Assessing Game Re-implementation: Reverse Engineering through Dynamic Analysis of the Velocity Engine for Ricochet Xtreme

Francis Dominic O. Fajardo and Joseph Anthony C. Hermocilla

I. Introduction

A. Background of the Study

Video games have been around for more than six decades, and with that in mind, classic and retro titles are gaining popularity with both older and younger gamers [1]. However, revisiting those games can be hindered by hardware, compatibility, and other technical issues. Because acquiring the original hardware is often impractical and expensive [2], gamers often turn to alternative methods, such as emulation.

Current game preservation research is focused on emulation, mainly because it does not modify the software [3], and as evidenced by recent conferences on video game preservation [4]. By contrast, re-implementation, an approach inconsistently termed as or associated with recreation [5] [6], re-enactment [6], or reprogramming [7], is often noted in literature as time-consuming, costly, or error-prone for more complex games [7] [8] [9]. Moreover, it should not be confused with migration, which requires access to the original source code and assets [6], modifying it to work on newer platforms, and involves updating them to newer formats [8], with a supposed threat of "progressive degradation if format conversion is not perfect" [3]. Many independent programmers have also documented their attempts in an informal manner, posting their findings in blogs and forums.

But beyond those anecdotes, there are little to no formal studies that systematically evaluate the viability of re-implementation when compared to other methods. This is despite its capability in several community-driven projects to enhance old games with support for modern platforms, quality-of-life improvements, and better modding tools. Daggerfall Unity, for The Elder Scrolls II: Daggerfall, fixes original bugs, provides widescreen support, and is playable across different platforms without emulation [10]. FreeSO, for The Sims Online, expands upon original gameplay, reuses all of the resources of the original game, and maintains backwards compatibility with the original Sims engine [11]. OpenTTD, for Transport Tycoon Deluxe, recreates the original game and extends it with many new features [12].

The lack of analysis into re-implementation has limited our understanding of the viability of this approach, considering the recent advancements in reverse engineering tools and the widespread availability of resources and guides in reverse engineering.

B. Statement of the Problem

Ricochet Xtreme is a breakout-style game released for Windows by Reflexive Entertainment in November 2001 [13] and was developed using the proprietary Velocity Engine [14] [15]. It is incompatible with modern operating systems such as Windows 11, leading to graphical issues and random crashes, reducing accessibility for contemporary players. Although it can be emulated, there is a performance overhead and inflexibility associated with running a full Windows VM to play this game. This exemplifies the need to investigate re-implementation as an alternative strategy for old PC games, particularly given advances in reverse engineering tools and resources.

C. Objectives of the Study

The general objective of this study is to determine whether game re-implementation is a viable method for preserving video games than traditional emulation, by employing software reverse engineering and producing a functional re-implementation of the Velocity Engine for Ricochet Xtreme. Specifically, it aims to:

1) analyze and decode the data structures and proprietary file formats used by the Velocity Engine;
2) document Velocity Engine's property management and plugin system;
3) reconstruct parts of the Velocity Engine on top of an open-source game development framework;
4) reconstruct parts of Ricochet Xtreme on top of the reverse engineered engine; and
5) evaluate and compare user experience, performance, and compatibility when compared to the original game.

D. Significance of the Study

Re-implementation is a method taken by many independent developers as an alternative to other methods such as emulation. However, the experience of doing so and its viability has not been thoroughly investigated, reviewed, and evaluated in a formal study given its informal nature. This study will be useful in understanding the process of re-implementation for old video games, with a focus on the Velocity Engine.

Presented to the Faculty of the Institute of Computer Science, University of the Philippines Los Baños in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science

© 2025 ICS University of the Philippines Los Baños

Lastly, this study serves as a foundation for future research and practical actions in reverse engineering old game engines. Likewise, this work could then be extended to other titles that were made using the same game engine.

E. Place and Time of the Study

The study will be carried out in the Institute of Computer Science (ICS) located at the University of the Philippines Los Baños, Laguna, Philippines.

The study will be conducted and evaluated online with the help of the Ricochet Players Discord Community (RPDC) with a prospect time of study from January 2026 to April 2026.

F. Scope and Limitations

This study will only focus on the re-implementation of the specific version of the Velocity Engine that was used for Ricochet Xtreme, and will not consider other games that were built on another version of the Velocity Engine. Different games have different requirements and may have modified the underlying game engine to use other types of file formats and structures that are absent in Ricochet Xtreme. Moreover, the re-implemented engine will not support the Velocity Engine's 3D feature set.

