Games for Research: A Comparative Study of Open Source Game Projects

Stig Magnus Halvorsen[1,2,3] and Kjetil Raaen[1,2,3]

[1] NITH, Norway
[2] Simula Research Laboratory, Norway
[3] Department of Informatics, University of Oslo, Norway

Abstract. Video games have proved to be an interesting platform for computer scientists as games demand the latest technology, fast response times and effective utilization of hardware. Finding the right games to perform experiments are however difficult. Some researchers create their own smaller prototype games to test their ideas, without performing tests in larger scale productions, which decreases the practical applicability of the conclusion. An important reason is the lack of suitable games for research. This paper proposes a list of qualities and features required by researchers for a video game to be suitable for computer science research. Further, it evaluates four games with open source code and discuss their usefulness. We also consider the current state of open source games and possibilities for enhanced cooperation between the professional and research communities.

1 Introduction

Decades have passed by since the first commercially produced and published video game was released, giving birth to the video game industry. Since then, both the popularity and demand for video games has exploded with an insatiable audience demanding more content and more technically advanced features for every major release. This has resulted in one of the world's toughest industries with high development costs and high risks for failure. The industry is thus a technology promoter that keeps pushing hardware and software to the limit in order to satisfy the ever demanding market.

This demand has since the late nineteen-nineties caught the eyes of computer scientists. Computer games have been a tool for research (such as Petlund et al. [11], Raaen et al. [12]), as well a topic of research (such as Claypool & Claypool [3], Eisert [10], Waveren [17]). There are, however, some crucial differences between commercial game developers and researchers. Game developers prioritize rapid creation of content and implementation of a large system, while the computer scientists prioritize investigating and researching individual components of the larger system. Scientists do not have the resources to develop a complete industry standard video game, while game developers rarely publish any results of their research. This indicates that cooperation between the industry and the researchers is beneficial.

Commercial games are rarely available in source code form. New, the code is considered company secret while releasing older code is either ignored or can be too costly. Developers are sometimes willing to co-operate with researchers, but will often limit

D. an Mey et al. (Eds.): Euro-Par 2013 Workshops, LNCS 8374, pp. 353–362, 2014.
© Springer-Verlag Berlin Heidelberg 2014

how much of the inner workings can be disclosed. This makes the research cumbersome as well as difficult to reproduce, potentially decreasing the quality of the research material.

The current situation makes it hard for computer scientists to get hold of industry standard systems, resulting in researchers creating their own small prototype games specialized for the use of their current research. Examples include White et al. [18] and [12], which both try different concepts in simplified environments. This research should be implemented and tested in complete massive multiplayer games in order to achieve more realistic results.

Other researchers follow the same approach. Ideas need to be embedded in a full scale production in order to get fully realistic results. There is thus a need for a universally available industry standard video game that can be freely used for scientific purposes.

This paper seeks to collect, evaluate and structure scientists' requirements of video games to use in their research. These criteria have been used to evaluate and compare a selection of open source computer games in order to investigate their suitability for scientific work.

2 Method

The primary approach in this work is analyzing gameplay, source code and documentation of open source video games and game engines. Features and qualities found in each game were compared with requirements collected from a online survey.

The authors selected 60 different features both common and uncommon, that were to be evaluated as either *unimportant*, *research requirement*, *engaging requirement*, or *both*. Survey participants were also invited to provide additional requirements. An "engaging requirement", is a feature that is needed for a game to be engaging for potential test subjects. The features were meant to cover most known technical features that are currently realistic to implement in games. Both the basics, such as user input through well-known devices, and less common features, such as voice control are covered.

The survey concerns software, and assumes that the necessary hardware and drivers are in place and working properly for the various features to be fully functional.

The survey was active for approximately a month. We sent invitations to computer scientists that had or were interested in using video games in their research. Most of these researchers were based in Norway among them some scientists from "Simula Research Laboratory" and "SINTEF", but it was also sent to various known researchers abroad. The survey had a total of 22 participants, which is a number that reflects the fact that there were few computer scientists which replied that they felt their experience and work was relevant for the survey.

Participants from the game industry were not invited, because the problem discussed in this paper is unique to academic research.

