player is required to perform to be successful (such as racing, shooting, chasing) and broad layout of the game (such as board games, puzzle, etc.). He describes forty-two genres, but argues individual games can fall into multiple genres. This typology, however, is not commonly used by gamers today. Rather, in gaming magazines and websites, games are generally defined in a narrower set of terms. Some of these will be discussed when I present methodologies used in gaming magazines. For the purpose of this study, I will use the narrower set of genres, and for games which
don't fit neatly into any of the major categories, I will simply define the genre as “other” and describe the gameplay.

Wolf's framework for formal analysis is useful in that it provides a means to analyze various components of games. As explained above, however, by itself it is insufficient for analyzing the overall “quality” of games. What this framework allows, however, is the evaluation of innovative and practical usage of space, time, and narrative in FOSS games. This will be a significant component of my evaluation, but it alone is insufficient.

The second source used for the development of a methodology comes from Game Informer Magazine (January 2008). In sharp contrast to the formal approach presented by Wolf, which is highly academic; the reviews in Game Informer are meant to provide advice to consumers as to which video games are worth buying and which are likely to be a waste of time and money. Such an approach, despite potential shortcomings in terms of validity (especially considering that the coding will be done by me alone, and subjective judgment is required), is useful as it comes closer than Wolf's formal approach to being able to analyze the quality of a game.

On page 84 of Game Informer magazine issue 177 (January 2008), a description of the rating system is given:

Our crack (or crackhead, we can never decide which) review team rates games in a number of categories to help you sort out the great from the stuff you'll hate. Most games are reviewed by two staff members, and you will find both their opinions on each review. To make things a little easier, we have put together some definitions of what the numbers mean, what we look for in a game, and also a cheat sheet so newbies can understand our advanced video game jargon. It is important to note that the Game of the Month is determined only by the main review score, not an average of the two opinions.

THE SCORING SYSTEM
10 Outstanding, A truly elite title that is nearly perfect in every way. This score is given out rarely and indicates a game that cannot be missed.
9 Superb. Just shy of gaming nirvana, this score is a high recommendation because the game reviewed is head-and-shoulders above its competition
8 Very Good. Innovative, but perhaps not the right choice for everyone. This score indicates that there are many good things to be had, but arguably so.
7 Average. The game's features may work, but are nothing that even casual players
haven't seen before. A decent game from beginning to end.
6 Limited Appeal. Although there may be fans of games receiving this score, many will be left yearning for a more rewarding game experience.
5 Passable. It may be obvious that the game has lots of potential, but its most engaging features could be undeniably flawed or not integrated into the experience.
4 Bad. While some things may work as planned, the majority of this title either malfunctions to varying degrees or is so dull that the game falls short as a whole.
3 Painful. If there is anything that's redeeming in a game of this caliber, it's buried beneath agonizing gameplay and uneven execution in its features or theme.
2 Broken. Basically unplayable. This game is so insufficient in execution that any value would be derived in extremely small quantities, if at all.
1 Failboat. All aboard!
> Concept: What new ideas the game brings to the table and how well old ideas are presented.
> Graphics: How good a game looks, taking into account any flaws such as bad collision or pop-up.
> Sound: Does the game's music and sound effects get you involved or do they make you resolve to always play with the volume down?
> Playability: Basically, the controller to human interface. The less you think about the hunk of plastic in your hands, the better the playability.
> Entertainment: Flat out, just how fun the game is to play. The most important factor in rating a game.
> Replay Value
  High – You'll still be popping this game in five years from now.
  Moderately High – Good for a long while, but the thrills won't last forever.
  Moderate – Good for a few months or a few times through.
  Moderately Low – After finishing it, there's not much reason to give it a second go.
  Low – You'll quit playing before you complete the game.

On the following page, definitions for jargon used in the magazine are presented. Among this jargon, several genres are named. These include Action, Adventure, Board, Fighting, First Person Shooter (FPS), Massively Multiplayer Online (MMO, or MMORPG), Platform, Puzzle, Racing, Role Playing Games (RPG), Shooter, Sports, and Strategy (which can encompass both turn based and real time strategy).

Part of the coding scheme will use a similar approach. However, for the purpose of this research, some modifications and clarifications are necessary. First, rather than using a numerical scoring system for the overall assessment of the games which are evaluated, this study will rate them as “Very Good, Good, Fair, Poor, or Very Poor”. The reason for this is that it creates broader (and self
explanatory) categories, thus arguably improving the validity of the assessment. Concept, Graphics, Sound, Playability, Entertainment, and Replay Value will all be assessed; but in addition to these categories this study will also assess complexity, “Buggyness”, user interface, installation, modifiability (along with presence or absence of modifications), and performance. Both the original categories taken from Game Informer and these additional categories are explained below.

This study will not necessarily use “concept” as a criteria for overall quality. It will, however, be used to evaluate whether the game as a whole is either exceptionally innovative or is subjectively “exceptionally lame in concept.” In general, the “concept of concept” (no pun intended) will be used as a general description of the game (going beyond genre, which will be discussed later). It will be noted if a game is a remake of a commercial game (as is the case with StepMania, which is a FOSS clone of Dance Dance Revolution). It will also be noted if the game is based on existing media (as is the case of Armagetron Advanced, which is based on the light racing game in the film Tron). Concept, then, will provide a general description of the game. While concept goes beyond genre, for simplicity genre will be included in the description of concept. Furthermore, for simplicity, the description of concept will also include the formal elements of space, time, and narrative.

As explained earlier, graphics is an evolving standard. Games which would have been considered cutting edge graphically in the late 1990's would now be considered graphically mediocre. Furthermore, defining high quality graphics, even applying contemporary standards, is no trivial task. Games with relatively simple graphics can still be aesthetically pleasing; and different genres of games make use of graphics in different ways. One simple component of graphics which can be objectively judged is whether the game is played in full-screen mode or is windowed, or played in either of the above configurations. Another easily judged graphical component is available resolutions (for example 320x200, 640x480, 800x600, 1024x768, 1600x1200) and the default aspect ratio of the game. However, it is unlikely that there will be much variance in these regards; thus there is little point in
evaluating these components unless they are conspicuously lacking in situations in which they should be present. Beyond these easy to detect components of graphics, there are several elements of graphics which would be appropriate to evaluate in some games, but not in others. Among these are texture detail, geometric detail, lighting detail, presence or absence of anti-aliasing and anisotropic filtering, and the degree to which graphics detail (in all the above categories) can be configured. In addition to these elements, a subjective judgment of how aesthetically pleasing the graphics are will be made. When appropriate, graphical comparisons will be made with commercial games (for example, it may be noted if a game is graphically equivalent to an older game such as Quake III). Also, when appropriate, and when the information is available, it will be noted which graphics engine was used (and whether the engine was originally developed commercially then made open source, or whether it was created open source from the bottom up). Because of the complexity involved in analyzing graphics, no single rating of “good” or “poor” will be of use across the spectrum of games reviewed. Rather, the graphics will need to be discussed on a case by case basis. Thus, while meaningful observations can be made about graphics, for the purpose of this research these observations are not collapsible to a single overall rating.

Evaluation of sound will be simpler. It will be judged in terms of whether it is overall aesthetically pleasing, and whether it adds to the overall experience of the game. If either sound or music are conspicuously lacking, this will also be noted. When appropriate, sound effects and music will be evaluated separately. Also, when appropriate, sound and music will be judged on its technical merits (although with less precision than graphics). An example of a technical consideration may be whether the music uses a format similar to MIDI, or whether it uses a more advanced format such as .MP3.

Playability will focus on the controls used during game play. This will focus on two considerations: overall ease of controlling applicable aspects of game play, and configurability of the
controls (whether, for example, the way the keyboard and mouse are used can be adjusted).

Entertainment will be a subjective judgment. It will consist both of my opinion of how “fun”
the game is, and an explanation as to why the game is described as fun or not fun.

“Replay Value” will also be fairly subjective. The replay value will be described as “High”,
“Medium”, or “Low”. A game with high replay value will be one which I am likely to continue playing
for some time. Medium will indicate that the game will hold my attention and keep me coming back
for at least a short period of time. Low will indicate that the game is boring, and after playing around
with it for a little while, I have no desire to continue playing, and am unlikely to ever play it again. For
each game, an explanation will be given for the judgment of replay value.

The overall complexity of games will also be assessed. It should be noted, however, that a high
degree of complexity is not always necessary for a game to be considered high quality. Thus, when
evaluating complexity it will be noted whether the game is sufficiently complex considering the genre
and style of the game. Complexity, like graphics, isn't a single dimensional consideration. There is
furthermore no single method for evaluating complexity which can be applied to all games. One
method for evaluating complexity will simply be a “prima facia” evaluation. A game in which a player
has to navigate through a three-dimensional world and perform a wide variety of tasks is on its face
more complicated than a game in which a player has to bounce a ball around a single screen or perform
some other simple task on a single screen. This, of course, by itself is not a sufficient measurement, but
it can be applied universally across all games. Other ways of measuring complexity which may be
applicable to some games but not to others include the number of items a player can interact with, the
degree to which game play can be configured and modified by the user, and the number actions in
which a player can engage. As these criteria cannot be applied universally to all games, however, the
complexity of each game will be discussed independently. In order to add perspective, however, where
possible comparisons will be drawn with commercial games. In addition to these measurements, the
size of the primary download for the game will be used as a proxy for complexity. While commercial
games can often take up an entire DVD (more than 4 GB) or even multiple DVDs, all the games in the
sample used for this study are significantly smaller. Although this proxy for complexity can provide
some insight, certain limitations should also be noted. First, many games rely heavily on user
modifications and add-on components. If these modifications are not included in the primary
download, they will not impact the file size. Also, a large download could simply be indicative of poor
compression or of large component files. For example, the game discussed in the appendix uses .mp3
files for music. Without these music files, the game would be much smaller. Thus in many cases
download size will not correlate closely with the complexity of game play.

Another important aspect to be considered is whether there are noticeable “bugs” in the games.
These can include graphics glitches, design faults in levels (such as allowing a playing to become stuck
so he can neither die nor escape nor complete the level), and instability or crashing. Bugs are not
unique to FOSS games, but their presence can have a significant impact on the overall enjoyment a user
can get out of a game. S.T.A.L.K.E.R. (2007), a commercial first person shooter with role playing
elements, is an example of a game which had great potential, but in my view was ultimately a failure
because of bugs. I gave up playing it after, on more than one system (each with a different hardware
configuration and operating system, one with Windows XP, one with Windows Vista), the game would
-crash every few minutes, rendering it nearly unplayable. As testing games on multiple systems with
different hardware and software configurations is beyond the scope of this thesis, the assessment of
bugs will necessarily be limited. It will be noted whether bugs interfere with gameplay or the overall
enjoyability of the game. Furthermore, if a game simply hangs or crashes, this will be considered less
serious than if it hangs the whole operating system and forces me to reset the computer. If for some
reason a game simply will not run on my system, this will also be noted. The most serious type of bug
anticipated (which hopefully will not be encountered) is if a game damages my system, and, for
example, forces me to reinstall device drivers or my operating system.

User interface, as used in this study, generally refers to the menus within a game (which for the most part will be outside of actual gameplay). The primary considerations will be whether the menus are easy to navigate and use, and whether they provide me with sufficient flexibility to configure aspects of the game which the user should be allowed to configure in order to enhance her playing experience.

Another component which will be noted is the installation process. This is a relatively simple consideration. The question is whether the installation process involves running a setup program (as is typical for commercial video games), or whether it involves decompressing a file to a destination folder. The reason this is note-worthy is because the use of a setup program makes the process seem more polished and professional. Furthermore, the absence of a setup program may intimidate less experienced computer users.

For some of the games, user modifications play a significant role in the overall value of a game. StepMania, for example, is in and of itself a relatively simple program. It is packaged with no playable content, and depends entirely on “step patterns” created by users. There are, however, a wide variety of these step patterns (created by a wide variety of people) available for download. This allows for a rich gameplay experience. Thus, whether a game is modifiable (apart from actually being able to “tinker” with the source code, which is by definition a prerequisite for all FOSS programs), whether such modifications are available, and whether they add to the overall value of the game.

Finally, performance will be evaluated. Essentially, the question is whether the game runs well on the system it is being tested on (the specifications of this system will be discussed later in this chapter). Does the frame rate fall noticeably when resolution and details are set to their maximum settings? If so, how does this compare to commercial games of similar complexity? Crysis (2007) runs extremely poorly on my system when resolution is increased to 1600x1200 and all details levels are set
to maximum; but it runs reasonably well at a resolution of 1024x768 with near maximum detail settings. This, however, is acceptable because of the extreme complexity of graphics and game play. If a relatively simple FOSS game performs poorly on my system, this will be noted.

Genre (which will be included under the “concept” description) will place games in one of several categories. Genre, as used in this study, can be defined within the following categories (this list is not comprehensive, but there are no games in the sample which do not fall into at least one of these categories): RTS, Turn Based Strategy, FPS, Racing, Action, MMO, and Other. Broadly speaking, RTS games can be defined as games in which the player must control elements of a society at war in real time (for the most part, game time does not stop while the player is making decisions). Elements being controlled in such games often include technological research, gathering or generating economic resources, building of military units and infrastructure, and combating enemies. Turn based strategy games are, for the most part, similar to RTS games, but the gametime generally does not pass while the player is making decisions (in FreeCiv, the turn based strategy game which will be evaluated, the player moves his/her units and allocates economic resources then the AI players do the same before the player can do anything else). FPS games are games in which the player moves around a three dimensional world, has a first person perspective, and fires projectiles at enemies (such games can be either single player or multiplayer). The term “racing” is for the most part self explanatory. It can involve either racing against a clock or other players. For the purpose of this study, “action games” will be defined as games in which the player must move around in a world and quickly make decisions, but do not fit within the category of FPS or racing. The category “other” will be used for games which do not fit neatly in any of the above categories.

The overall assessment of “Very Good”, “Good”, “Fair”, “Poor”, or “Very Poor” will take all the above concepts into consideration. In addition to this assessment, two other proxies will be presented. First, popularity, operationalized as number of downloads, will be used (this is used as one
criteria in selection of the sample of games evaluated in this study). Using number of downloads as a proxy for popularity, however, has a few limitations which should be noted. First, this number is taken from SourceForge (www.SourceForge.net), and does not include downloads from other sites. Second, this number includes total number of downloads from a project page on SourceForge, including source code downloads, binary downloads, add on downloads, and patch downloads. This is likely to somewhat skew the results. Finally, the download numbers listed on SourceForge are measured as the number of downloads since the project page was created; thus games which have been around longer are likely to have more downloads than projects which began more recently. Despite these limitations, however, the number of downloads can be used as an approximate measurement of popularity. In addition, in order to increase reliability, user ratings will also be used. As with the measurement of number of downloads, however, there are some limitations to the usefulness of this measurement. First, SourceForge does not provide information on user ratings; which means these ratings must be gathered from other web sites. Unfortunately, there does not appear to be any single source which has ratings for all the games in my sample, which means the ratings cannot all be collected from a single source. The website download.com, however, has user ratings for a majority of the games in my sample. Therefore, this site is used as the primary source. For games not rated on download.com, however, a Google search is used to find user ratings (the source where the ratings are found will be noted). Another limitation to using user ratings is that, while ratings for all games in my sample can be found, some of the ratings are for different versions (for example an older version or a version for the Macintosh or Linux platform). Finally, some of the games have been rated by more users than others; thus the average of a large number of ratings will be a more valid measurement than a rating by a single user. Thus the number of user ratings for each game will also be noted. As not all games in my sample can be found at download.com, providing download numbers from this site would not be useful.

Having given a broad overview of the methodology to be used, an actual code sheet will now be
presented. The sheet will be used as a crude tool for evaluating the games. Some aspects of the code sheet may only be applicable to certain games. After collecting the data using the code sheet, a description of each game in terms of data collected on the code sheet will be presented. In addition to the detailed descriptions of the games, some of the criteria on the code sheet will also be summarized in a table at the end of the results chapter.

**Code Sheet:**

- **Concept:**
  - Genre:
  - General Description:
  - Use of Space:
  - Use of Time:
  - Use of Narrative:
  - Subjective evaluation of these elements and other comments:
- **Graphics:** (Using criteria described above)
- **Sound:** (Using criteria described above)
- **Playability:** (Using criteria described above)
- **Entertainment:** (Using criteria described above)
- **Replay Value:** (Using criteria described above)
- **Complexity:** (Using criteria described above)
- **Buggyness:** (Using criteria described above)
- **User Interface:** (Using criteria described above)
- **Installation:** Setup Program Included or Not?
- **Modifiability:**
  - Dependent on modifiability?
  - User modifications available?
  - Do these modifications significantly increase the value of the game?
- **Performance:** (Using criteria described above)
- **Overall Assessment:**
  - Bottom line rating
  - Other notes
- **User Rating:** (From 0 to 5, average of all ratings from the source)
- **Number of downloads:** (As discussed above)

Above the criteria for evaluation of the games has been discussed. Now criteria for selecting a sample will be discussed as well as information on the data collection. The population from which the games is drawn is SourceForge. SourceForge is a website which describes itself as “the world's largest Open Source software development web site”, and was chosen simply because it provides the single
most comprehensive collection of FOSS projects while at the same providing download statistics and
descriptions of projects. In addition to being listed on SourceForge, several other criteria must be met
for a game to be included in my sample. First, in order to be included in my sample, the game must
have a single player mode. There are several reasons for this decision. First, it is far easier to explore
and evaluate a game without having to concurrently evaluate the actions of other players. Second,
evaluating multiplayer games would involve analysis of network performance which is beyond my
control or ability to influence. Finally, evaluation of multiplayer games would require finding servers
with other players, which may or may not exist. For this reason, while multiplayer modes may be
mentioned, my evaluation will be limited to single player modes. The second criteria for selection is
that the evaluated game must actually be a full game, and not a modification to an existing commercial
game. Third, the content of the games must have been originally developed for a FOSS project;
although games which use derivatives of commercial engines which have been made open source can
be included. Fourth, the games must have windows binaries so they can be played on the system used
for this study. Finally, the games must have been updated since January 1st 2007. The reason for this
final criteria is to allow the games to be fairly judged against contemporary standards. It would not, for
example, be fair to judge a game not updated since 2002 to the same graphical standards as a game
from 2008. The final sample consists of the eight most popular games on SourceForge (popularity
operationalized as number of downloads which is discussed above). Data on these games was collected
between February 15th 2008 and February 29th 2008; thus the data on these games was all collected
within a fairly narrow time period. The eight titles evaluated listed in Table 5.

Before the results of this study are presented, one final detail remains to be discussed. This
study will be limited to FOSS games available for the Microsoft Windows platform, as this is the
platform installed on my most powerful computer. The specifications of the machine being used to test
the games are as follows: AMD Athlon 64 X2 6400+ processor (duel core), 4 GB 800 MHz DDR2
RAM, NVIDIA 8800GTS 640 MB Video Card, SoundBlaster FX ExtremeMusic Sound Card, reasonably high capacity and high speed hard drives, Microsoft Windows Vista x64 Ultimate Edition. At the time of this research, these specifications can be seen as fairly high end, but not top of the line. A majority of currently cutting edge commercial games run reasonably well on this system even at fairly high resolution and detail settings. Note should be taken of the fact that the machine is running Vista x64, however, as this could potentially present compatibility issues (compared to Windows XP x86).

Table 5 Games in Sample

<table>
<thead>
<tr>
<th>Name</th>
<th>Date version was released</th>
<th>Version number</th>
<th>Number of downloads</th>
<th>License</th>
</tr>
</thead>
<tbody>
<tr>
<td>StepMania</td>
<td>January 2008</td>
<td>Intermediate release version 4.0</td>
<td>2,760,254</td>
<td>MIT License</td>
</tr>
<tr>
<td>Battle for Westnoth</td>
<td>February 2008</td>
<td>1.4 pre-release version 1.3.19</td>
<td>2,128,682</td>
<td>GNU GPL</td>
</tr>
<tr>
<td>Sauerbraten game &amp; engine</td>
<td>December 2007</td>
<td>n/a</td>
<td>1,323,240</td>
<td>Zlib/libpng License</td>
</tr>
<tr>
<td>Scorched 3D</td>
<td>January 2008</td>
<td>Build 41.3</td>
<td>1,292,524</td>
<td>GNU GPL</td>
</tr>
<tr>
<td>Nexuiz</td>
<td>May 2007*</td>
<td>2.3*</td>
<td>1,292,524</td>
<td>GNU GPL</td>
</tr>
<tr>
<td>Armagetron Advanced</td>
<td>February 2008</td>
<td>0.2.8 Alpha</td>
<td>1,000,408</td>
<td>GNU GPL</td>
</tr>
<tr>
<td>Freeciv</td>
<td>January 2008</td>
<td>2.1.3</td>
<td>901,470</td>
<td>GNU GPL</td>
</tr>
<tr>
<td>Secret Maryo Chronicles</td>
<td>December 2007</td>
<td>1.4</td>
<td>866,478</td>
<td>GNU GPL</td>
</tr>
</tbody>
</table>

*after data was formally collected, version 2.4 of this game was released which included major improvements; thus when this title is discussed, the improvements in version 1.4 will also be noted in passing
Chapter 5: Results

Up to this point, we have broadly discussed copyrights, the copyleft movement, FOSS software, and the video game industry. In the previous chapter a methodology for selecting and evaluating FOSS games was presented. The previous chapter ended with a list of games to be evaluated. This chapter presents the data and ends with a chart summarizing the findings.

StepMania:

The first game reviewed is entitled StepMania. The version evaluated was released in January 2008, was the version reviewed for this study. As only the executable and no actual content was included in the download, the download size was only 13.7 MB. However, a large amount of content for this game can be found on the Internet. There are no user ratings for this version of StepMania available on download.com, but eleven people posted ratings for version 3.9 on download.com. The average user rating is 4.5 out of 5.

This game is clearly a clone of the commercial Dance Dance Revolution (DDR) series of games. In this game, the player must synchronize her movements to the rhythm of music and the corresponding instructions on the screen, which take the form of arrows indicating which button on the keyboard or dance pad must be pressed. While the game can be played with a keyboard, DDR-style dance pads are also supported; although an adapter is generally required in order to use these dance pads on a PC as they are typically designed to plug into proprietary console connection ports.

As this game is a clone of the DDR games, the usage of space and time is virtually identical to the DDR games. During game play, arrows indicating the required movements scroll up from the bottom of the screen toward static grayed out arrows on the top of the screen. The player must press the corresponding button on the keyboard or dance pad precisely at the time the scrolling arrow reaches the grayed out arrow at the top of the screen in order for the step to be considered successful. The moving arrows, which represent required movements, could be said to exist off screen before they are
seen, but cease to exist after they pass the grayed out static arrows on the top of the screen. If a player is familiar with a particular song and its corresponding step pattern, she can anticipate the arrow before they are even displayed. Proper timing is absolutely essential in order to succeed at this game. The player has both visual and auditory cues to aid in proper timing. Visually, the player must watch the scrolling arrows and prepare to press the corresponding key on the keyboard or dance pad when the scrolling arrow reaches the static arrow on the top of the screen. In addition, the scrolling arrows reach the static arrows at times which correspond to the rhythm of the song being played; thus setting a pace to which the player can coordinate. Furthermore, in songs in which the music slows down or speeds up, the movement of the arrows on screen appropriately slows or speeds up.

The graphics in this game are more than adequately configurable considering the style of game play. The user has the options of adjusting resolution and whether the game is played in fullscreen or windowed mode. Animation within the menus is lively and adds greatly to the overall user experience. As the download for the software is not bundled with playable content, the user is required to install step patterns and songs developed by users. Therefore, the graphics within actual game play depends largely on the downloaded songs. Some of the song/step pattern files have animated backgrounds, while others have static background images. The fact that the software supports animated backgrounds, however, indicates that the game can be considered graphically sophisticated. Although the background graphics during game play depend on the designs implemented by users; the software is capable of aesthetically pleasing graphics.

The evaluation of sound for this game requires the sound effects and music to be evaluated separately. Included with the primary download are sound effects and music for the menu system. The sound effects seem to serve two primary functions: one is to provide audio feedback to user interaction with the menus, and the other is to give the menu system a more lively feel. Although simple, these sound effects achieve both these functions quite well. The music for the menu system has a techno-like
sound, and generally improves the experience of using the menu system. Within actual game play, sound effects are not present as they would likely interfere with the experience of the music. The music within game play relies entirely on the decisions of the song/step pattern designers; but the software supports high quality music.

The primary point to be made about playability is that control with either keyboard or dance pad is highly usable. Naturally, it is easier to do well at the game when controlling it with a keyboard, as using the keyboard only requires the player to use her fingers while the dance pad requires movement of the whole body. However, playing with the dance pad provides more challenge and arguably more entertainment. Furthermore, using the dance pad provides a physical workout; which, while tiring, is also highly enjoyable.

The question of the complexity of this game can be addressed in more than one way. While the size of the primary download is relatively small, and this game does not require as sophisticated of a game engine as many other genres of games, this software offers a high degree of configurability and flexibility. There are several game modes supported; including support of different types of dance pads, multiplayer, and various rule sets. The player can also configure such aspects as how strictly timing is judged. Thus this software can be said to be as complex as it needs to be.

During the evaluation, only one noticeable bug was observed. Within the menus, some text fell outside the boundaries of the screen, rendering it unreadable. It is possible that this would not be an issue in a wide-screen configuration; but this was not tested on a system with a wide screen. Beside this problem; no other bugs were observed. The performance of this game was superb on the system used for testing; with no noticeable slowdowns. The user interface is easy to navigate and understand. With the exception of the issue of text being cut off by the side of the screen, the menu system is excellent. Installation of the game is done through an installation program rather than requiring the user to manually decompress the game. Furthermore, installation of modifications is also conducted
through a streamlined process.

The issue of modifiability for this game deserves special attention. As the primary download does not include any songs (except for the menu music) or step patterns, this game is completely reliant on user contributions for its value. User contributions come in the form of “step patterns” which integrate a song, a background (either static or animated), and a predefined set of steps to which the users must respond into, all contained in a single playable unit. A wide variety of these step patterns are available for download. Often, these step patterns come in packages including multiple songs. It is worth while noting, however, that many if not most of these user generated step patterns constitute unauthorized derivative works. Often the songs and background graphics used are copyrighted material which the creators of the step patterns most likely use without the permission of the copyright owners.

Taken together, the graphics, sound, flexibility in game play modes, and physical activity; in addition to the wide variety of music and step patterns provide a great degree of both entertainment and enjoyability. For this reason, my subjective analysis of both entertainment value and replay value are high. This piece of software is highly polished; and seems quite comparable to commercial games. My overall bottom-line rating of this game is therefore “Very good”. Because of the large number of user generated step patterns, this game can be said to be, if anything, superior to its commercial counterparts. The fact that much of the downloadable content for this game makes unauthorized use of copyrighted content, however, raises interesting issues for this study. On the one hand, piracy is essentially allowing for the creation of a superior product. This raises the question, however, whether music which can be used for this type of work would be created in the first place in the presence of wide-spread copyright infringement or absent copyright law. An argument can be made, however, that music was being produced long before the introduction of copyright law and continues to be produced despite the large amount of copyright infringement currently occurring. Furthermore, the very fact that
creative works such as open source software and user modifications to video games are being produced demonstrates that sophisticated content can indeed be produced without the monetary incentives created by copyright law. The use of copyrighted content in the production of complicated and novel derivative works also demonstrates the importance of artistic inputs for the further development of cultural and entertainment goods. Thus, this game is significant both because it is the most popular of the FOSS games on SourceForge which meet the criteria for inclusion in my sample described in the previous chapter; and because it directly relates to the question of the applicability and impact of copyright law in today's digital environment.

**Battle for Wesnoth:**

A pre-release of version 1.4 (1.3.19), which was released in February 2008 was reviewed. According to the SourceForge download statistics, Battle for Wesnoth has been downloaded 2,128,682 times. The size of the primary download is about 107 MB. Although no user ratings for the exact version reviewed are available on download.com, ratings do exist for the Macintosh version 1.3.11 of this game. Out of twelve ratings, the average user rating is 4.5 out of 5.

This game falls into the broad genre of turn based strategy. In this game the player must complete missions by creating military units, gathering gold, and defeating enemies in a fantasy world. Game play takes place on a 2-dimensional tile based map. Only part of the map is visible at a time, but the viewable area can scroll through the entire map on both the x-axis and y-axis. A minimap in the top right corner of the screen can also be clicked on for quick navigation within the game world. Game play takes place within “campaigns”, which consist of a series of scenarios; each scenario within a campaign takes place on a separate map.

Time is used in several different ways in this game. As is typical in turn based strategy games, military units can only move a finite number of spaces per turn. Also, within each scenario, the number of turns the player is allowed to use is limited; thus creating an incentive for the player to complete the
designated mission of the scenario quickly. Every turn, the quantity of wealth possessed by the player (the primary resource to be collected in the game; which allows users to build and support more military units), is adjusted each turn based on how many units are being supported and how many villages are controlled. Within each scenario, another way time can be said to be passing is through advancement of the narrative, which sometimes occurs when a player completes a stage within the larger goal of the scenario (such as when a “hero” reaches a specific point on the map). Also, within each scenario, the time of day continually changes (a set number of hours pass with each turn); this impacts game play since, depending on the affiliation of the various units, their combat abilities are affected by whether it is day time or night. Game time also apparently passes between each scenario within a campaign; as the narrative is advanced and the location changes (the change in location is represented by the fact that each scenario takes place on a different map). On another level, time passing throughout each campaign is implied by the fact that new military units and heroes become available to the player and heroes gain experience points.

Narrative is also present on multiple levels within this game. The fantasy setting of the game world constitutes a larger “outside” narrative which provides structure and context to the game. Within each campaign the narrative progresses both between and within each scenario. Dialogs and story cut scenes (again, both between and within scenarios) move the story along. Within the scenario when dialog occurs, the “camera” (the portion of game space displayed on-screen) moves as to show the character which is speaking, and a dialog box appears with a more detailed picture of the character and the words the character is saying.

The usage of space within the game is fairly typical for the genre; yet it works quite well in achieving the goals of the game. The fact that both time and narrative are present on multiple levels and both play a significant role in game play demonstrates a fairly high degree of sophistication. Taken together, these components work together well in creating a compelling game world.
The graphics of this game are completely 2-dimensional, and are tile-based (large landscapes are made up of tiles). Thus, this game can be said to be graphically comparable to strategy games from the mid 1990s. While the graphics are relatively simple, there is a fair degree of graphical variety to be found in the game. Also, the graphics are quite aesthetically pleasing, and scrolling within the game world is smooth and the limited animation in the game helps to bring the environment to life. Not much discussion is necessary for the sound effects and music for this game; except to say that they both fit well in the fantasy setting and generally add to the experience of the game. In terms of playability, the controls work well considering the style of game play; and a tutorial campaign is included to help familiarize the player both with the controls and the general rules of the game. Overall, these components combine to produce an enjoyable game with enough variety and content to provide a medium level of replay value.

The question of complexity can be addressed in a couple ways. On the one hand, the relatively antiquated graphics (which is likely the primary reason for the relatively small size of the primary download) represent relative simplicity. However, the large variety of military units and the sophisticated usage of time and narrative could be said to represent a fairly high degree of complexity.

In the evaluation of the game, there were no perceivable bugs, and the game ran flawlessly and smoothly (thus the performance on my system was high). The user interface was clean and easy to use, and an install program was included. In addition, the game has an integrated system for installing user modifications (in the form of extra campaigns), making it extremely easy to obtain extra content. These extra campaigns add a tremendous amount of value to the software, and represent the work of a vibrant and active community of individuals taking part in improvement of the game.

My overall “bottom line” assessment of the game is “Good”. While it is highly polished and well designed, the absence of modern-looking graphics which would be expected in commercial turn based strategy games prevents me from assigning this game a rating of “Very good”.

Sauerbraten Game & Engine:

The version of this piece of software reviewed was released in December 2007. The number of downloads according to SourceForge is 1,323,240. The size of the primary download is about 215 MB. Unfortunately, no ratings for this game are available on download.com, but user ratings for the Linux version can be found at http://www.happypenguin.org/show?sauerbraten (a Linux related site); of the 15 user ratings, the average rating is 4.64 out of 5. The developers of this game note on their web site (http://sauerbraten.org/) that the game engine is licensed under the ZLIB license, which allows it to be used in commercial games; and it is indeed, according to the developers, being used by third parties in commercial games. The developers also note that they run a business offering technical support to developers using their engine in commercial games. During the evaluation of Sauerbraten, at least part of the purpose of the game appeared to act as a “technical demo” for the engine.

The game falls clearly in the category of a First Person Shooter (FPS) and has both a multiplayer mode, and a single player mode in which the player must shoot monsters without being killed. According to the Sauerbraten website: “Much like the original Cube, the aim of this game is not necessarily to produce the most features & eyecandy possible, but rather to allow map/geometry editing to be done dynamically in-game, to create fun gameplay and an elegant engine.” Evaluation of the editing capabilities is, however, beyond the scope of this study. When played in multiplayer, both deathmatch games (which simply involve killing the other players) and a mode called capture (in which players must control points on the map) exist. Due to the extremely poor AI, however, the capture mode is not available in the single player mode.

As is the case in all FPS games, the player navigates in a 3-dimensional world and has a first person perspective. Timing in this game is only important to the extent that the user must react quickly to the environment and time movements carefully. Narrative does not play a role in this game. For the purpose of this game, however, the usage of these elements is acceptable.
On the surface, the graphics in this game seem fairly impressive. The geometric detail seen in the levels appears somewhat superior to those seen in Unreal Tournament 2004 (UT2K4) (released late 2003). For example, some levels include fairly realistic looking jagged rock faces and plant life. However, the geometric detail of the monsters seen in the single player mode is quite limited. The texture detail of the world, however, is fairly impressive. The way texture is used actually seems to give a sense of depth to features such as road tiles. The lighting detail is relatively advanced; perhaps comparable to the lighting detail of Doom 3 (2004). The player, the monsters, and objects in the game cast realistic looking shadows. Moving light sources, such as projectiles and explosions, however, do not affect the lighting of the surrounding environment. Water reflects and refracts light in a fairly convincing manner, although the water only reflects the environment; and not the player or monsters. The physics modeling, however, seems fairly unimpressive. Some objects, such as grenades, do bounce around a bit, although not as convincingly as in Nexuiz (discussed below). When enemies are killed, sometimes they are blown apart, occasionally producing fragments which bounce around, although again, not as convincingly as in Nexuiz. Often, however, when an enemy is killed it simply falls to the ground in an unrealistic way; as might be expected in games from the 1990s. Another way in which physics is not properly modeled is that explosions and what appear to be flaming projections can exist underwater. Some simple particle effects are used; for example, when a bullet hits an enemy, small sparks will fall to the ground and vanish. Explosions do not look realistic, and often do not seem to influence the environment around them. Also, when a weapon is shot at a wall or floor, it does no damage; which negatively affects the experience of realism. One level evaluated for this study, however, leads me to believe that some reasonable (although not cutting edge) physics effects may be included; although not fully utilized. In this level, there were large stacks of chairs and large chains of barrels which would explode when shot. When a barrel exploded near a stack of chairs, the stack would tumble to the ground; although in what looked like a fairly simple way (the chairs did not, for
example, tilt or rotate either in response to the shock wave of the explosion or when falling, and they all landed in an upright position). After the explosion of the barrels, some derbies was produced and bounced around for a short period before vanishing. It should be noted that the physics in UT2K4 were significantly more compelling. For example, in UT2K4, when a player is killed with a projectile, the body will sometimes bend as it flies through the air before bouncing on the floor to its final resting place. On a positive note, a large number of maps are included with the game; and the environments depicted in these maps are extremely diverse. These environments range anywhere from ancient ruins to urban environments.

The use of sound and music in this game are sufficient and rather standard for games in this genre. The sounds produced by weapons sounded as one would expect in an FPS; and both the player and the monsters make groaning sounds when hurt or killed. The various levels also offers a variety of music. Although not especially memorable, the music nicely complements game play.

In terms of playability, the controls are adequately configurable allowing the player to adjust the game to her preferences. The controls are highly responsive, and navigation within the environments is easy.

At least in the single player mode (multiplayer mode was not formally analyzed for reasons described in the previous chapter), the entertainment value of this game is sadly limited for a few reasons. Generally speaking, the game has a highly unpolished and unfinished feel. For example, as soon as the program is opened, the player is immediately placed within a map (in most games, when the application is launched, the player starts within the menu system rather than directly being placed within a map). What's more, the map loaded has no enemies; thus, while the player can explore this map, there is really no game to be played in this map. When the player launches a single player level with enemies, the monsters the player is presented with are unimpressive. The design of the monsters, although somewhat diverse, nonetheless seem quite generic in their appearance; and the monster
designs do not fit into any consistent theme. The AI of the monsters is extremely limited; they essentially seem to run right toward the player and simply attack. Most likely the reason single player mode consists of fighting monsters rather than “bots” (AI virtual players bound to the same rules as the player herself) is because of the limited AI. The relatively limited physics modeling and lack of such expected features as being able to damage walls by shooting them also cause this game to feel unpolished. For these reasons, both the entertainment value and replay value can be said to be moderate at best.

The game is also fairly mixed in terms of complexity. The fairly advanced graphics imply a highly advanced game. In addition, there is a fair variety of weapons and power-ups (although fewer than can be found in Nexuiz, which is discussed below). On the other hand, there is not nearly the degree of diversity of game modes as is seen in Nexuiz and many deathmatch style commercial FPS games. The limited physics and AI can also be said to indicate lack of complexity.

This game actually appears to be nearly free of bugs. During my assessment, the only bugs encountered were related to the poor AI. The monsters sometimes appear to get stuck due to their lack of intelligence. The user interface offer a fair degree of configurability (both in terms of controls and graphics); and are arranged in a fairly reasonable manner. By default, a 3-dimensional menu system is used, which, while interesting, is not easy to use. For this reason, the 3-dimensional menu system was deactivated in favor or a more traditional looking 2-dimensional menu system. The primary download for this game takes the form of an install program. The performance of this game, on the system on which it was tested, is flawless. Even with maximum details and at a resolution of 1600x1200, the frame rate rarely deviates significantly from sixty frames per second.

As described by the developers on their website, a major focus of this game is to provide easy level editing. A large number of maps are included with the game; and many more are available for download. For this reason, the game can be said to be successful in this regard; and the ability to install
more maps greatly increases the value of this game.

Despite the fairly impressive graphics, my bottom line assessment of this game is only “Fair”. This project seems to have a great deal of potential, but the lack of polishing taken together with the limited physics and AI greatly diminish the actual experience of gameplay. If development of this game continues, however, it could one day earn a higher bottom line score.

**Scorched 3D:**

The version of this game which was reviewed was released in February of 2008. The download number according to SourceForge for this game is 1,292,524. A slightly older version, released in January of 2008, is rated on download.com. The average user rating (out of 131 ratings), is 3.5 out of 5. The size of the primary download for this program is about 46 MB.

It is difficult to definitively classify this game within the genres outlined in the previous chapter; but it has some elements of turn based strategy; as the game is based on turns; and the player must make some strategic decisions (such as which upgrades to purchase between turns). However, this game does not involve gathering resources and many other elements frequently found in turn based strategy games, so the genre could also be defined as “Other”.

According to the description of this game provided on the SourceForge project page, “Scorched3D is a 3D remake of the popular 2D artillery game Scorched Earth. Scorched3D can be played against the computer, other players and remotely across the Internet or LAN.” Players take turns firing artillery at each other. In order to succeed, players must decide which weapon to use, adjust the direction of their weapon (both in terms of rotation and vertical tilt), adjust how much power to use for the weapon (which impacts how far it will fire), and also account for the strength and direction of the wind. By successfully destroying enemies and avoiding being destroyed, players earn points which can be used to purchase better weapons, defensive equipment, and other powerups for use in subsequent rounds.
In terms of usage of space, on each level the players (both human and AI) are randomly placed on a 3-dimensional terrain. The player controls a “camera” which allows her to rotate her perspective of the world, pan over the world (on both the x-axis and y-axis), and zoom in or out. The camera must be adjusted in order to gain different perspectives of the world. While the world presented within each level is spatially finite, shots can be fired outside the map which constitutes a wasted shot. In order to aid the player in orienting through the world, both a 2-dimensional and 3-dimensional minimap is present.

The management of time is turn-based. Within each round, players take turns lobbing shots at each other. Once a player (either human or AI) takes a shot, it is the next player's turn. In an alternative game play mode, all players have the opportunity to fire simultaneously, and the next turn begins once all players have fired. There is a time out between each round in which players can purchase upgrades based upon the number of points they have earned. Shots are animated, which takes up time in the real world. Narrative does not play a role in this game.

Although graphics in the game are rendered 3-dimensionally; geometric detail, texture detail, and lighting detail (even when configured to their highest settings), are quite limited. The graphics are, however, aesthetically pleasing, and, considering the type of game, sufficient in detail. The sound and music are appropriate for the game and overall add to the gameplay experience.

In terms of playability, the controls work well and are configurable. Furthermore, a tutorial helps familiarize the player both with the controls and with the various aspects of gameplay. As far as entertainment value and replay value go, this game provides entertainment for a short time; thus the entertainment level can be said to be relatively high, and the replay value can be said to be moderate.

In terms of complexity, although the graphics are relatively simple and there are there exist a limited number of gameplay modes, the fact that the AI players can be configured according to the needs of the players, and the fact that there are a large number of items which can be purchased, make
this game as complex as it needs to be considering the goals of the game.

The menus are easy to navigate and use, and provide all necessary options. The download consists of a setup program for easy installation. Furthermore, in my evaluation, no bugs were encountered. The game runs well on the system on which it was tested.

Although the maps in which gameplay takes place are computer generated; there are, however, a large number of user generated modifications available for download. These modifications greatly expand the variety of graphics available and add additional gameplay styles; thus they can be said to add significant value.

My final bottom line rating of this game is “Fair”; although it is highly polished and runs well, the relative simplicity combined with the limited replay value limit how much utility can be derived from this game.

Nexuiz:

As this game represents perhaps the most sophisticated game in my sample, it seems appropriate to briefly describe a couple commercial games in the same genre which represent the current cutting edge in graphics and physics. The game Crysis (2007), which will be referred to in the description of Nexuiz, demonstrates the extremely high bar which FOSS games compete in order to be described as graphically on par with state of the art commercial games. Unreal Tournament 3 (UT3), also released in 2007, is on its face somewhat less visually impressive than Crysis, but still useful as a benchmark against which to measure Nexuiz.

Unlike Nexuiz, the single player mode in Crysis is narrative driven. Without going into depth describing the plot of the game, it is sufficient to say that the player lands on a jungle island inhabited by hostile North Koreans and extra-terrestrials. Graphically, both plant life and human development are at times difficult to distinguish from video footage (although they are completely computer rendered). Lighting and shadow detail is taken to the point where shadows from individual leaves on
trees (which are being blown around by the wind) can be seen on the ground and on other trees and walls. This lighting detail makes the game extremely life-like. The physics detail in Crysis is also taken to the extreme. For example, rockets can blow apart buildings, and trees can be knocked over with projectiles. Dust can be seen blowing in the wind. As we will see, although Nexuiz (especially version 2.4) is quite graphically impressive, it is unable to reach the extremely high graphical bar set by Crysis.

UT3, both stylistically and in terms of gameplay, is more comparable to Nexuiz than to Crysis. In UT3, as is the case with Nexuiz, the single-player mode consists of the player playing against computer controlled bots in a series of arenas. The geometric detail, although not on par with Crysis, is still quite complicated. Numerous levels in UT3 are set in fairly detailed cities. Although not photorealistic, the stylistic setting seems to be more surreal and science fiction based. For the purpose of comparison to Nexuiz, the most useful graphical characteristic of UT3 seems to be the lighting detail, as Nexuiz 2.4 seems to be approximately on par with UT3 in this regard (and, although lighting in UT3 is less sophisticated than in Crysis, it can still be considered close to the cutting edge). The impressive lighting features in UT3 include, but are not limited to, realistic shadows (which are casted on both players and immobile objects, and casted by both moving and stationary light sources), reflections (including distortions depending on the reflecting material), and shading of surfaces depending on nearby light sources.

Out of all the games reviewed for this study; this is the game which seemed most impressive to me. The version formally reviewed for this study is version 2.3, which was released in May of 2007. However, after the formal data collection process for this research, version 2.4, which made some significant improvements, was released and will also be discussed in some detail. The SourceForge download number for this software project is 1,265,153; the download size is 272 MB (the download size for 2.4 about 100 MB larger). There are no user ratings for version 2.3 on download.com; but the
Windows release of version 2.1 has an average (out of eleven ratings) of 4 out of 5; and the Macintosh release of version 2.4 has an average (out of six ratings) of 5 out of 5.

This game falls squarely within the genre of First Person Shooter (FPS). Unlike some FPS games, there is no focus on narrative; game play takes place in individual rounds within separate maps (although several modes of game play and rule sets exist). The player has the choice of either playing against other humans (either over a local network or the Internet) or playing against “bots” (which are AI simulations of other players). Game play is quite diverse; and some levels are designed for simple “deathmatches” in which players simply try to kill the other players as many times as possible; “team deathmatches”, in which players try to kill players on the opposing team as many times as possible, Capture the Flag, and Domination (in which teams have to control various locations on the map in order to score). Other variations include levels with reduced gravity, rule sets in which players are healed by damaging the health of other players, and a mode in which a single shot with a special weapon instantly and automatically kills other players. Nexuiz uses the DarkPlaces engine, which is an extremely heavily modified version of the open sourced Quake engine (originally developed by Id Software). The modifications have been so extensive and have been made over so many years, however, that visually the engine is not recognizable as being related to the Quake engine.

The usage of space in this game is quite typical for the genre of FPS games. The player navigates through a detailed 3-dimensional world and has a first person perspective. Within the levels, however, there are items such as “teleporters” which allow the user to instantaneously transport herself to a distant location on the map and spring boards which push the player into the air; allowing her to cover large distances in a short amount of time. Time is only significant to the extent that it is essential for a player to react quickly to the environment and time actions (such as moving or shooting) carefully. Narrative does not play a role in this game; and there is not even an obvious background narrative to give context to the game. Due to the nature of game play, however, the lack of narrative
does not subtract from the experience of the game. The usage of space and time; while fairly standard for the genre and not very original, work well considering the nature of the game.

In the evaluation of this game, the aspect which impressed me the most was the graphics. The geometric detail and detail of the textures in the game, while clearly not on par with current cutting edge commercial FPS games such as Crysis (2007), are roughly comparable to high end commercial games released a few years ago such as UT2K4 (released near the end of 2003). The geometric and texture detail seemed somewhat less detailed than those seen in Sauerbraten; but not by much. The detail of lighting (such as shadows and changes in the brightness on textures in response to static and moving light sources), on the other hand, appears closer to that of contemporary commercial games. In version 2.3, both the player and other characters in the game cast shadows, and light greatly increases the realism of the environment. Version 2.4 makes even further use of lighting; it supports features such as realistic reflections of both the environment and the player on water; and improves the visible detail of lighting over that present in 2.3 in ways which are hard to characterize. Furthermore, in 2.4, projectiles, while in the air, cast light which affects the appearance of the surrounding environment. The level of lighting detail in 2.3 seems to be roughly comparable to in Doom 3 (2004); while the lighting in 2.4 appears close to being on par with Unreal Tournament 3 (released late 2007). As a whole, the superior lighting detail to Sauerbraten, especially the 2.4 lighting detail, more than compensated for the slight deficiency in texture and geometric detail in comparison to Sauerbraten. The various maps included with the game present the user with a wide range of environments in which to play; presenting fairly diverse graphics. The physics modeling in game play; while not on par with those seen in Crysis, in which large structures such as buildings can realistically be blown apart with rockets or grenades; are approximately on par with those seen in UT2K4. Explosions look fairly realistic (and, at least in 2.4, produce far more realistic lighting effects than seen in UT2K4); certain projectiles, such as grenades, bounce around the environment realistically; walls can be damaged
(although not destroyed) by weapons; and certain weapons cause enemies to be blown apart into little pieces which bounce around. Thus, as a whole, the graphics can be said to be on par with the commercial games of a few years ago; while the lighting effects found in 2.4 seem comparable to some contemporary commercial games. It should be noted that Nexuiz appears to attempt to imitate both the visual style and the gameplay style of the Unreal Tournament series of games (although lacks features such as the vehicles found in UT2K4 and Unreal Tournament 3).

The sound effects seem typical for games in the FPS genre. Weapons make the type of sounds one would expect and both the player and enemies make sounds when they are injured. Furthermore, an announcer makes comments such as “Five Frag Combo” (indicating the player has made five kills since the last time she was killed). This clearly seems to have been borrowed from the Unreal Tournament series of games. The music, while not particularly memorable, is fast paced and nicely complements the experience of the game.

The game is highly playable; controls can be configured to meet the preferences of the user, and the mouse sensitivity can be adjusted. In version 2.3, one feature which would have enhanced gameplay was missing: Information about the current score within a match could not be seen except when the player was dead. In 2.4, however, a box in the bottom left hand corner of the screen is included which tells the current score of the player, the players rank, and the difference in score between the player and the enemy with the highest score.

Overall, the entertainment value and replay value of the game are quite high; although some minor improvements could add a great deal to the overall experience of the game. For example, in single player mode, one would expect that details such as the number of bots and the goal score along with other rules could be configured. However, despite the fact that such features seem universally present in commercial FPS games which focus on death matches, these features are conspicuously lacking.
In light of the sophisticated graphics, diverse modes of gameplay, the large number of weapons and powerups, and relatively large download size compared to the other FOSS games in my sample; this game is highly complex. The primary area in which complexity is lacking is the poor configurability mentioned above.

Although as a whole this game has a highly polished and professional feel; some bugs were observed. In version 2.3, the game seemed to run slowly unless I ran it shortly after resetting the computer (this issue seemed to have been resolved in version 2.4). In version 2.4, issues with sound would occasionally occur. At times the sound effects would be out of sync with the graphics on screen, so, for example, the sound effect for the firing of a weapon would not be heard until between 0.25 and 0.5 seconds after the weapon actually fired. Also, in 2.4 the sound effects would also occasionally stutter. Version 2.4 also crashed a couple times; and on two occasions colors in Windows appeared distorted until the computer was reset. Considering the complexity of the game, however, these bugs were not excessive or unreasonable.

In both 2.3 and 2.4 the menus were easy to navigate; and offered a fair degree of configurability in terms of graphics, controls, and sound. In 2.4, however, the menus had a more pleasant visual appearance (transparencies and animation were used in the menus); and the menus in 2.4 also allowed for finer configuration of the graphics. The installation of the game is done through decompression of a file rather than through the use of an installation program. Although it is possible to use user generated maps; I was unable to make extra maps I downloaded to appear in the list of maps available during game play (a similar problem was encountered with the commercial game Unreal Tournament 3; although in the case of the commercial game it was eventually resolved). User generated maps are, however, significant, since with each new version additional user generated maps are added.

In version 2.3 the game ran reasonably smoothly even with all detail settings at their maximum and at a resolution of 1600x1200. In 2.4, however, as the graphics are more sophisticated, resolution
had to be reduced in order for reasonable performance to be maintained. This was somewhat disappointing as, while the graphics are impressive, the game does not seem to perform as efficiently as commercial games of comparable complexity. The efficiency of the DarkPlaces engine is, however, continuously being improved; and the developers of Nexuiz note on their website (http://www.alientrap.org/nexuiz/) that with each subsequent version efficiency is improved.

While this game impressed me more than any other game in my sample, my bottom line assessment of it is only “Good” rather than “Very Good”. The reason for this is because of the limited configurability of game play and the less than stellar performance. Having observed the significant improvements between Version 2.3 and Version 2.4, however, it seems likely that this game will continue to improve in subsequent releases.

Armagetron Advanced:

The version of this game which was reviewed was released in February 2008. The number of downloads according to SourceForge is 1,000,408. The download size for this game is quite small; only approximately 3.6 MB. Out of the five user reviews for this game on download.com, the average rating is 4.5 out of 5.

In terms of the genres described in the previous chapter; the most appropriate description for this game is “Action”. According to the description on the SourceForge project page, this game is “Another version of the simple action game modeled after the lightcycle sequence of the movie Tron.” (Tron was released in 1982). This game supports both single player and multiplayer modes (in single player mode, the player competes against AI players). The player(s) are represented by “lightcycles” which leave a trail which can trap other players. When a cycle is traveling near a trail, it accelerates. The goal of the game is to outlive all other players. Players are killed when they either crash into a trail left by a cycle or a wall. Success requires both quick reflects and on the fly strategic thinking.

By default, the user takes a third person perspective of her cycle, and has the ability to zoom in
or out. It is also possible, however, to take a first person perspective (i.e. viewing the game world from inside the cockpit of the cycle). Each arena is a 3-dimensional square area whose size can be set by the user. Although the area is rendered in three dimensions, players only move on the 2-dimensional plane of the floor. The finite nature of the playable area of each area is essential to gameplay; as it makes it possible for players to trap each other with the trails left by the cycles. Time is used in a couple different ways. Successful timing and quick reaction are essential for survival within the game. Within each arena, game time passes at a uniform rate. The user has the ability to adjust the “speed” of each level, which can be seen as compression of time to the extent that when speed is doubled, spaces are covered in half the time. In the configuration of each arena, the length of the trails, defined by the amount of time they persist, can be defined. How long the trails persist is specified in seconds of real time, and is independent of the speed of gameplay specified by the player. Although narrative is not explicitly defined for this game; the fact that this game draws from a scene in the movie Tron can be said to provide a background narrative. These conceptual notions, taken together, create a unique, fast-paced, and enjoyable game.

To evaluate graphics, it must be kept in mind that this game attempts to recreate the lightcycle game from the movie Tron. In order to better evaluate this game, especially in terms of graphics, the lightcycle scene from the movie was viewed on YouTube (an online collection of short videos contributed by users). As the movie was created in 1982, the graphics in the movie were quite simple by today's standards (it should be noted that an arcade game based on the movie was also released the same year; as computer hardware was so limited at the time, there is simply no comparison between the graphics of Armagetron Advanced and the arcade game). Also, in the movie the camera is used in such a way as to be conducive to the medium of film rather than video games. Thus Armagetron Advanced attempts to modernize the graphics while remaining true to the movie, and also changes the usage of the “camera” (in this context the word camera refers to the point of view of the user) in order to make
the game playable. Thus in this game, the graphics are intentionally designed to not look realistic; but rather to appear computer generated.

Both textures and geometry in this game are extremely limited, but this seems to be more of a stylistic decision than a technical decision as the game attempts to recreate a scene from an old movie. The simplistic textures and geometry, however, most likely greatly simplified the development process of this game. The textures, however, can be altered by installing user modifications. In order to make this game feel more graphically modern than the scene in the movie, however, detail is added in several ways. There are two layers of sky which are moving, which makes the graphics look more lively. Furthermore, the floors reflect the sky, the walls, the players, and the trails left by the players; thus making the graphics look more sophisticated than the graphics in the film. Various “special effects” are also used in the game in order to enhance the visual appearance. For example, sparks fly when the player is rubbing against a trail left either by herself or another player (either AI or human). When a player dies, the explosion is simplistically depicted as lines flying in all directions; this simplicity, however, seems to be a stylistic decision, as realistic explosions would not fit with the theme of the game. In short, the graphics, while intentionally simplistic, are great looking and do an excellent job in creating a feel consistent with the scene from Tron.

The sound effects in this game also attempt to create a computer generated feel. They are not, however, an exact copy of the sound effects in the movie. The engines of the bikes make a sound which both resembles the sound of a motorcycle and sounds “computer generated”. The pitch of the engine sound increases as the player accelerates and decreases as the player decelerates. There are also sound effects for actions such as turning, breaking, rubbing against a trail, and crashing (an explosion sound). Together, these sounds add greatly to the experience of the game. Music, however, is not included. It is unclear whether this is a stylistic decision or an attempt to simplify development. The lack of music, however, does not seem to make the game less entertaining.
In terms of playability, the controls are both simple and highly configurable. The one weakness in the control scheme, however, is that control of the camera is not intuitive. Although the user has the ability to zoom in and out or switch to a first person perspective, due to the fast paced gameplay and the necessity of keeping both hands on the primary controls, the actual ability to control the camera during gameplay is limited. This, however, is not nearly as serious a problem as it may seem. When the player is not actively altering the point of view of the camera, the camera is automated, and in general does an excellent job in producing a highly playable point of view.

In terms of entertainment and replay value, this game is capable of offering many hours of entertainment. It offers an adequate degree of entertainment to draw the user back over a long period of time. On the other hand, after playing for a long period of time, it is easy to become bored. Thus the entertainment value can be said to be high, and the replay value can be said to be between medium and high.

In terms of complexity, the graphics, as described above are relatively simple. The gameplay is also relatively simple, and the size of the download is quite small. However, gameplay is highly configurable. The user has the ability to alter many aspects of gameplay, such as the size of the playable area, how long the trails left by the player and AI characters persist, whether the game is played as a “team game”, what the conditions are for winning, the speed of gameplay, the blast radius of the explosions which occur when a player or AI character is killed, and whether a player's or AI's trail persists after she is killed. Thus, considering the goals of the game, it can be said that the game is as complex as it needs to be.

No bugs were observed during my evaluation, the menu system is easy to navigate, good looking, and offers a high degree of configurability. The download for the game takes the form of an installation program. Also, the performance of the game on my system is flawless and smooth. These elements, taken together, give this game a highly polished feel.
Despite the simplicity of this game, user modifications are supported. Most of these modifications, however, take the form of modified graphics (textures) and sound effects. Thus the primary purpose of these modifications is simply to alter the look and feel of the game and not affect the actual gameplay. These modifications, however, improve the replay value of the game.

My bottom line assessment of the quality of this game is “Good”. It was tempting to assign a rating of “Very good” to this game, but the simplicity of the graphics and gameplay indicate that this game cannot directly compete against cutting edge commercial games. This game, however, could compete quite well with budget commercial games; and if properly marketed as a commercial game could probably sell quite well if priced in the $10 range. Also, if this game were adapted to a handheld device such as the PlayStation Portable (PSP) and sold commercially it could likely be quite successful. Overall, this game demonstrates that high quality and entertaining FOSS games can be developed; and games do not require state of the art graphics to be highly enjoyable.

**Freeciv:**

This game is a clone of the commercial game Civilization II (1996) with similar appearance and gameplay. It falls squarely in the turn based strategy genre. The version evaluated for the study was released in January of 2008. The SourceForge download number is 901,470. No ratings for this game are available on download.com, but according to “Softonic” (ratings found at [http://freeciv.en.softonic.com/](http://freeciv.en.softonic.com/)), the average of twelve user ratings is 7 out of 10; which when converted to a five point scale is 3.5 out of 5. The size of the primary download is only about 8.7 MB.

Just as in Civilization II (and the other games in the Civilization series), the player must build cities, manage public opinion, set tax rates, develop terrain, build a military, and develop technology while competing with other civilizations (the other civilizations are controlled by the computer (although a multiplayer mode is also available). The world exists on a 2-dimensional map. At any time, only a limited portion of the world can be displayed on screen; thus the player must scroll on the
x-axis and y-axis or click on the minimap to change which portion of the world is visible. Terrain is only revealed after it is explored; unexplored terrain appears black (although terrain can also be “explored” through diplomatic relations in which maps are traded with AI players). A “fog of war” effect also exists; the player can only see what is happening within a portion of the map when it is within visual range of a city or military unit belonging to the player. Within the 2-dimensional world; each tile represents a portion of the terrain. Different types of terrain have different properties. For example, military units move faster on some types of terrain than others. Some types of terrain allow the player to build roads or railroads which allow faster movement and greater economic productivity. Also, some terrain types allow irrigation or mines to be built, which produce certain advantages.

Time is quite important in this game. As Freeciv is a turn based strategy game, the player can only take a finite number of actions during a turn. During a turn, a player can move military units a finite distance, alter what production is taking place in each of the cities, and take various other actions such as adjusting the tax rate and monetary usage (such as allocating portion of the tax income to science or luxury or profit which can be used to further military or social production). Other actions which can be taken during a turn include beginning a revolution in order to change government type (the government forms in the game include civil war, anarchy, communism, democracy, despotism, monarchy, and republic). Once a player finishes her turn, the AI players take their turns. Each turn represents the passage of a certain number of years. The game begins at 4000 BC, and depending on settings, can progress into the future.

As Freeciv is a nation building game, narrative can in a sense be said to be defined by the user. The history of the civilization and the world will depend on the actions of the user. However, there are certain limitations to the way the player can develop her civilization. For example, the technologies available for research are limited, and organized into a “technology tree” (meaning some technologies have prerequisites; a player must research basic technologies before she can research advanced
technologies).

Taken together, these elements produce a game virtually identical to Civilization II. While they work together to produce an interesting game, they cannot be said to be original.

The graphics in this game are tile based and have minimal animation. The graphical detail of the game is comparable to what is seen in Civilization II. Thus the graphics can by no means be said to be anywhere near graphics seen in contemporary commercial games. The sound effects are reasonable and similar to those in Civilization II. Unlike Civilization II, however, Freeciv has no music.

While the controls work fairly well once the player is familiar with them, they are not configurable. There are also some annoying aspects of the control system. For example, the player cannot scroll through the game world simply by moving the mouse to the edge of the screen. Also, the lack of a tutorial increases the learning curve for this game; especially for those unfamiliar with the Civilization series of games.

The game is somewhat entertaining, but the fact that the newer games in the Civilization series are far more advanced, and that Freeciv offers little which cannot be found in Civilization II, the entertainment value is quite limited. The replay value is moderate. It, however, would be high if it were better designed and modernized to the level of technical sophistication as the more recent Civilization games.

Although the download size is small and the gameplay is comparable to a commercial game from 1996, in a sense the game can be said to be quite complicated. The reason for this is the large number of technologies, government types, military units, city improvements terrain types, and other such features. Furthermore, all aspects of the game are described in a fair amount of detail in the help menu.

The only bug observed was a crash of the game when I attempted to find a multiplayer game. As far as the user interface goes, the menus are usable but seem somewhat poorly organized. They
offer a high degree of configurability (there are actually configuration options for the game which are not seen in the Civilization series of games), although these options are not intuitive or easy to use. Installation is done through the usage of a setup program. There seems to be a fairly active community developing modifications for this game, and the modifications include new tile sets and rule sets which allow for significant transformation of the game. The performance of this game was flawless on my system.

My bottom line assessment of the quality of this game is “Fair”. Although this game offers a fair amount of complexity, it is nowhere near to being on par with the more recent civilization games or contemporary strategy games.

Secret Maryo Chronicles:

The version of this game evaluated was released in December of 2007. The download number on SourceForge is 866,478. On download.com there was only a single user rating; the user who rated this game gave it a score of 5 out of 5. The side of the primary download is 26 MB.

As implied by the title of this game, it can be classified as a “knock off” of the old 2-dimensional Mario games for the Nintendo Entertainment System (NES) and the Super Nintendo (SNES).

The levels through which the player navigates are 2-dimensional; and scrolling takes place on both the x-axis and y-axis (although there is more horizontal scrolling than vertical scrolling). Each level (represented by an individual map) is part of a larger “world”, when the player completes a level, she is returned to the map of the “world”, in which she proceeds to the next level. Once a player completes a world, she can proceed to the next world.

Timing is important for successful gameplay, as the player must precisely time her actions in order to succeed. What Wolf (2001) refers to as “cyclical” time exists in that enemies move in repetitive patterns. A timer times the play in each level, but has no impact on gameplay, and simply
tells the player how long she has spent on a level. Narrative is not apparent in this game as it is in the Mario games. However, the player moves through the level and worlds in a sequential manner, which can be seen as a form of narrative.

This game attempts to use these conceptual elements in a way which resembles the look and feel of the Mario games for the NES and SNES; and, while this game cannot be said to be original, these elements nonetheless work well together.

Gameplay is completely 2-dimensional. Within the levels, graphics exist on two planes; the foreground and a non moving background (this is similar to the game described in the appendix). In this regard, this game is different from Super Mario Bros. 3 (1988), in which the background layer also scrolled, but at a slower rate, giving an illusion of depth. The game makes some attempts to give the illusion of a third dimension, however. For example, coins within the levels, which the user can collect, are animated in such a way as to make them appear to be rotating in a third dimension. Many of the graphics are nearly direct copies of the graphics in the NES and SNES Mario games. Although the graphics were redrawn for this game, many of them conceptually copy images from the Mario games. For example, the coins, the turtles (which are a type of enemy), the “question mark” powerup boxes, and the “fire flower” can all be seen in the Mario games. The protagonist character Maryo also resembles Mario in both his appearance and name. A major difference between the graphics in this game and the Mario games for the NES and SNES is that high resolutions are supported. The old Mario games were meant to be played on a non-high definition television. Secret Maryo Chronicles, on the other hand, is meant to be played on today's high resolution computer monitors.

The sound effects also mimic the sounds in the old Mario games, and work nicely in this game. The primary download for this game does not include music; but a music package is available for download on SourceForge. Much of the music in this download closely resembles (as in being the same songs played differently) music from various Mario games. Other music packages are available
The game is fairly entertaining for short periods of time, although it is clearly not original. For gamers who have become accustomed to sophisticated 3-dimensional graphics, this game is unlikely to hold their interests for very long. For this reason, the replay value can be said to be “Medium”.

Compared to contemporary commercial games, this game cannot be said to be very complex. There are, however, a wide variety of graphics, enemies, and powerups. Overall it can be said to be similar in complexity to the 2-dimensional Mario games for the NES and SNES, but without the narrative. The user interface is well done, the menus are well designed, easy to use, and offer an adequate degree of flexibility in terms of configuring the game. Installation is conducted by using a setup program. During my evaluation of this game, no bugs were observed. Performance was flawless on my system. A large number of modifications for this game are available. These include new levels which can be played individually, new graphics, modifications to gameplay (for example new tiles and object which the user can interact with), and new sounds and music.

My bottom line assessment of the quality of this game is “good”. It has a retro feel, but plays smoothly. Furthermore, older gamers who are nostalgic for classic video games will find this game refreshing.

**Table Summarizing Results:**

Below is a table summarizing the results of my analysis. In the description of the games, graphics and sound were described and compared to other games instead of being given a definitive rating. For this chart, however, a definitive rating is given. It should be noted, however, that a definitive rating for sound and graphics is difficult to give in any valid way. Therefore, these ratings should be taken with a huge spoonful of salt. The ratings for these elements are broken into three separate columns: How aesthetically pleasing the graphics are, how technically sophisticated the
graphics are, and how much the sound effects (including music, if present) contribute to the game. All three of these measurements are on a scale from “Very Poor” to “Very Good”. The only one of these three measurements which requires any more definition is the measurement of how technologically sophisticated the graphics are. For the purpose of this chart, we will say that “Very Poor” represents graphics typical of PC games in 1995, and “Very Good” represents the sophistication seen in Crysis (thus none of the games in the sample are at either extreme).
### Table 6 Summary of Results

<table>
<thead>
<tr>
<th>Name</th>
<th>StepMania</th>
<th>Battle for Wesnoth</th>
<th>Sauerbraten Game &amp; Engine</th>
<th>Scorched 3D</th>
<th>Nexuiz (2.3)</th>
<th>Nexuiz (2.4)</th>
<th>Armagetron Advanced</th>
<th>Freeeciv</th>
<th>Secret Maryo Chronicles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size</strong></td>
<td>13.7 MB</td>
<td>107 MB</td>
<td>215 MB</td>
<td>46 MB</td>
<td>272 MB</td>
<td>380 MB</td>
<td>3.6 MB</td>
<td>8.7 MB</td>
<td>26 MB</td>
</tr>
<tr>
<td>(Program only, no step patterns included)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(Separate download required for music)</td>
</tr>
<tr>
<td><strong>Number of Downloads</strong></td>
<td>2,760,254</td>
<td>2,128,682</td>
<td>1,323,240</td>
<td>1,292,524</td>
<td>1,292,524</td>
<td>n/a</td>
<td>1,000,408</td>
<td>901,470</td>
<td>866,478</td>
</tr>
<tr>
<td><strong>Number of User Ratings</strong></td>
<td>11</td>
<td>12</td>
<td>15</td>
<td>131</td>
<td>11 (for version 2.1)</td>
<td>6</td>
<td>5</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td><strong>Average User Rating (0-5)</strong></td>
<td>4.5</td>
<td>4.5</td>
<td>4.64</td>
<td>3.5</td>
<td>4 (for version 2.1)</td>
<td>5</td>
<td>4.5</td>
<td>3.5</td>
<td>5</td>
</tr>
<tr>
<td><strong>Playability</strong></td>
<td>Very Good</td>
<td>Very Good</td>
<td>Good</td>
<td>Good</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Poor to Fair</td>
<td>Very Good</td>
</tr>
<tr>
<td><strong>Replay Value</strong></td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>Medium to High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td><strong>User Interface</strong></td>
<td>Good</td>
<td>Very Good</td>
<td>Fair</td>
<td>Good</td>
<td>Fair</td>
<td>Good</td>
<td>Very Good</td>
<td>Fair</td>
<td>Good</td>
</tr>
<tr>
<td><strong>Buggyness</strong></td>
<td>Minimal bugs</td>
<td>No bugs observed</td>
<td>Minimal bugs</td>
<td>No bugs observed</td>
<td>Acceptable bug profile</td>
<td>No bugs observed</td>
<td>Acceptable bug profile</td>
<td>No bugs observed</td>
<td>Minimal bugs</td>
</tr>
<tr>
<td><strong>Installation Program (Yes/No)</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Modifications Available (Yes/No)</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes (but limited)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Very Good</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Very Good</td>
</tr>
<tr>
<td><strong>Graphical Aesthetics</strong></td>
<td>Very Good</td>
<td>Very Good</td>
<td>Good</td>
<td>Fair</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Fair</td>
<td>Good</td>
</tr>
<tr>
<td><strong>Graphical Technological Sophistication</strong></td>
<td>Fair</td>
<td>Poor</td>
<td>Good</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
<td>Very Good</td>
<td>Poor</td>
<td>Poor</td>
</tr>
<tr>
<td><strong>Sound</strong></td>
<td>Very Good</td>
<td>Good</td>
<td>Very Good</td>
<td>Fair</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td><strong>Bottom Line Rating</strong></td>
<td>Very Good</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
<td>Good</td>
</tr>
</tbody>
</table>
Chapter 6: Conclusions, Limitations, and Future Research

Conclusions and Discussion:

My sample includes quite a diverse set of games. Furthermore, these games vary greatly in terms of sophistication and entertainment value. None of these games are as large (in terms of download size) as what can be stored on a single DVD. Of the games reviewed, Sauerbraten and Nexuiz (both first person shooters), by far represent the most advanced graphical capabilities. These capabilities seem roughly comparable to cutting edge commercial games approximately three years ago. The only game in this sample which achieves the rating of “Very good” is StepMania. StepMania, as noted above however, depends completely on user generated step patterns; many of which constitute unauthorized derivative works. Furthermore, this game conceptually copies the commercial Dance Dance Revolution games. To various extents, many of the games in the sample draw a high degree of inspiration from commercial games. Nexuiz, for example, simply seems to draw inspiration and various conceptual elements from the Unreal Tournament series; while FreeCiv is nearly identical to Civilization II.

How do open source video games compare to their commercial counterparts?

As noted in previous chapters, a large number of highly sophisticated and popular FOSS programs successfully compete with commercial software. As seen in chapter 3, much FOSS development is done by commercial firms. The economic incentives which prompt firms to involve themselves in FOSS development, however, largely do not seem to apply to video games. From my sample, the only games which appear to have a commercial element are Sauerbraten and Nexuiz. The lead developers of the Sauerbraten engine offer service and support for the engine for a fee. However, the unpolished and unfinished nature of Sauerbraten detracts from its overall quality. Nexuiz, as discussed above, makes use of the DarkPlaces engine, which is a heavily modified derivative of the Quake engine. As discussed in Chapter 3, the Quake engine was released under the GNU GPL by Id
software in order to promote modifications to their own software. Due to the extreme degree of modification, however, the DarkPlaces engine, for the most part, cannot be said to have been developed by a for profit firm. Thus, to answer the first research question, it appears that high quality FOSS games can indeed be produced without monetary incentives. While the games in my sample do not fully compete with cutting edge video games to the same extent as other FOSS programs compete with their commercial counterparts, they are nonetheless comparable to commercial games from just a few years ago, and are capable of providing a high degree of entertainment. Furthermore, as seen by the download statistics of FOSS games, it would appear that some have a fairly large following. The list of download statistics of FOSS games presented in Chapter 3 were taken only a few months after the download statistics for games in my sample was taken. Yet, as can be seen from download figures for games which appear on both lists, many of these games appear to be downloaded at a fairly rapid rate (although at least part of this difference likely is due to people upgrading from earlier versions).

One significant difference between these games and commercial games, however, is the lack of detailed “cut scenes” between levels. This likely has to do with the difficulty and expense of creating sophisticated computer generated movie clips. The only game in the sample which has any type of “cut scenes” is Battle for Wesnoth; and the cut scenes in this game are quite simple and are rendered completely in two dimensions. Another interesting observation is that several of the games include sophisticated and original music. This indicates that, at least to some extent, FOSS game developers do not need to rely on the recording industry in order to obtain music for their games. As games with quality original music demonstrate, components of games other than the source code and visual art can indeed be generated within the FOSS paradigm. An important implication of this finding is that payment of royalties to the recording industry is not necessary.

How can any disparities in quality be explained?

All the games in my sample allow user modifications, which to different degrees add value to
the software. The fact that these games allow user modifications by itself implies some degree of modularity. User contributions can be said to be modules which fit in with the larger project. Games with multiple levels can also be said to be modular, as each map has to be created independently.

Although evaluating the source code of these games is beyond the scope of this study, it is reasonable to assume that some degree of modularity exists within the development process; as all the games evaluated contain multiple components, such as sound and graphics, which all need to be coded separately.

Many of the games reviewed seem to have active communities which are involved both in playing the games and to the development of the games. For example, a large number of user generated maps for both Nexuiz and Sauerbraten can be found, Battle for Wesnoth is significantly enhanced by its integrated system for installing user generated campaigns, and StepMania derives nearly all its value from user contributions. As discussed in chapter 3, commercial games seem to derive much of their value from user generated content. This seems to be even more the case with FOSS games. Furthermore, the very fact that the source code for FOSS games is open and can be modified by anyone (although only the best modifications are likely to make it into the formal project builds), the line between users and developers is further blurred. Both source code and content (such as graphics and maps) created by users have a chance to be included in future releases of FOSS games.

The high degree of dependence on user contribution implies a connection with commercial video games which are enhanced by user modifications. As discussed in chapter 3, the relationship between commercial game developers and hobbyist modders can be said to be exploitive. Essentially these commercial developers commodify the fruits of unpaid labor in order to increase private profit. Furthermore, these commercial developers generally give little back to the modding community (although Id Software's releasing of source code under the GNU GPL represents an exception). This raises the question as to what the effect would be if these creative people, whom are generally not
primarily motivated by private profit, contributed their skills to the development of FOSS games rather than to commercial games. It can be assumed that increasing the number of technically skilled workers involved in the development of FOSS games and modifications to FOSS games would greatly improve the FOSS game scene both in terms of technical merit and in the quantity of FOSS games available.

The importance of an active user base and modding community for the success of FOSS games (and to a lesser extent, commercial games) could be seen as a basis for a model. In this hypothetical model, a FOSS game will be unlikely to succeed until a “critical mass” of users is reached. While fully asserting the validity of this model is beyond the scope of this research, it is worthwhile to briefly discuss potential implications of such a model. One hypothetical implication is a “chicken and the egg” effect. If a large and active user base is necessary for a FOSS game to be successful, how can a new project in the early stages of development hope to attract users to further develop the game? The relative success of Nexuiz can be argued to present at least a partial solution to this dilemma. As the DarkPlaces engine is based on the Quake engine, which was originally developed commercially and had a large user base long before work on Nexuiz or even DarkPlaces began; generating enthusiasm and a user base may have been easier. Another implication of this “critical mass” model is that FOSS games will be at a significant disadvantage to commercial games simply because of the lack of an advertising budget. It remains an open question as to whether FOSS games would be more visible absent a commercial game industry (perhaps users would more actively seek FOSS games if advertising for commercial games were absent).

Relying entirely on the ideas presented in the previous paragraphs, however, would be an oversimplification. As mentioned above, many FOSS games draw inspiration from commercial games. However, it should be noted that to some extent commercial games draw inspiration from each other. For example, commercial games (and FOSS games for that matter) fall into various genres which follow various conventions which have developed over time. Another way in which FOSS games may
benefit from commercial games is that some FOSS games make use of the same tools as commercial games. For example, Nexuiz supports levels designed with the GtkRadiant (available for download at http://www.qeradiant.com/cgi-bin/trac.cgi), which is a FOSS program initially designed by Id software, and is used in the development of levels for a large number of commercial games. This piece of software is significant for more than one reason. First, it fits in with the practice of commercial game developers releasing a tool in order to promote user modifications. Second, it represents a game related FOSS tool developed by a commercial firm. Third, as it is used in the development of many commercial games, a large number of people are familiar with this software and thus have already learned the skills for developing levels for Nexuiz.

The discussion in the preceding paragraphs demonstrates that at least part of the explanation as to why FOSS games are not quite on par with their commercial counterparts may be because a large enough user base simply has not developed for FOSS games in order to generate active participation. This could be blamed on the lack of advertising budgets, or due to the “chicken and egg” dilemma described above. Yet it could also be that certain elements (such as video cut scenes) are simply too expensive and difficult to make without a strong commercial incentive. It is also quite possible that lack of modularity for some components of games, such as narrative, render them difficult to be produced under a FOSS model.

**What does all this mean for the utility of copyright law in regard to video games?**

This study has demonstrated that fairly sophisticated video games can be produced under FOSS licenses (and thus without the restrictions on copying imposed by copyright law). However, other considerations need to be taken into account in order to ultimately determine the utility of copyright law as it relates to video games. Furthermore, as video games make use of other forms of media such as music and visual art, the application of copyright law to all media must be considered. As is illustrated by the discussion of the Creative Commons licenses in Chapter 1, other forms of media can,
at least to some extent, be produced without strong copyright protection. However, further research is required to establish the viability of production of other media forms in order to determine what can and cannot be produced without copyright protection. The focus of this study has been to look at contemporary FOSS games. This, however, does not provide a complete picture of the utility of copyright law for the production of video games. It should be noted that FOSS licenses depend on copyright law in order to function. Without copyright, these licenses could not mandate that derivative works be distributed under the same terms. Also, without copyright law, there is no guarantee that source code would be distributed with software. Beyond this consideration, the snapshot of the current state of FOSS games does not tell us how video games would have evolved if copyright law did not exist, or how it would evolve in the future if copyright law were abolished today. While addressing these questions in depth is beyond the scope of this study, the issue of how computer hardware relates to video games should be considered in brief. Aside from the fact that the consoles which currently dominate the video game market constitute specialized gaming hardware, specialized gaming hardware also exists for the PC. One of the most significant types of specialized gaming hardware found on the PC is the integrated hardware 3D acceleration (which, after being developed primarily for games, eventually became commonplace and necessary for modern Graphical User Interfaces, as can be seen in the Aero theme for Windows Vista). This hardware greatly improves the visual appearance and rendering speed of modern video games. Whether this and other specialized gaming hardware would have been developed absent a commercial video game industry is an open question. One can speculate that absent copyright protection, a video game industry could have arisen with a different business model and spurred the development of specialized gaming hardware. Another possibility is that a robust video game scene would have resulted as a result of hobbyist developers; thus spurring the development of this hardware. Yet another possibility is that hardware manufacturers could have become involved in producing video games in order to create a market for their hardware. These
scenarios, however, are highly speculative. Beyond these considerations, it is also possible that a video game industry could have arisen based on a subscription model. For example, it seems likely that business models similar to that of subscription based online role playing games such as World of Warcraft could still emerge even absent copyright protection. The point which should be taken away is simply that, while this study demonstrates that people will contribute their skills for development of video games without a monetary incentive, more research is required to more fully evaluate the utility of copyright law to cultural production in general and video game production in specific.

Yet another consideration which needs to be taken into account has to do with values. Do video games really need cutting edge graphics in order to be enjoyable? Furthermore, is the moderate technical edge commercial games possess over FOSS games worth the trade off of not being able to freely copy, distribute, and modify the software? Also, considering the significant capital investments which go into developing, marketing, and distributing commercial games, is the moderate technical edge they possess worth the economic costs? These are serious questions which cannot be answered in an objective manner. Rather, they are considerations society needs to take into account when deciding on laws which regulate cultural production.

The preceding paragraphs demonstrate that the question of whether copyright is useful as applied to video games has no clear cut answer. Yet it is obvious that technically skilled people are willing to contribute their effort and able to produce impressive results in video game development without a monetary incentive. Thus, it must be concluded that an active video game scene, at least in some form, could exist absent copyright law.

As can be seen in phenomena such as YouTube, Facebook, and blogging, much cultural content is currently being created by individuals for reasons other than generating profit. In the cases of YouTube, Facebook, and user modifications to commercial games, however, commercial firms are essentially commodifying the creative works of these unpaid laborers. While much FOSS development
is done by for profit firms as a matter of business; FOSS video games represent an example of cultural production which, for the most part, has neither been commodified, nor produced for the purpose of generating profit. Thus, the results of this study can be seen as an example of an emerging form of cultural production.

**Limitations and Future Research:**

While thus study has provided a snapshot of the state of FOSS games and has attempted to provide at least some insight into the observations made; both the methodology used for this evaluation and the scope of this study present room for revision and expansion. Some of these limitations have been implied throughout this research, yet a more formal discussion of limitations and future research is warranted.

While self-imposed limitations on methodology were necessary to make this study manageable, it is important for the reader to be aware of these limitations in order to adequately evaluate the conclusions which have been drawn. The most obvious limitations have to do with the selection of games for my sample and the evaluation of these games. Simply improving on the sampling method would allow for a more comprehensive picture of the FOSS game scene. Also, different evaluation tools could be used to provide a different perspective on the quality of FOSS games.

Perhaps the single most significant limitation in sampling was the omission of games which lack a single player mode. As seen in Chapter 3, subscription based online games are among the most popular commercial games for PCs. Thus the lack of multi-player only games from my sample significantly narrows the range of conclusions which can be drawn. Related to this limitation in sampling is the lack of an evaluation of multi-player capabilities and performance in FOSS games (and comparison in these capabilities to commercial games).

Stronger conclusions could also be drawn if, in addition to selecting and evaluating FOSS games, a sample of commercial games were selected and evaluated using the same evaluation standards
as the FOSS games. Because of space limitations, while a number of commercial games were mentioned in this study for comparison, they were discussed in significantly less detail than the FOSS games in the sample. The commercial games were merely used to highlight certain aspects of the FOSS games in the sample and to put various components of the FOSS games in perspective. In a future study, there is more than one way in which commercial games could be selected for evaluation. They could be selected based on sales data while applying the same criteria as were used for selecting the FOSS games, or they could be selected in such a manner as to ensure the same genres of commercial games and FOSS games are evaluated. Also, research could be conducted to explore which genres represent the highest degree of technical and/or artistic sophistication in both FOSS games and commercial games (and analyze whether different business models and/or modes of production are more appropriate for certain genres of games).

In addition to the above suggestions for future sampling methods, a future study could also benefit by evaluating a larger sample. In addition, the sample could be drawn from sources other than SourceForge. Considerations such as when each FOSS project was started, how long it had been in development, the level of present and past activity in development for each project, and the number of people involved in the development could also be taken into account when selecting a sample.

The approach used for the evaluation of the sampled FOSS games involved applying a formal analysis, a qualitative evaluation of quality in terms of technical criteria such as graphics, and reporting the ratings which other users gave the games. Taken together, this approach allows for a meaningful assessment of quality to be made. While this approach has the advantage of providing a detailed description, performance of a quantitative study using multiple coders would also be a valid approach for studying the quality of video games.

Many of the games in my sample continue to be actively developed. Furthermore, these games can be said to vary greatly in terms of the current development state. Thus lack of polishing in some of
the games evaluated could simply be because the games are in an early development stage. Thus, future research could include a longitudinal study of various FOSS games to evaluate how they mature over time. Such a study would provide more insight into the development process, provide information on the speed of development, and allow more meaningful conclusions about the level of polishing in FOSS games (as the fixing of bugs and improvements in user interface could be observed). The release of Nexuiz 2.4 after Nexuiz 2.3 had been selected for my sample granted me the opportunity to observe such improvements to a limited extent for one game in my sample, but if the same could be done over a longer period of time for all games, much could be learned.

Thus far the limitations discussed and ideas for future research presented have all been related to sample selection and evaluation of the sample. While changes in sampling and evaluation techniques could provide a different perspective on the state of FOSS games, there are other approaches which could be taken to improve understanding of FOSS games and the utility of copyright law for spurring the creation of high quality video games.

This study has largely been limited to evaluating the product of FOSS game development (i.e. the games themselves). Furthermore, the evaluation of these games was, for the most part, limited to what could be observed by playing the games. Thus, it is worthwhile considering research which could be conducted relating to the process of FOSS game development, and to “under the hood” evaluation of FOSS games.

In Chapter 3, a study of FOSS development practices evaluated by looking at FOSS game development communities by Scacchi (2004) was discussed. This presents a whole different avenue for research. Practices in FOSS game development could be studied through the evaluation of project websites, message boards, and chat rooms. Interviews with developers could also be used. Such a study could be conducted using methodologies common in ethnographic research. This type of research, in addition to providing further insight into FOSS game development practices, could also
shed light on the various motivations for participating in such a project. Furthermore, this type of research, when coupled with the methodology used in my study for evaluating end products, could potentially aid in determining what practices and social dynamics predict success or failure in the development of a high quality FOSS game. This approach may be easier and more practical when applied to FOSS development than when applied to commercial software development, as it seems likely that the entire process of FOSS development would be more transparent than that of commercial software development (during the development process, discussions on progress and distribution of labor may be publicly available on the Internet for all to see, and people participating in the projects are not likely to be bound by non-disclosure agreements). If it is indeed the case that it is easier to study development practices in FOSS communities, however, this would imply that it may be difficult to gather information about commercial development for comparison.

Throughout this thesis, the concept of modularity has been repeatedly mentioned. In order to adequately evaluate modularity in FOSS games, however, a technical evaluation of both source code and resources would likely be necessary. Such an evaluation, however, would likely be difficult if not impossible without extensive technical knowledge. Furthermore, the simple fact that proprietary, closed sourced code is not easily accessible to researchers would present problems for comparing FOSS code to proprietary code.

Other research on the technical side could involve comparing the tools used by FOSS developers to those used by closed source developers. Above, the FOSS level editor GtkRadiant, which was developed by the commercial development company Id Software is mentioned. It would be interesting to figure out to what extent FOSS developers make use of proprietary tools, to what extent closed source developers make use of FOSS tools, what considerations developers take into account when deciding on which tools to use, and what impact this decision has on the outcome of a software project.
Yet another avenue for future research would be further legal and economic analysis. One possibility for a future study would be to evaluate how various licensing terms (for both closed source and open source software) impact the outcome of video game projects. Similarly, a study on outcomes of different business models (ranging from “no profit incentive” to a duel licensing model to pure retail sales of closed source software) impact the quality of video game projects.

While my study has served as an entry point into the exploration of the topic of FOSS video games, it is clear that the topic has barely been scratched. The insights described in this study require future research for validation, and at the same time raise additional questions related to the topic. As technical, economic, social, and legal conditions continue to change, hopefully this topic will be explored and expanded by others. Some questions, however, may only be answered by time despite the best efforts of researchers to predict the future and evaluate hypothetical legal frameworks.
Appendix A: Inside the Development of a Simple Game

The purpose of this appendix is to give the reader an idea of what is involved in developing a video game. This will be presented as a description of the development process of a simple “side-scroller” I developed over the summer of 2005 which is titled “D-Command Escape”. Although this game is indeed simple compared to popular modern games (both commercial and FOSS), the source code alone, which only represents a portion of work involved in developing this game, amounted to forty single space printed pages in a size 10 font. Furthermore, in developing this game, tools and libraries developed by others (both FOSS and proprietary) were used which represent an extremely large quantity of labor. It should also be noted that, as this game was not produced with the intention of wide distribution, attention was not paid to obtaining permission to use various artistic content produced by others. Thus, all of the music, many of the graphics, and one of the sound effects, were taken from the Internet and used in the game without the permission of the copyright holders. Furthermore, the final game lacked the polishing which would be expected of a commercial game. The amount of source code required for polishing would have taken a significant amount of time to write; and as the game was not meant to be distributed (except to friends for bragging rights), it simply did not seem worth the extra effort. The reader should be under no illusions that either the development methods or code architecture of this game represent the methods and code architecture seen in professional software engineering.

As the focus of this thesis is an evaluation of the viability of FOSS game and not computer science, this appendix will, to the extent possible, avoid the use of highly technical language and rather break the process of development down into smaller components.

The very first step in developing this game was deciding the genre and concept. I decided I wanted to make a “side scrolling” game (in which the on-screen area can scroll on both an x-axis and y-axis in a larger 2-dimensional level). In the broadest terms, the general format would be moving
through 2-dimensional worlds, jumping from platform to platform, collecting items, shooting enemies, and avoiding contact with enemies. While this describes the general style of game play, it provides little detail on the feel and concept of the game. Thus a background narrative was needed in order to provide a context for the game world. At the time I was developing this game, I was extremely interested in anime (Japanese animation). Thus I decided to base the background narrative on an anime series called “Kiddy Grade”. In order to make the game “work” as intended, however, it was necessary to focus on a specific portion of the plot in the anime and deviate slightly from the original plot. At this point it should be noted that the only role of narrative in this game was to provide a context for game play; the plot was not further developed within the game as this would have added a large degree of complexity to the development process. Below is a description of the narrative taken from the “readme.txt” file included with the final game (in addition to providing the narrative this excerpt also provides limited information on the mechanics of game play) (misspellings in original document):

In this game you are a female cyborg named Lumiera, a former member of the GOTT (Galactic organization of Trade and Terrifs), however, you recently disobeyed orders by protecting a large group of protesters from being killed by nobles when the protesters were trying to socialize institutions on their planet. The protesters succeeded in their revolution thanks to you and your partner. However, the galactic elites are after you and want to kill you. The elites have used a weapon on you which initializes a "D-Command", which forces the nanotech inside your body to expand, thus causing organ damage, and is intended to kill you. You have the power, however, to telepathically interface with computers. In an attempt to escape death from the D-Command, you interface telepathically with the computer issuing the command, and are taken into a virtual reality where robots are trying to kill you as you collect encryption key parts to turn off the D-Command. To save yourself and your partner you must collect all the encryption key parts on all the levels. Control the player with the left and right arrow keys, the space bar to jump, and the left control key to fire your weapon (which can kill the robots). After you collect all the key parts on a level, proceed to the next level by going through the level exit. There are a total of 10 levels.

Before writing any code, numerous decisions had to be made. What resolution would the game be rendered in (for simplicity, the decision was made to only allow the game to be played in a single resolution as adjusting the resolution would have involved scaling graphics, and thus would have added
extra math and complexity to the development)? What would the various components of game play be (in terms of objects, sounds, and graphics)? How would graphics be organized on the screen? How would animation of various components work? What would be the dimensions of the tiles (maps were stored as a set of “tiles” which represent different portions of the game environment)? How would the files containing tile maps be formatted? Should these files contain information about the background images and sounds for each level? What is the broad flow of the game?

The decision was made to use a resolution of 640x480 and a color depth of 32 Bits Per Pixel (BPP) for the game. The resolution of 640x480 was chosen for several reasons. First, nearly all monitors can support this resolution. Second, it is a standard resolution and has a 4:3 aspect ratio. Finally, it is high enough to look fairly good (considering the game is a 2-dimensional side-scroller), but not so high as to put unneeded strain on hardware. The decision to use a 32 BPP mode for the game was made because most modern machines are set to this color depth in Windows, because it is an easy mode to work with, and because it offers a great deal of color depth. In the final version of the game, only windowed game play (as opposed full screen) for two reasons: First, an unknown bug, likely having to do with the rendering library, prevented the game from rendering properly in full screen mode; also, on wide screen monitors, the game would appear stretched horizontally if it were rendered in full screen mode (the initial plan was to have one binary which rendered in a windowed mode and another in full screen mode, but because of the rendering glitch in full screen mode, only a windowed executable was included in the final version).

Here is a list of objects which can be found in game play:

The player

Enemy robots

Two separate types of solid tiles through which neither the player or her telepathic “shot” can pass through. The only difference between these two types of solid tiles is appearance; one of them is opaque and looks like a brick wall, the other has transparencies (so the background can partially be
seen through it), and has a cartoon-like look).

Level exit (which the player can only pass through after collecting all “encryption key parts” on a level)

Players' psychic weapon

Encryption key parts

Here is a list of the graphics which are included in the game:

Opening screen graphics

Win game screen

Lose game screen

Ten background images (one for each level)

Four separate images for the player (one of player facing left with feet together, one of player facing left with feet apart, one of player facing right with feet together, one of player facing right with feet apart).

Four separate images for the enemy robots (as with player, they are of the robot facing left, right, with feet apart, and feet together).

One image for each of the two tile types

Two images for the player's psychic weapon (to allow animation as it flies through the air away from the player).

Image of the encryption key parts

Image of the level exit

Here is a list sound effects included in the game:

Player jump sound

Player die sound

Sound for when the player picks up an encryption key part

Sound for when player fires her psychic shot

Shot for when a robot dies
The following songs are included in the game:

Intro song
Lose game song
Win game song

One song for each of the ten levels

Before writing code, the layout of the screen was also decided upon. This layout necessarily relates to how the screen is rendered. For each level, there would be a static background which does not move. During rendering of a frame, the background would be the first thing drawn. The tile map was to be rendered on to a large “surface” as soon as a level loaded. This “surface”, before having the tile map rendered onto it, was to be transparent so the background could be seen through it (transparency is accomplished by using a specific color which the computer is instructed to see as transparent and thus not be drawn when copied to the screen). During rendering of each frame, a 640x480 pixel portion of the surface containing the tile map was to be drawn to the screen, the portion to be drawn to the screen is determined by the position of the player within the two dimensional world. After this, the bitmap of the player (taking into account her “state”, i.e. Facing left or right, or with her legs together or apart) was to be drawn in the middle of the screen; and any encryption key parts or robots which are within the on-screen space were also to be drawn at this point.

The decision was then made to have the size of the tiles be 40x40 pixels. There were several reasons for this decision. First, 40 divides evenly into both 640 and 480, thus keeping the math relatively simple. Second, this size seemed about right for the purpose of allowing an optimal portion of the map to be seen at any given time. Finally, if the tiles had been smaller, it would have made designing maps large enough to allow a large amount of scrolling on both the x-axis and y-axis. Key parts were also designed as 40x40 images. The player, the robots, and the level exit were assigned the dimensions of 40x80; thus all three were the same width of a single tile but twice the height.
Another decision which had to be made before further progress was possible was how information about each level would be stored in a level file. At the beginning of each map file, there are two 32-bit integers; one representing the horizontal size of the map (in tiles), the other representing the vertical size of the map (in tiles). Following these two integers is a large string of integers; each of these integers represented some element of the map. For example, a zero would translate into an empty square (neither of the solid tiles). A different value was used for each of the two types of tiles; and yet other values for the starting locations of the player, the robots, the exit, and the key parts. Rather than including the file names of the background images and the level song for each level within the map file, it was decided that these files would be specified within the main game code.

After these decisions, the decision of how to handle animation within the game had to be made. The decision was made that the player's psychic shot, while in the air, would rapidly alternate between two images; thus giving it a “shimmering” appearance. The robots and player would face either left or right depending on the direction they were moving, and as they moved, their legs would alternate between being apart and together.

Yet another decision had to be made about the large scale flow of the game. Thus, it was decided the game would start in a title/intro screen and remain in this mode until the player pressed the 'S' (for “Start”) key. Next, the first level would load. If the player collected all the encryption key parts on the level and reached the level exit, the next level would be loaded. On the other hand, if the player died, the game would jump to the lose game screen (which would continue to be displayed until the player pressed the escape key which would end the program). If the player pressed the escape key within the level, the game would immediately free all allocated memory, close and exit to windows. If the player completed all ten levels, the game would proceed to the win game screen, which would be displayed until the player pressed the escape key in order to quit to Windows.

After making these decisions, graphics, sound effects, and music had to be selected and created.
The graphics for the intro screen, win game screen, and lose game screen, were taken from an anime fan site, resized to 640x480 resolutions, and edited to display information (in the intro screen, the name of the game, the name of the pseudonym of the author (me), and text telling the player to press 'S' in order to begin playing was added; appropriate text was also added to the win game and lose game images). The tiles, key parts, level exit, and robots were drawn from scratch using Windows Paint. The graphics for the player were initially taken from an anime fan site; but heavily edited in Windows Paint in order to have the correct size, transparencies, and stances (left, right, legs apart, legs together).

As the game had an anime theme, all thirteen songs selected (and downloaded from the Internet) fell into the genre of “J-Pop” (Japanese pop music). The sound effects were created using Windows Sound Recorder. Most of the sound effects were simply created using my voice then adding effects such as echo; and speeding the playback up to alter the pitch and make the sounds not sound like a human voice. The sound effect for picking up a key part, however, was made playing a ring tone from my cell phone into the microphone. In addition to designing these graphics, it was necessarily to make another set of graphics to be used in the level editor (a separate program which had to be developed and which will be discussed later). The graphics for the level editor, however, were comparatively quite simple and appeared as icons rather than detailed graphics, and were easily produced with Windows Paint.

After all this, yet another decision had to be made before any work could begin on writing the source code. What tools and libraries would be used for the game? The decision was made to use the FOSS library Allegro for rendering graphics, handling keyboard input, and managing the timer (the timer helps guarantee that the game will run at the same speed on any system). As the Allegro library does not support .mp3 playback, a different library had to be used for playback of both sound and music (Allegro couldn't be used for sound effects as it would interfere with the settings in the other library). Thus the FMOD library was chosen for playback of both sound and music (this is a proprietary library, but the developers allow it to be used for free for non-commercial purposes). The
decision was made to write the level editor using a language and Integrated Development Environment (IDE) called Ibasic (this is a language closely related to the Basic language, and was chosen because it seemed easier to use for development of GUI applications (Graphical User Interface applications with standard menus and the like) than C/C++. The game itself would be written in C/C++ using an IDE called DevC++; which is a FOSS development environment which works with a FOSS C/C++ compiler.

Next, the “rules” of the game had to be figured out, as these rules would dictate how levels had to be designed in order to be playable. This included determining how the enemy robots would move (it was decided that robots would walk to the end of the platform they were standing on until they either reached the end of the platform or reached a wall, then turn around and walk in the opposite direction). How high the player could jump also had to be decided. It was determined that the player should be able to jump up to a height of 2.5 tiles, or 100 pixels; during a jump, the player would continue to move upward at a constant rate until either the player's head hit a solid tile or the maximum height had been reached; then move downward at a constant speed until the ground was hit. Constant speed during jumps was used as accelerated motion would have required more calculations, and would have required the use of floating point math; and I only wanted to use integer math in the game. Furthermore, the player would have the ability to move either left or right in the air during a jump or a fall off the edge of a platform. It was also decided that the shots fired by the player could only travel a finite distance. Once the player fired a shot, the shot would begin moving in the direction the player was facing (it would appear to come out of the player's head). The shot would move faster than either the player or the robots could move (player and robots would move at the same rate). The shot would continue moving until: it reached the maximum range, it hit a wall, or it hit an enemy robot; if any of these three things happened, the shot would vanish. Furthermore, only one shot would be allowed to exist at a time; a player would not be able to fire another shot until the previous shot had vanished.
because of any one of the three reasons mentioned above.

The next step was the creation of a level editor, which could be used to design levels for the game. This level editor needed to be able to allow the user to place tiles, robots, the player, the exit, and key parts on a large 2-dimensional map. Furthermore, it was required to allow the user to set both the horizontal and vertical size of the map in tiles. Finally, it had to be able to load maps from the hard drive for further editing and also be able to save the completed map to the hard drive. Despite a fair degree of complexity in this tool; the final product was not well polished, intuitive, or user friendly. However, it did not have to be, as my intention was simple to develop a tool which I alone could use in the development of the game. After completion of the level editor, the actual levels for the game were designed.

After all these decisions and design tasks were completed, the code for the actual game had to be written. Describing every step of this process is beyond the scope of this appendix; but at least a brief discussion is warranted. Before writing code that would actually “do” anything, data structures had to be created in order to organize information in the game. These data structures were all nested into a master structure called “everything”. Within the “everything” structure, information about the current state of the game (i.e. Intro screen, win screen, lose screen, or in a level) could be stored. In addition to this, the everything structure contained elements in which graphics, music, sound effects, tile maps, information about the level files, information about the sound and music files, and information about the positions of the player, the key parts, the robots, and the exit for the level would be stored.

After the data structures were created, code had to be written to initialize the libraries, configure the timer, and load sound, music, and graphics into the “everything” structure. After this, specialized code had to be written in order to open map files and load the information contained within them into the “everything” structure.
Broad code was required to lay out the flow of the game, starting with the intro screen and proceeding through the game to an ending of either winning, losing, or quitting by pressing the escape key. After writing this broad flow control code, the core logic of the game had to be written. This involved a loop which went through one iteration for each frame. This loop was responsible for processing user input, looking for collisions between the player, the robots, their key parts, tiles, the player's shot, and the level exit and respond appropriately. Also, within the loop the position and animation frame of the player, robots, and shot had to be coded. After this game logic was completed, the next step within the loop was to render the frame. Rendering involved rapidly copying images and the tile map surface to the screen; and involved a significant amount of logic and math to achieve.

Although the previous several paragraphs don't come close to touching every component of the code, they do provide an idea of the complexities involved in programming a game.

After writing code, debugging was required. Minor glitches in game play logic had to be fixed, and one level had to be partially redesigned as it was possible for the player to get stuck in a part of the level where she could neither complete the level nor die. While many bugs were successfully fixed, even after debugging some issues remained. However, at this point the game was in a very playable state, and was a work I could take pride in.

The description of the development process given above sheds light on how much work is required to develop even a simple game. Modern commercial games and sophisticated FOSS games, however, are exponentially more complex. This description of the process, however, puts the discussions of previous chapters into context.
Appendix B: Screenshots

While a description of the graphics and gameplay of the various games was provided in the results chapter, but as the expression says, a picture is worth a thousand words. This appendix contains screenshots from all games in my sample along with screenshots from some select commercial games for comparison.

*Figure 2 Nexuiz 2.4*

Source: [http://www.alientrap.org/nexuiz/screenshots/nexuiz_screenshot_06.jpg](http://www.alientrap.org/nexuiz/screenshots/nexuiz_screenshot_06.jpg)
Figure 3 Nexuiz 2.4 (second screenshot)

Source: http://www.alientrap.org/nexuiz/screenshots/nexuiz_screenshot_10.jpg
Figure 4 Sauerbraten (December 2007 Version)

Source: http://sauerbraten.sourceforge.net/screenshots.html
Figure 5 Sauerbraten (December 2007 Version)

Source: http://sauerbraten.sourceforge.net/screenshots.html
Figure 6 Stepmania 4

Source: http://upload.wikimedia.org/wikipedia/commons/1/1c/StepMania_4_Gameplay.png
Figure 7 Battle for Wesnoth 1.4

Source: [http://www.wesnoth.org/images/shots/wesnoth-1.4-8.jpg](http://www.wesnoth.org/images/shots/wesnoth-1.4-8.jpg)
Figure 8 Scorched 3d 41.3

Source: http://www.happypenguin.org/images/island-2.jpg
Figure 9 Armagetron Advanced 3.0

Source: http://linux.softpedia.com/progScreenshots/Armagetron-Screenshot-3923.html
Figure 10 Freeciv 2.1.3

Figure 11 Secret Maryo Chronicles 1.4

Source: http://www.happypenguin.org/images/maryo02.jpg
Figure 12 Civilization IV

Source: http://cache.kotaku.com/gaming/images/Civ4ScreenShot0032.jpg
Figure 13 Super Mario Bros 3

Figure 14 Crysis

Figure 15 Unreal Tournament III

Source: http://www.someonesimages.com/temp/ut3_cap.jpg
Figure 16 Unreal Tournament 2004

References