G. Definition of Terms

1) subsystem – a module of a larger system that handles a specific area of functionality.
2) game engine – a framework for game development that provides features, tools, and subsystems for game developers, such as a renderer.
3) Velocity Engine – a data-driven C++ game engine developed by Reflexive Entertainment.
4) overhead – extra time, memory, or other resources a system consumes beyond what is needed for a specific task or function.
5) serialization – a process of converting the state of a data structure to a storable format.
6) deserialization – the inverse of serialization.
7) property management – a subsystem of a game engine that is used for serializing and deserializing several forms of data, often using a single file format.

II. Review of Related Literature

A. Reverse Engineering

Software reverse engineering, in the field of computer science, is the process of analyzing compiled programs and understanding its components and relationships. The process of reverse engineering has been around since the 1980s, with applications in legacy software, software engineering, and anti-malware detection [16]. Under this, the process of software analysis can be performed using three ways: (a) static, (b) dynamic, and (c) historical [17]. Static analysis is performed without executing the program [17], but is limited in characterizing novel or obfuscated code [18] and can only extract important information or characteristics [19]. Dynamic analysis requires program execution and involves examining, observing, and tracing its actions [17]. Moreover, changes to files in the environment, network communication, and file structures can be identified using this method [18]. An isolated virtual environment is often used to avoid damaging the host system or network [19]. Historical analysis requires access to the source code and a log of the changes recorded by the versioning system [17].

B. Gray-box File Formats and Taint Analysis

Cui et al. [20] defined gray-box file formats as those whose documentation of their structure is limited, despite the availability of sample files and programs that can parse and handle them. In the context of video games, the executable functions as the specific processor for assets that are otherwise encoded in proprietary and undocumented file types. One method of inferring the structure is through taint analysis. Taint analysis is a data-oriented software analysis technique that involves monitoring the origin and propagation of user inputs to establish a direct association between specific data bytes and their processing paths within a program [20]. Lin et al. [21] devised a reverse engineering scheme known as REWARDS, based on dynamic analysis, that is able to reconstruct the layout of data structures in memory based on derived type information even with only the program's binary executable.

C. Archaeogaming and Methods of Game Preservation

Reinhard [22] defined archaeogaming as the archaeology of both "in and of digital games". In the context of this study, the latter definition will be used, with a focus on the tools used in video game development, particularly video game engines.

Before establishing and evaluating re-implementation as a viable method, it is important to understand the strategies used in digital preservation. Ponsard [3] identified and compared seven (7) digital preservation strategies: (a) total preservation, (b) standardization, (c) encapsulation, (d) extraction, (e) migration, (f) emulation, and (g) a (universal) virtual machine. In terms of game preservation, most formal studies are focused on emulation. Salvador [2] claimed that it is the easiest and most cost-efficient way of reissuing games, adding that it has been considered viable in the 1990s, but was consistently criticized because of its associations with piracy.

Re-implementation, on the other hand, does not have a standard definition despite its use in many community projects. Its description varies depending on the source being used. In several studies, the process is referred to as a recreation [5] [6], re-enactment [6], or reprogramming [7]. The maintainer of a repository that tracks game re-implementations referred to it as open-source game remakes [23]. Guttenbrunner et al. [5] [24] considered game engine recreation as a "migration" approach despite being composed of entirely new code. McDonough et al. [6] contrasted this with a new classification known as "Re-enactment", where recreation or re-implementation would fall under experience-based re-enactment. Despite the high variation of terms used, they all refer to the same action or process. For instance, although ScummVM officially describes itself as a *virtual machine* in name, it is fundamentally a re-implementation of several different game engines in a single package [25].

D. Related Studies

Reverse engineering is often applied when analyzing computer malware and viruses, but this approach can also be used for examining and reconstructing historical, classic video games as part of the nascent field of archaeogaming. Older video games have become increasingly difficult to run on modern hardware. This is particularly important in the context of video game preservation, where a fear of losing the ability to play them again in the future is a growing concern [26]. As explained by Salvador [2], the commercial availability of pre-2010 video games is limited at only 13%. Aside from limited availability, making them work on newer hardware would be difficult if there is an inadequate understanding of the original program and its underlying platform. One way to approach this is through reverse engineering, which has been applied in many fields of computer science. Yong Wong et al. [27] confirmed that reverse engineering heavily relies on static analysis, but found that dynamic analysis is also effective for certain types of malware. These approaches help towards the understanding of how the original program operates. While emulation is the most common and economical way of reissuing and playing older games [2], reconstruction also has value in terms of understanding how older games work, as revealed by an examination of the code of Entombed [28]. Modding communities have employed the latter in terms of improving and building upon the original game. The viability of video game re-implementation (through reconstruction) can be assessed and viewed through the role it plays in video game preservation and analysis, the use of decompilers and related tools and the utilization of dynamic analysis in the creation of custom file type signatures as part of the process, and its practical use in the field of archaeogaming.

First, it is explored by understanding its relevance in video game preservation and analysis. Salvador [2] discovered that the commercial availability of classic, pre-2010 video games in the United States is extremely limited at only 13%. This is comparable to figures presented for other at-risk media, such as pre-World War II audio recordings (10% or less) and American silent films (14%). Moreover, Salvador [2] added that the Commodore 64 platform is an abandoned ecosystem, with its game library effectively abandoned, despite its historical significance. This exemplified the importance of understanding how older games were made in order for them to continue being playable in the future. Aside from availability, other factors must also be considered. In a study by Bettivia [29], properties in the code, environment, and the surface and social aspects of games were seen as significant by stakeholders, as implied by data from Preserving Virtual Worlds II (PVWII). In terms of reverse engineering, there is interest from players to expand and improve upon the original game. Dym et al. [26] found that most of the study's participants became interested in reverse engineering the extracted game code to further understand how the game works and were subsequently involved in modding communities, as a result of their experience with playing modified and emulated game ROMs. Moreover, Dym et al. [26] explained that many of the participants aimed to further improve and create new experiences with the archived games rather than leaving them unchanged. A few examples include developing ROM hacks and mods for the games or even new multiplayer experiences.

Several decompilers and related tools can be employed to ease the process of understanding the original game. Reverse engineering from scratch is not a heavily daunting experience anymore because of many modern tools that have been developed in recent times. One such tool is a decompiler. A study by Gao et al. [30] evaluated the decompilers using two metrics: (a) Recompile Success Rate (RSR) and Coverage Equivalence Rate (CER). Hex-Rays demonstrated the best performance among traditional decompilers, with the highest single optimization level RSR of 70.6% for programs compiled with no optimization, and in terms of averaged metrics, with a 58.3% RSR and 41.7% CER. However, Gao et al. [30] found that there is a consistent decline in both RSR and CER as the optimization level increases, highlighting the challenges in recovering compiler-compatible code from aggressively optimized binaries. These findings are further supported by Cao et al. [31] who discovered that based from the results of using Hex-Rays (IDA), it is estimated that 30%-50% of the decompiled code can be directly recompiled. However, other decompilers were only able to successfully recompile 5.6% of the decompiled code. Cao et al. [31] added that based on the automated test findings, the Hex-Rays decompiler performed the best on average, passing all test cases and achieving high similarity with the original binary.

Video games often use proprietary file formats and borrowing from the field of malware analysis, custom file type signatures can be created by integrating dynamic analysis. Yong Wong et al. [27] explained that dynamic analysis is often effective for malware that runs and produces an output. Otherwise, several evasion techniques have to be employed in order to make it execute instructions, which varies depending on the malware sample being tested. Moreover, Yong Wong et al. [27] added that emulating the targeted environment accurately in the analysis environment is necessary in inducing certain types of malware to reveal their malicious behavior. In terms of video games, the working runtime environment of the original game will have to be emulated. Unlike malware, games almost always run and produce a visual output on the screen that can be easily observed. Video games employ a variety of file formats, such as in the graphics that it uses for displaying in the screen. This is often proprietary and non-standard, which means that standard libraries (e.g., JPEG) cannot be used directly. Analyzing these file formats therefore becomes necessary. Fereli et al. [32] revealed in their study that custom signatures for all specialized file types significantly enhanced the identification and recovery rates, as opposed to the default PhotoRec configuration. For all specialized file types (RFID, NFC, Sub-GHz, and iButton), the identification rates (I), recovery rates (R), and the correct extension rates (TPI and TPR) all improved to 100%. Additionally, Fereli et al. [32] claimed that unique patterns and data structures are often used for some file types and cannot be detected easily by general-purpose recovery tools. Adequately resolving this requires the analysis and creation of custom signatures.

Lastly, the game's inner workings are important in the field

of archaeogaming. Aycock & Copplestone [28] found that the code for maze generation algorithms used by different games, including Entombed, are dissimilar, even among those with solvable mazes. The code of retrogames reveal programmer creativity, supporting the point that the implementation features of these games should be examined. Aycock & Biittner [33] noted that several aspects of the original II/64 hardware cannot be reconstructed with full accuracy. Moreover, Aycock & