



The Humm and Strumm Project

Engine Requirements Document

March 21, 2012

Patrick Niedzielski
PatrickNiedzielski@gmail.com

Ricardo Tiago
rtiago.mendes@gmail.com

This document is licensed under the Creative Commons Attribution ShareAlike 3.0 United States license.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 5 |
| 1.1 | Conventions | 6 |
| 1.2 | The Humm and Strumm Project | 6 |
| 1.3 | Revision History | 6 |
| 2 | Goals | 7 |
| 3 | Features | 11 |
| 4 | Technologies | 13 |
| A | CC-BY-SA 3.0 | 19 |
| A.1 | Definitions | 19 |
| A.2 | Fair Use Rights | 20 |
| A.3 | License Grant | 20 |
| A.4 | Restrictions | 21 |
| A.5 | Representations, Warranties and Disclaimer | 23 |
| A.6 | Limitation on Liability | 23 |
| A.7 | Termination | 23 |
| A.8 | Miscellaneous | 23 |
| | Index | 29 |

1

Introduction

A central premise of engineering is the identification of requirements early in the engineering process. These requirements deal on a higher level than the actual design of the item—an engineer tasked with spanning a valley with a bridge must figure out the volume of traffic that will utilize it before he or she deals with the design of the truss structure that will support it. If one considers the development of software as a form of engineering, one will realize the importance of clarifying the requirements early in the software process as well. Were developers not to do this, claiming, perhaps, that *I have the requirements here in my head*, requirements would be fuzzy and improperly communicated to team members, resulting in the different and expanding requirements for a piece of software.

As useful as it is to avoid this so-called “feature creep,” one cannot create a static requirements specification. Requirements are a continual dialog between the developers of the project and the users of the project. Eric Braude hints at this in his three difficulties of creating a successful requirements document: one must “express requirements in ordinary, clear English,” that a non-technical user may still be able to understand the requirements, “organize the requirements into logical groupings” to facilitate ease of use, and “arrange for the management of requirements” in order to allow for changes in the requirements during the software development process (Braude, 5).

As such, it becomes necessary to reevaluate the requirements of a piece of software often through the process, in order to adjust to changes. This has been cited as one of the greatest faults in the “Waterfall Method” of software development, a practice coming from the United States Department of Defense’s project development standards (Wong). Instead, modern development paradigms give greater weight to the idea of continual requirements analysis (Larman and Basili).

The above, though, comes from a business perspective, where there is often a user contracting the software firm for the creation of some software they need. Where, then, does this leave the Free Software project, a project where the developer might not have a clear, single user entity to which he or she can defer?

Requirements analysis still plays a role in the development process for Free Software projects. Often, such projects are multi-developer, multi-site enterprises, where developers are working more individually than are developers in geographically centralized projects. As such, it becomes imperative that each developer understand clearly and critically the requirements of the software they are writing, to avoid any possible

miscommunication. In this case, early on in the process, the analysis of requirements is initiated by the developers themselves, unlike in a contracted firm, where the users initiate the process. The early formalization of requirements gives a basis for future users to work off of in their side of the conversation.

Though there are many methodologies used in Free Software projects, the Humm and Strumm Project believes a slightly modified version of the “VW Spiral Process” (Cockburn) is applicable to Free Software projects following the common adage of “release early, release often.” This process calls for a cyclic form of development, in which requirements are reevaluated often at many levels. Though the Humm and Strumm Project chooses to do the lower levels at an informal level, this document describes the top levels of requirements, global to the entire game engine formally.

1.1 Conventions Used In This Document

This document is meant to be read sequentially, starting off with high level goals in the engine and narrowing the focus to individual components and technologies used within. Each section describes a slightly narrower field than the last.

Central requirements will be defined in boxes separated from the discussion and prose of the section. Within these boxes, the key words **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** in this document are to be interpreted as described in RFC 2119 (Bradner).

1.2 The Humm and Strumm Project

The Humm and Strumm Project is a Free Software Project to create a 3-D adventure game for PCs. Our goals are twofold: our first and current stage is to create an extensible game engine for use both in our game and in others; our next stage will be to create the actual Humm and Strumm game.

Humm and Strumm will be a 3D 2-player video game with unique split-screen game play. The game will feature scalable graphics, ranging from older hardware support to new special effects. The goal is to allow as many people as possible to play Humm and Strumm.

The Humm and Strumm Project is composed of unpaid volunteers from around the world, collaborating on this project in their spare time. Communication is at the heart of the project, because of the great distances that separate us and the collaborative nature of the work. As such, we understand the need for transparency in our work: we need to have information such as design decisions available at a moment’s notice to guide us. We also need to be able to provide all such relevant information to contributors. All our design decisions are available on-line, either on our mailing list or on our websites.

1.3 Major Revision History

| | | |
|------------|-------------|------------------------------------|
| 2011-12-13 | Version 1.0 | Initial version. |
| 2012-03-21 | Version 1.1 | Added annotations to bibliography. |

2

Goals

As explained in the introduction, the Humm and Strumm Project has two goals: to create a Free Software 3D Adventure Game, and to create an accompanying game engine that is not just specific to this game, but rather can be used by other GPL video games. The former goal is largely dependent on the latter, as the engine needs to be in place before work on the game can be done. The early stages of the Humm and Strumm Project will be focused on the specification and implementation of the game engine, as described here.

What is a game engine, exactly? Stefan Zerbst likens it to an actual car engine, its namesake:

You might define an engine as a machine that is used to propel a vehicle. With regard to game programming, an engine is a part of your project that propels certain functionalities of your program. It is quite clear that if you put a key into a car's ignition lock and turn it, the car's engine should crank up and the car should start. You then put your foot on the accelerator to get the car to move. Inside, your car's engine transfers kinetic energy to the axle-drive shaft to make things move. The driver does not need to know exactly what is going on inside the engine, nor does he care if he just wants to drive the car. (Zerbst and Düvel, p. 3)

Zerbst describes an engine as a "black-box," whose internals the user sitting at a higher level does not need to understand, but whose internals provide the functionality of the engine; without the engine, there is no functional car, or in the case of a game engine, functional game. It is important to think of the engine as a library that can be reused across several games.

Because this engine will be used not just by the Humm and Strumm video game, but also by other games, perhaps made by different development teams that are not involved in the creation of the engine at the moment, it is especially important that its purpose be laid out clearly. Furthermore, in analyzing the requirements, we must keep in mind that this engine must not just fulfill requirements needed by Humm and Strumm, the game, but also those that would be applicable to a general game engine. Humm and Strumm will be built upon the engine just like any other game would, with no hidden APIs to be used.

General Purpose. The Humm and Strumm Engine must be a general purpose, three-dimensional video game engine for computer systems.

Though the selection of our intended audience to PC users limits the number of people who will be able to play games written using the Humm and Strumm Engine, it drastically reduces the complexity required to optimally support PCs as well as consoles. While Windows and OS X are dissimilar, a small layer can be written into the engine to abstract the differences. With consoles, the differences are far more fundamental than are the differences between various computer systems.

While we could decrease complexity even further by limiting our allowed platforms, this would adversely affect our audience. In the interest of spreading quality Free Software gaming, the Humm and Strumm Project has decided against this. The Project's goal, instead, is to support as wide a range of computer platforms as possible, both in terms of operating systems and hardware setups. Though it would be a waste to attempt to support every platform that exists, we want to have as wide a base as possible.

Cross-Platform. The Humm and Strumm Engine should be written to be completely cross-platform. This means any system-specific portions of the code that are not required by a game engine should not prevent the code from running without that functionality on other systems.

FOSS gaming generally is not up to the same standards as proprietary games, developed by development teams. While recognizing that this could be due in part to the processes of game development differing between the approaches, the Humm and Strumm Project believes that the Free Software development process is not drastically less suited to game development than is a process in which all designers, developers, artists, and musicians are on the same site at the same time, yielding much more instant communication. The Humm and Strumm Project attributes this more to a lack of high quality tools with which to develop games. While there exist high quality libraries and software packages that each perform a function required by games well, they integrate together very poorly and only with substantial effort. There are not many full-scale, modern game engines that are also Free Software.

The term "modern" in itself is somewhat void of meaning; what does it mean to be a "modern game engine"? The meaning of "modern" is constantly shifting, so, that we may define the word to a narrow and specific meaning, we will call a "modern game engine" an engine that follows current industry practice in using new technologies, such as OpenCL and shaders, and in using new designs, such as those implementing optimal multi-threading techniques for real time systems.

Modern. The Humm and Strumm Engine should take advantage of modern technologies and designs for game engines, including multi-threading, OpenCL, and efficient rendering techniques.

It might seem that the last two requirements are at odds with one another. To a degree, they are. To use modern technologies seems to be to exclude older systems, and to support older systems seems to prevent the use of modern technologies. The Humm and Strumm Project aims to balance these two requirements as much as possible. Whenever a modern technology is available, it ought to be used; whenever it is not, the engine should be able to fall back to a less intensive technology that is supported. While this may reduce visual quality on the lower end system compared

to the higher end system, it does bring the game engine to that lower end platform, while maintaining the higher performance and quality of the better system.

Scalable. The Humm and Strumm Engine must be scalable, able to run on older systems as well as utilize modern technologies when present on newer systems.

With these overarching goals set down, the specific features of the Humm and Strumm Game Engine can now be discussed.

Features

Here are presented the intended features of the engine. This list is neither binding nor exhaustive: it is subject to change at a future date.

Framework

- Event system
- Dynamic type system
- Runtime modifiable entities

Language Support

- Unicode support
- Localizable strings

Rendering

- Foliage
- Water
- Particle effects (smoke, etc.)
- Lighting and shadows
- Bump mapping and parallax occlusion
- Shaders
- Clouds and weather
- Time of day
- 2D rendering

Scene Graph

- Effect nodes
- Model nodes
- Integration with physics

Physics

- Rigid body mechanics
- Particle physics
- Offloading onto GPU where possible

Collision

- Quad-tree and oct-tree subdivision
- Impulse forces
- AABBs

Audio

- 3D sound
- Integration with scene graph
- Environmental effects

Containers

- Locked Queue, Stack
- Lockless Queue, Stack
- Bounds-checked array
- Unchecked array

- List, Tree, Map

- Iterators

File Formats

- PNG images
- JPEG images
- Ogg Vorbis audio
- uncompressed WAV audio
- Ogg Theora video
- WebM video

Programmatic Support

- C++ linking
- High-level scripting language

File System

- Limited write permission to one directory
- Seamless reading of ZIP and TAR archives
- Decompression of GZIP and BZIP2
- XML parser
- IRI support

Networking

- Local area network (LAN) connectivity
- Integrated with event system

4

Technologies

Keeping in line with our goal of being a cross-platform game engine, we have tried to select technologies that are wide-spread and known to work on many platforms. Any choice of technologies will provide an inherent limitation of allowed platforms, so it is imperative to select a set of technologies that are supported by a wide range of platforms.

As a 3D game engine, we need to use some technology to interface with the graphics hardware of a system. While it is possible to do rendering completely in software, most systems have had dedicated graphics chipsets going back many years. In order to keep our engine cross-platform, the Humm and Strumm Project has decided to use the industry standard OpenGL, which works across platforms, and hides from us the differences between different graphics units and even allows the use of software rendering.

OpenGL must be supported by a system for the engine to run on it. Most major platforms have this: Windows and OS X both have built-in support for it; other platforms can use libraries such as Mesa3D¹ which implement the OpenGL standard.

OpenGL also provides a large feature set, ranging from control of the time rendering data is sent to the hardware to graphics shaders that can run on the graphics hardware itself, replacing part of the set pipeline. These features give the game engine very fine control over its rendering, while maintaining its cross-platform nature. OpenGL is the obvious choice for cross-platform 3D rendering development.

OpenGL. The Humm and Strumm Engine must support the OpenGL rendering API, as well as associated technologies.

OpenGL's sister project, developed by Creative Labs, is called OpenAL. OpenAL is an audio specification which, like OpenGL is cross-platform, and abstracts away the complexities of the system's native sound libraries. Instead of needing to support code for WinMM, PulseAudio, and ALSA, the game engine can support only OpenAL, which will automatically select the correct back end to interface used by the system.

OpenAL, too, provides useful features, such as the ability to play "3D sound," which is mapped to stereo or surround sound output, producing the impression that a sound in the game is coming from a certain direction relative to the listener. OpenAL's 3D sound system also calculates distortions from the Doppler effect and allows

¹Mesa3D can be found here: <http://mesa3d.org/>.

for various objects to block the sound. OpenAL additionally allows effects to be applied to the audio, yielding an environmental effect which can be applied to certain locations within the game world.

OpenAL. The Humm and Strumm Engine should support the OpenAL audio API.

There is a push in recent games to utilizing the extra power provided by the GPU when it is not rendering, offloading some tasks from the CPU and thereby allowing the CPU to do other tasks. This is still a very new technology, but early on, it was standardized by the Khronos Group. This standard, called OpenCL, provides an API for using extra computing devices for general purpose computing using a special dialect of the C programming language. This standard is not supported by all systems yet, though.

OpenCL represents an opportunity to use the GPU (and even PPU^s—Physics Processing Units) to do general purpose computing in parallel, such as particle simulations within a game engine. On systems where it is present, it has the potential to increase the efficiency of games and allow them to produce better, more realistic effects.

OpenCL, however, relies on implementations, most of which are not Free Software. Because the standard is freely available, there have been attempts to provide Free Software libraries and drivers to support it²; it is not against the principles of Free Software to implement code to support a Free standard, even if most of the implementations are not Free Software. It may, however, require an exception in the licensing terms of the engine as allowed under the GPLv3 to link with the non-Free implementations.

OpenCL. The Humm and Strumm Engine should support the OpenCL API for offloading certain tasks from the CPU onto alternate hardware processors, such as GPUs.

In games, there is a need to save data in a computer-readable manner that is highly structured. Though it is possible to do this in a binary format, this hides the data from the user looking at the save file and requires the specification of a binary format with each byte accounted for. Instead, the Humm and Strumm Project has decided to use an XML-based file format parser for saving and loading most files generated by the engine.

The eXtensible Markup Language (XML) is a standard specified by the W3C and, at the time of writing, is at version 1.1. XML is a tag-based text file format specification that can be used to implement any type of data storage. XML is a highly structured version of SGML, with simpler syntax for easier machine parsability. Though its original purpose was to store and transmit files on the web, its general specification has led it to use elsewhere, as well.

XML has several related technologies, such as XSL, the eXtensible Style-sheet Language, for transforming and displaying XML files, XML Schema, which specifies how a specific file format using XML must be structured to be valid, and XML Namespace, a method of associating a certain file format using XML with a URN and allowing multiple file formats to be used in one file.

²Clover is one such attempt, started during the 2011 Google Summer of Code. You can find the Git repository here: <http://git.freedesktop.org/~steckdenis/clover/>

XML. The Humm and Strumm Engine should support XML and its related technologies for saving and reading data to disk.

In the interest of reaching the widest audience possible, The Humm and Strumm Project wishes to translate all user-visible strings into other languages, allowing a user to run the game in his or her native tongue. Although this is not nearly as difficult as it was even 15 years ago, when multiple, national encoding formats dominated (Spolsky), it still represents a challenge. The modern mechanism for dealing with multiple languages is to use a standard named Unicode, which allows for transfer of plain text files containing multiple languages without any problems caused by incompatible character encoding formats.

Unicode accomplishes this by setting aside a range of over a million code points for use by modern and ancient scripts, as well as several private use areas. With this space, a file can contain Han characters, Latin characters with circumflexes, and Kannada characters without the need to somehow switch character encodings within a file.

Unicode does come with its own problems, however. Software that uses Unicode must beware of attributes of each code point, requiring a large database. This can be optimized with multistage tables (Allen et al., p. 140), but such is not trivial. One also needs to normalize multiple equivalent code points representing the same text to a canonical form, and deal with multiple ways of encoding the Unicode file itself, UTF-8, UTF-16 (big and little endian), and UTF-32 (big and little endian).

Unicode. The Humm and Strumm Engine must support Unicode in all strings and string algorithms.

Appendices



Creative Commons Attribution-ShareAlike 3.0 United States License

Creative Commons Corporation is not a law firm and does not provide legal service. Distribution of this license does not create an attorney-client relationship. Creative Commons provides this information on an “as-is” basis. Creative Commons makes no warranties regarding the information provided, and disclaims liability for damages resulting from its use.

The Work (as defined below) is provided under the terms of this Creative Commons Public License (“CCPL” or “License”). The work is protected by copyright and/or other applicable law. Any use of the work other than as authorized under this license or copyright law is prohibited.

By exercising any rights to the work provided here, you accept and agree to be bound by the terms of this license. To the extent this license may be considered to be a contract, the licensor grants you the rights contained here in consideration of your acceptance of such terms and conditions.

A.1 Definitions

- a. “**Collective Work**” means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with one or more other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
- b. “**Creative Commons Compatible License**” means a license that is listed at <http://creativecommons.org/compatiblelicenses> that has been approved by Creative Commons as being essentially equivalent to this License, including, at a minimum, because that license: *i*) contains terms that have the same purpose, meaning and effect as the License Elements of this License; and, *ii*) explicitly permits the relicensing of derivatives of works made available under that license under this License or either a Creative Commons unported license or a Creative Commons jurisdiction license with the same License Elements as this License.

- c. **“Derivative Work”** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image (“synching”) will be considered a Derivative Work for the purpose of this License.
- d. **“License Elements”** means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, ShareAlike.
- e. **“Licensor”** means the individual, individuals, entity or entities that offers the Work under the terms of this License.
- f. **“Original Author”** means the individual, individuals, entity or entities who created the Work.
- g. **“Work”** means the copyrightable work of authorship offered under the terms of this License.
- h. **“You”** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

A.2 Fair Use Rights

Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

A.3 License Grant

Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

- a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
- b. to create and reproduce Derivative Works provided that any such Derivative Work, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked “The original work was translated from English to Spanish,” or a modification could indicate “The original work has been modified.”;
- c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

- d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works.
- e. For the avoidance of doubt, where the Work is a musical composition:
 - (i) **Performance Royalties Under Blanket Licenses.** Licensor waives the exclusive right to collect, whether individually or, in the event that Licensor is a member of a performance rights society (e.g. ASCAP, BMI, SESAC), via that society, royalties for the public performance or public digital performance (e.g. webcast) of the Work.
 - (ii) **Mechanical Rights and Statutory Royalties.** Licensor waives the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work (“cover version”) and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).
- f. **Webcasting Rights and Statutory Royalties.** For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved.

A.4 Restrictions

The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

- a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of a recipient of the Work to exercise any of the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. When You distribute, publicly display, publicly perform, or publicly digitally perform the Work, You may not impose any technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise of the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective

Work any credit as required by Section 4(c), as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any credit as required by Section 4(c), as requested.

- b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under: (i) the terms of this License; (ii) a later version of this License with the same License Elements as this License; (iii) either the Creative Commons (Unported) license or a Creative Commons jurisdiction license (either this or a later license version) that contains the same License Elements as this License (e.g. Attribution-ShareAlike 3.0 (Unported)); (iv) a Creative Commons Compatible License. If you license the Derivative Work under one of the licenses mentioned in (iv), you must comply with the terms of that license. If you license the Derivative Work under the terms of any of the licenses mentioned in (i), (ii) or (iii) (the “Applicable License”), you must comply with the terms of the Applicable License generally and with the following provisions: (I) You must include a copy of, or the Uniform Resource Identifier for, the Applicable License with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform; (II) You may not offer or impose any terms on the Derivative Works that restrict the terms of the Applicable License or the ability of a recipient of the Work to exercise the rights granted to that recipient under the terms of the Applicable License; (III) You must keep intact all notices that refer to the Applicable License and to the disclaimer of warranties; and, (IV) when You distribute, publicly display, publicly perform, or publicly digitally perform the Work, You may not impose any technological measures on the Derivative Work that restrict the ability of a recipient of the Derivative Work from You to exercise the rights granted to that recipient under the terms of the Applicable License. This Section 4(b) applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of the Applicable License.

If You distribute, publicly display, publicly perform, or publicly digitally perform the Work (as defined in Section 1 above) or any Derivative Works (as defined in Section 1 above) or Collective Works (as defined in Section 1 above), You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (a) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or (b) if the Original Author and/or Licensor designate another party or parties (e.g. a sponsor institute, publishing entity, journal) for attribution (“Attribution Parties”) in Licensor’s copyright notice, terms of service or by other reasonable means, the name of such party or parties; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and, consistent with Section 3(b) in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., “French translation of the Work by Original Author,” or “Screenplay based on original Work by Original Author”). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear, if a credit for all contributing authors of the Derivative Work or Collective Work appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance

of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

A.5 Representations, Warranties and Disclaimer

Unless otherwise mutually agreed to by the parties in writing, Licensor offers the Work as-is and only to the extent of any rights held in the licensed work by the Licensor. The Licensor makes no representations or warranties of any kind concerning the Work, express, implied, statutory or otherwise, including without limitation, warranties of title, marketability, merchantability, fitness for a particular purpose, noninfringement, or the absence of latent or other defects, accuracy, or the the presence of absence of errors, whether or not discoverable. Some jurisdictions do not allow the exclusion of implied warranties, so such exclusion may not apply to You.

A.6 Limitation on Liability

Except to the extent required by applicable law, in no event will Licensor be liable to You on any legal theory for any special, incidental, consequential, punitive, or exemplary damages arising out of this License or the use of the Work, even if Licensor has been advised of the possibility of such damages.

A.7 Termination

- a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
- b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

A.8 Miscellaneous

- a. Each time You distribute or publicly digitally perform the Work (as defined in Section 1 above) or a Collective Work (as defined in Section 1 above), the Licensor

offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

- b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
- c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
- d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
- e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Creative Commons Notice

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, Creative Commons does not authorize the use by either party of the trademark “Creative Commons” or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons’ then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time. For the avoidance of doubt, this trademark restriction does not form part of this License.

Creative Commons may be contacted at <http://creativecommons.org/>.

Bibliography

Allen, Julie D., et al., eds. *The unicode standard, version 6.0*. 1st ed. Mountain View: The Unicode Consortium, February 2011. ISBN: 978-1-936213-01-6.

The Unicode Standard is, of course, the definitive reference when it comes to understanding and implementing Unicode in software applications. Although mainly a reference to the interchange requirements, the Unicode Standard does also list implementation guidelines for authors wishing to use Unicode in their applications. Version 6.0 adds many characters, for more complete African and East Asian language support, for additional emoji icon characters, and for the new Indian Rupee currency symbol.

Bradner, Scott. *Key words for use in RFCs to indicate requirement levels*. RFC-2119. Request for Comments. The Internet Engineering Task Force, March 1997. <http://www.ietf.org/rfc/rfc2119.txt> (accessed 03/14/2012).

Many IETF specifications use standard and precise terminology to indicate with what level of necessity an implementation must follow the specification's individual requirements. This terminology was first laid out in RFC 2119 and has since become widely used, both in IETF RFCs and in other specifications. Because this requirements document is simply a collection of requirements for a later implementation of a game engine, we follow these conventions throughout this document to the degree they are applicable.

Braude, Eric. *Software design: from programming to architecture*. 1st ed. Ed. Christine Cervoni. Hoboken: John Wiley & Sons, Inc., 2004. ISBN: 0-471-42920-1.

Braude's primer on software engineering puts forth many recommendations for students in the field of software engineering, with a focus on modern Java development style. Despite focusing design focuses (perhaps too heavily on design patterns, it still contains a clear description of the considerations one must make during the requirements specification phase of software engineering. This section of the book is rather language agnostic and also discusses several requirements processes, such as the Waterfall method and the Iterative method.

Cockburn, Alistair. *Using VW staging to clarify spiral development*. HaT TR 1997.02. Technical report. Humans and Technology, May 1997.

Unlike many authors, Cockburn realizes the need for a linear calendar view at the level of project managing. Cockburn describes a technique to integrate the standard iterative development of software developers with the unidirectional flow needed by project managers and planners, a technique he terms the “VW Spiral Process.” Cockburn states that this process “lets developers work in the incremental, iterative manner they need, and lets the calendar owner map that work to straight calendar time.”

Larman, Craig, and Victor R. Basili. Iterative and incremental development: a brief history. *IEEE Computer* 36, no. 6 (June 2003): 47–56. ISSN: 0018-9162.

In their article “Iterative and Incremental Development,” Larman and Basili provide a comprehensive overview of the history of such iterative development practices that have become common in the past decade, but were before then less common than the Waterfall method. Each process that Larman and Basili touch upon may have different details, but all fall under the category of the Iterative method. Though the Humm and Strumm Project chooses not to use the Iterative method proper, it does use an incremental development process with the same intent of the Iterative method: to remove the single-pass aspect of the Waterfall development process, to allow for flexible requirements that may change as development progresses, and to often reanalyze requirements to make sure the software is up-to-date.

Spolsky, Joel. *The absolute minimum every software developer absolutely, positively must know about unicode and character sets (no excuses!)* October 2003. <http://www.joelonsoftware.com/articles/Unicode.html> (accessed 12/13/2011).

Driven by the prevalence of Unicode missupport in modern applications, author Spolsky wrote his famous and must-read article to educate the modern developer on the basics of Unicode. Spolsky deals with character encodings and code pages from a historical perspective, leading to the current technology of Unicode. This article is from, at the time of writing, eight and a half years ago, though, and many of the arguments about support for legacy, non-Unicode character encodings are representative more of an earlier era in Unicode support. The points that are made, though, are still relevant and should still be considered by the modern software engineer in any discipline that deals with strings.

Wong, Carolyn. A successful software development. *IEEE Transactions on Software Engineering* 10, no. 6:714–727. ISSN: 0098-5589.

The headache of broken budgets and broken schedules has been a problem in the software industry since its inception. Wong describes the method used by the System Development Corporation in 1980 in its creation of an air defense system for a foreign contractor, in which the software was created within the budget constraints and within the optimistic 25 month schedule. Wong contrasts this paradigm with the industry standard of the time, the Waterfall method, which she attributes to the Department of Defense.

Zerbst, Stefan, and Oliver Düvel. *3d game engine programming*. 1st ed. Ed. Mark Garvey. Boston: Premier Press, 2004. ISBN: 978-1-592003-51-8.

The creation of a game engine often represents the most significant period of work for a development team. Professional game engines often run with licensing costs

of hundreds of thousands of dollars. For a hobbyist developer, these costs are prohibitively high. Zerbst, founder of the online game creation community ZFX.info, and Düvel, a project manager with three decades of 3D graphics development experience, share their tips for creating a game engine a hobbyist can use, starting from scratch and working up to a general purpose engine, which they showcase with a first-person shooter game. This guide, while definitely representative of older-style engines, provides a solid basis for understanding the internal structure of a game engine.

Index

- A**
Apple Mac OS X, 8, 13
- C**
CC-BY-SA, *see* Creative Commons, License
compression
 BZIP2, 12
 GZIP, 12
 ZIP, 12
conventions
 in this document, 6
Creative Commons
 License, 19
cross-platform, 8, 13
- D**
Department of Defense, 5
DoD, *see* Department of Defense
- E**
engine, *see* game engine
- F**
feature creep, 5
Free Software, 5, 14
 development, 8
 gaming, 8
- G**
game engine, 7
- H**
Humm and Strumm, *see* The Humm and Strumm Project
- J**
JPEG format, 12
- M**
Macintosh, *see* Apple Mac OS X
Mesag3D, 13
Microsoft Windows, 8, 13
- O**
Ogg
 Theora format, 12
 Vorbis format, 12
OpenAL, 14
OpenCL, 8, 14
OpenGL, 13
OS X, *see* Apple Mac OS X
- P**
PNG format, 12
- R**
requirements analysis, 5
- T**
The Humm and Strumm Project, 6–8, 13, 14
- U**
Unicode, 11, 15
United States Department of Denfense, *see* Department of Defense
- V**
VW spiral process, 6

W

waterfall method, 5

WebM format, 12

Windows, *see* Microsoft Windows

X

XML, 12, 15