Games were selected from the list of open source games at Wikipedia [20], from previous research, as well as from the authors' own experience. This approach should give a reasonable overview of which such games are available. Games from Wikipedia's list with updates prior to 2012 are considered inactive and therefore excluded. The games

we examined more thoroughly are, *Doom 3 (FPS)*, *OpenArena (FPS)*, *Vega Strike (Action Simulator)* and *PlaneShift (MMO)*.

The first phase of the data gathering process involved installing and playing the respective games. It is a quick and an effective approach to get a great overview of each game, indirectly providing information on various game mechanics, its quality and whether or not it is a stable product. Graphics quality is evaluated visually while playing the game. This gives an immediate indication on the quality of the graphics, and whether more advanced graphical features, such as *shaders*, are supported or not. The approach also reveals implemented user interface mechanisms, such as game menus, buttons, in-game chat and similar.

Some information on embedded physics were retrieved by interacting with the games in ways that might not have been considered by developers. This involves jumping off various ledges, firing weapons at objects not usually considered targets, and experimenting with various game mechanics. Such gameplay reveals much about interaction between objects in the game. Important information is also retrieved by looking at various menus and game settings. Setting up a local server will for instance let you know how many interacting concurrent players the server supports.

The second phase was to retrieve information by collecting data from official websites and documentation available for the games and engines. Primarily to find technical specifications revealing features either implemented in the games, or "hidden" engine features not used by the games. It can alone reveal a significant amount of features and qualities not visible by simply playing the game. It is also more efficient than trying to interpret features hidden within the source code. The amount and quality of the documentation is highly variable. Because open source games require extensive cooperation between geographically distributed developers, much more documentation is available than for commercially developed games.

The final phase is to investigate the undocumented features by studying parts of the source code. This requires obtaining a copy of the source code from the official version control repository or file server, and to open the files in the intended development environment, if any. Some will immediately compile, link and run successfully, while others require the retrieval of game data or additional resources. For games with commercially licensed data, researchers need to purchase or in some way legally obtain these files. These are not needed to analyze the source code, but can make it easier by allowing you to navigate the code through testing and debugging.

The next step is to get an overview of the structure of the code to find out where to look for various potential features. Analyzing the code reveals code standards, features, and the code quality. It can be a time consuming task, we were only interested in examining the code for specific functionality and overall quality, which does not require examining every line of code manually. Once done, a final in-depth search for the unknown features are done, involving basic file search and reading. Some features are easier to determine than others, due to their complexity and nature. Features are marked as they are found or declared missing.

3 Results

The research process of collecting a list the most important features for a computer game resulted in a surprising engagement from some of the researchers invited to participate. A total of 22 researchers participated in the online survey. We received various valuable and interesting feedback on both the contents of the online survey and the research topic itself. The most important was regarding the content of the quantitative survey, which was by some perceived as too limited and that a larger of scale qualitative interviews would have been better.

Table 1. Evaluated features & games

Feature	**Requirement %**	**Unimportant %**	**Doom 3** (idTech 4)	**Open Arena** (ioquake 3)	**Vega Strike** (Vega Strike)	**PlaneShift** (Crystal Space)
Performance	91	5				
Logging system	77	18	✓	~	✓	✓
Latency intolerant	68	18	✓	✓	✗	✓
Simple Physics	68	23	✓	✓	✓	✓
3D Support	68	14	✓	✓	✓	✓
Technical Documentation	68	5				
Simple AI	64	27	✓	✓	✓	✓
2D Support	63	14	✓	✓	✓	✓
Adaptability	59	23				
Scripted AI	59	23	✓	✓	✓	✓
Good Code Quality	59	18	✓	✓	✗	✓
Computational Heavy AI	55	32	✓	✓	✓	✓
Reliability	54	27	✓	✓	✗	✗
Player vs Environment	54	23	✓	✓	✓	✓
Large networked (13-64)	54	36	✗	✗	✗	✓
Packet loss intolerant (TCP)	50	32	✗	✗	✗	✗
Single Player Mode	50	45	✓	✓	✓	✗
Player vs Player (PvP)	50	27	✓	✓	~	✓
Massive networked (65+)	50	41	✗	✗	✗	✓
	✓ implemented		~ partially		✗ absent	

The remainder of this section contains an in-depth analysis of the selected games, examining them for the quality of the features we found are important in the survey. Results from the survey are presented in table 1, which shows the most requested features. Qualitative features such as performance are not evaluated in the table, but described further in the section for the individual game.

3.1 Doom 3 (IdTech 4)

Doom 3 including the *idTech 4* engine is to the knowledge of the authors the most complex and the most advanced both technically and visually open source game available as of this date. Primary development on the engine was completed before the game's

commercial release in 2004 [8]. The source code was published by Timothee Besset (user "TTimo") on *GitHub* in November 2011 [2].

However only the source code has been made freely available, and not the data files. The data files can be obtained by purchasing the game and configuring the compiled game to read these. However, you will not be able to publish new content or updates with the original game data as this is only available under commercial license [1]. This can be used for research, but the complete experimental setup can not be shared.

The engine is mainly written in C++, using OpenGL for 3D graphics, and contains 601032 lines of code [13]. The source code defines various shared libraries as well as an executable.

The source code is of high quality with good naming conventions, some good comments and it is relatively easy to read. It contains clear traces of optimization, which is expected as it is a major production that sold more than 3.5 million copies [15]. This is also an indication of stability, as it has been under industry standard tests and been played by millions. Id Software have also implemented their own memory manager, or *garbage collector*, for faster dynamic memory allocation.

The code contains some less common hacks, where the most notable one overrides all *protected* and *private* keywords setting them to *public*. This is required by their custom run-time type information (RTTI) functionality.

Doom 3 is primarily a single player FPS game, with a online multiplayer mode supporting up to 8[1] players in four different *Deathmatch* modes, allowing free for all fights. However, one of the programmers, John Carmack, has stated that the engine can support more than 4 players but that they decided to restrict it for game design reasons [7]. This can naturally be modified as the source code is freely available. The game is an intense action FPS experience, implying the need for low response times as suggested by [3]. This is reflected in the code with built in latency-hiding mechanisms. Trading reliability for response time, the latency sensitive section of the network code uses the UDP protocol.

A natural part of the single player functionality is the need for "savegames", which is fully supported by the engine. The game was released before the concept of *Cloud storage* went public, so it is not a part of the engine itself but can be obtained by running the game through *Steam*, although not open source.

The original project found on *GitHub* does currently only build on Windows using *Microsoft Visual Studio 2010* or later, and on OSX using *Xcode 3.2* or later. It is using operating system (OS) specific libraries for various low level operations including threading and network. It makes it somewhat cumbersome to port to other platforms, but ports are already available. Some with their own open-source *Git* repositories (Besset [1] and Sanglard [13]). Threads have been utilized for time-critical functions that should not be limited to the frame rate of the game, including sound mixing.

Adapting the engine or game itself to be used for other game types than FPS is possible as it is open source. The engine has, however, primarily been used by FPS games [19], so it is likely that it will require a significant amount of work to modify it into other genres.

[1] Original 4 players, the support for 8 players were added in the "Resurrection of Evil" expansion.

There is unfortunately little or no official technical documentation of the engine apart from the source code. An idTech developer published a engine modification guide which is useful, though not totally complete. There is also a very good code/engine review available by Fabien Sanglard [13] but it also lacks some information, including networking. The only thing available close to design documentation is the game manual, but no official documents have been published.

Other available and relatively industry standard engine features are logging, physics, support for scripting, artificial intelligence (AI) and a library for graphical user interface (GUI) components. See table 1 on page 356 for an overview of the most important implemented features.

3.2 OpenArena (Ioquake 3)

OpenArena is a pure first-person shooter developed using *ioquake3*, an open source community developed enhancement of the *idTech 3* engine. idTech 3's source code was released as open source together with the game *Quake III Arena*'s source under the "GNU General Public License" in 2005 [14]. The binary data files generating the content of the game are as with Doom 3 still closed and copyrighted. OpenArena's intention is to provide a free and enhanced alternative to *Quake III Arena*. *Quake III Arena* is, as Doom 3 a commercial computer game, implying a polished and stable game. The *ioquake3* project claims to have fixed many known bugs in the original engine, patched some security issues and that they have enhanced the overall stability of the entire engine and gameplay.

The engine and the game is primarily developed using C, using OpenGL for rendering 3D graphics. The original *idTech 3* engine consists of 367815 lines of source code [13], which is almost half the size of *idTech 4*. The idTech engine and OpenArena are both well recognized and previously used in computer scientific research (such as [17], [10], and Parry [9]). Threads are not utilized to balance the load in *idTech 3* which can create uncommon, yet possible blocking issues in the games. *ioquake3* has fixed this by utilizing threads through the Simple Directmedia Library (SDL) 1.2 (SDL_thread), and by using the platform independent *OpenAL* library for sound. The engine also supports video playback, which is also true for the unmodified version.

OpenArena's gameplay is as its predecessor, Quake III Arena, primarily focused on the online multiplayer experience. It also supports a simple single player game mode, which behaves as a multiplayer game where the opponents' are pre-programmed AIs simulating players and the local machine runs the server. The bots are programmed in C, using a well structured collection of functions for in-game artificial intelligence. OpenArena natively supports up to 12 simultaneous players in one match. Single player progress, achievements and settings are saved on the local hard drive. Multiplayer saves primarily the settings, such as user name and character.

The code quality is in general well structured and documented, through naming conventions, comments and file directory structure. There is a log system available, and an in-game command window with various options. Technical documentation is found on OpenArena's and ioquake3's official websites and *Wikimedia* based encyclopedias.

The game compiles and runs on Windows, OSX and most Linux based platforms and it thus qualifies as easily portable. It is one of the enhancements made by the *ioquake3*

project, replacing platform specific libraries with cross-platform supported libraries. Both the idTech 3 and ioquake3 engines are primarily FPS engines, somewhat limiting the adaptability of the engine. It is possible to modify it to support other game genres, but probably not without changing engine specific features.

3.3 PlaneShift (Crystal Space)

PlaneShift is an open source community developed multiplayer online role playing game (RPG) developed using the open source *Crystal Space* engine. All code and the engine is released under the "GNU General Public License" (GPL), while the data content files are copyrighted to "Atomic Blue Non Profit Corporation" and distributed under the "PlaneShift Content License". The license details can be found on their official website [6]. There is only one game mode, multiplayer, where you are able to interact and fight with other players or non-player characters (NPC). Each player needs an individual game account, stored in the game's database, together with character progress and other game data. This gives a "cloud like" storage solution with a globally available database, where you can access your game data from any computer. The official database is however currently only one server, where your data may be erased at any given time. This is because the game is still currently under development, even with most of the key functionality is in place [5]. The game can therefore not be categorized as completely stable, and optimized. Unexpected behavior and bugs do exist as we experienced while playing, but it never crashed. It is however of surprisingly high quality, and will be an interesting candidate when finally released as stable.

Crystal Space, the game's engine, is an highly portable and acknowledged open source software development kit (SDK) for developing 3D applications. It is released under the "GNU Lesser General Public License" and is developed using C++. Various games have been created with the SDK, and it can be classified as highly adaptable. The SDK includes all functionality required by a game, including physics, collision detection, graphics, user input, GUI, and more. It is also possible to find or develop plugins for special needs [16].

The game itself is a more specialized piece of software, serving its purpose as a multiplayer RPG. It is probably easier to adapt than the previous evaluated games, due to the nature of the engine and that the engine provides development tools to make content development easier. PlaneShift has a menu system, a 2D GUI system, some physics and supports joystick as an alternate input device.

3.4 Other

The other games were rejected for various reasons. Most as they are perceived as too old, including graphical rendering techniques which are obsolete for the project to represent a modern, near state of the art video game.

Vega Strike, an Action Spaceship Simulator, was evaluated but rejected as it turned out to be an incomplete game without any official support for online multiplayer gameplay. The game's graphics is also primarily 2D textures with few 3D models, and is thus not considered as a game representative of the industry standards. Other unfinished open source games are *0 A.D.*, *Ancient Beast*, *Chaotic Rage*, *Dungeon Crawl*

Soup, *Flare*, *FreeCol*, *FreeOrion*, *Hedgewars*, *Minetest*, *SpaceZero*, *Teeworlds*, *X-moto*, *SuperTuxKart*, *Sintel The Game*, *Rigs of Rods*, *Unknown Horizons*, and *Unvanquished*.

Other quite popular games such as *BZFlag*([11]) a *Tank FPS*, and *QuakeWorld* (Cordeiro et al. [4]) have been rejected even though they have been previously used for computer science. This is primarily because the games origins back to the late 90ies [20] and are not good enough, even though they are recently updated.

This also includes games as *The Battle for Wesnoth*, *Battle City*, *Flight Gear*, *FreeCiv*, *Open Hexagram*, *PokerTH*, *StepMania*, *Chocolate Doom*, *Tales of Maj'eyal*, *Advanced Strategic Command*, *Angband*, *Biniax*, *OpenRA*, *Zero-K*, *Crossfire*, *M.A.X.R.*, *TripleA*, *UFO: Alien Invasion*, *Widelands*, *C-Dogs*, *Katana Shoujo*, *Mari0*, *CorsixTH*, *Exult*, *Freesynd*, *Gigolomania* and *NX Engine*.

Other games were also rejected as they were too similar, if not on the same engine, as earlier selected and evaluated games. This includes games as *Oolite*, *Red Eclipse*, *Doom 3 BFG Edition*, *Xonotic*, *Cube 2: Sauerbraten*, *Smokin' Guns* and *Warsow*.

4 Discussion

Our results show that no single game is perfect for all scientists. We have evaluated the features of some open source games on a individual project basis. Some projects are so distinctive that their requirements will rarely be similar to the general perception of important features. The feature evaluation scheme can also be used to evaluate the qualities of other computer games for use in research. We find *Doom 3* and *OpenArena* the most useful games for research within the field of computer science. Both are initially made of commercial projects. *PlaneShift* can be used if the experimental state of the game is acceptable.

The quality of the open source games indicates that they are far behind the technology pushing commercially produced industry standard games. This is confirmed by *Doom 3* as it was released in 2004, and is still among the more polished and technically advanced open source games available. Commercial games and engines made open source, like *Doom 3* with the *idTech 4* engine, has poor documentation compared to open source games. Open source games and commercially produced games might be considered as incomparable due to the usual great difference of dedicated resources for the different project types.

Doom 3 reveal some interesting facts regarding the complexity when compared to the older releases. It has a significant amount of more code than the other games, and the project structure is in many ways more cluttered with functionality spread across a heap of files. It is not a surprise as it is a vast game compared to the other, but it makes the code harder to understand and work with. This is probably also true for many new and even larger productions and should be considered when planning a research project.

5 Conclusions

This paper has gathered a list of features and qualities required of a video game suitable for use in computer science research, and has evaluated the suitability of games released with open source code. Our results shows that there are no ideal games that can be used

for all research purposes. We have, however, developed a list of features and qualities that are worth considering when looking for attractive games, and found two computer games that are potential candidates.

The two highest rated candidates, *Doom 3* and *Open Arena*, are both made from or based on commercial produced computer games with the source code made publicly available. Among them is *Doom 3* which was released in 2004, the latest, most polished and professional production. It is negative as it is an almost ten year old release, and still one of the best open source alternatives available. This is an indication of the open source game industry's current state, which is way behind the commercial industry. It can be discussed whether or not open source and commercial projects are comparable due to the vast difference in available and dedicated resources.

The rejected games have been flagged as rejected because they are either incomplete and unstable, or no longer can be described as state of the art games; Games that are not comparable to new commercial released games. Some will consider *Doom 3* outdated as well, but it is currently one of the most modern open source alternatives. Eventually this game will also be obsolete. We hope better open source alternatives will emerge.

5.1 Limitations

The informal procedure for gathering the list of candidate games creates a risk that we have missed some potential games. Due to the vast amount of code in each of these projects, we might also have overlooked some undocumented features despite our thorough search.

5.2 Future Research

Our findings should be used to verify previous research using large scale games, where the research have purely been tested with smaller prototypes. The games evaluated as suitable may be used directly to verify the research credibility, and the feature list may be used to evaluate other open source games.

For a more generally useful result, similar work should be performed at a more global level, contacting computer scientists at research institutions all over the world. Further, qualitative work such as interviews with scientists experienced in using games for technical research or as topic for research could give more detailed results regarding requirements. There are other open source games available, and more should be evaluated to obtain a greater list of suitable games. Future reevaluations are also required as games of the current generation games will become outdated and new options will arrive.

Ideally we would like to see more game studios releasing their games as open source. Researchers and industry could achieve mutual benefit from more cooperation on new technologies. Releasing games as open source will require some time and effort from the companies, and is thus usually not perceived as a good investment. Tighter cooperation will alleviate this situation. Researchers could also co-operate with open source projects to achieve a similar goal, but it will likely require more work on the part of the researchers.

References

[1] Besset, T.: Doom 3 repository on github (November 2011), https://github.com/TTimo/doom3.gpl

[2] Besset, T.: Ttimo/doom3.gpl/commit history (November 2011), https://github.com/TTimo/doom3.gpl/commits/master?page=2

[3] Claypool, M., Claypool, K.: Latency Can Kill: Precision and Deadline in Online Games. In: 1st ACM Multimedia Systems Conference (2010)

[4] Cordeiro, D., Goldman, A., da Silva, D.: Load balancing on an interactive multiplayer game server. In: Kermarrec, A.-M., Bougé, L., Priol, T. (eds.) Euro-Par 2007. LNCS, vol. 4641, pp. 184–194. Springer, Heidelberg (2007)

[5] Atomic Blue Corporation. About planeshift (April 2013), http://www.planeshift.it/about.html

[6] Atomic Blue Corporation. Planeshift license (April 2013), http://www.planeshift.it/license.html

[7] GameFront. Game info: Doom 3 | multi player | overview (May 2013), http://doom3.filefront.com/info/Multiplayer

[8] GameSpot. Doom 3 tech info (May 2013), http://www.gamespot.com/doom-3/techinfo/platform/pc/

[9] Parry, L.: L3dgeworld 2.1 input & output specifications. CAIA Technical Report 070808A (August 2007)

[10] Peter Eisert, P.F.: Remote rendering of computer games. In: International Conference on SIGMAP (2007)

[11] Petlund, A., Evensen, K., Griwodz, C., Halvorsen, P.: Tcp enhancements for interactive thin-stream applications. In: 18th International NOSSDAV Workshop, NOSSDAV 2008, pp. 127–128. ACM, New York (2008)

[12] Raaen, K., Espeland, H., Stensland, H.K., Petlund, A., Halvorsen, P., Griwodz, C.: Lears: A lockless, relaxed-atomicity state model for parallel execution of a game server partition. In: 41st ICPPW, ICPPW 2012, pp. 382–389. IEEE Computer Society, Washington, DC (2012)

[13] Sanglard, F.: Doom3 source code review (June 2012)

[14] Shacknews. Quake 3 source code released (August 2005), http://www.shacknews.com/article/38305/quake-3-source-code-released

[15] Shacknews. John carmack and id software's pioneering development work in 3d game engines recognized with two technology emmy awards (July 2008), http://www.shacknews.com/article/38305/quake-3-source-code-released

[16] Crystal Space Team. Crystal space official site (April 2013), http://www.crystalspace3d.org/main/Main_Page

[17] van Waveren, J.: Quake III Arena Bot. PhD thesis, University of Technology Delft (2001)

[18] White, W., Sowell, B., Gehrke, J., Demers, A.: Declarative processing for computer games. In: The 2008 ACM SIGGRAPH Symposium on Video Games, Sandbox 2008, pp. 23–30. ACM, New York (2008)

[19] Wikipedia. Id tech 4 — wikipedia, the free encyclopedia (2013), http://en.wikipedia.org/w/index.php?title=Id_ Tech_4 (Online; accessed April 9, 2013)

[20] Wikipedia. List of open-source video games — wikipedia, the free encyclopedia (2013), http://en.wikipedia.org/w/index.php?title=List_of_open-source_video_games (Online; accessed April 2, 2013)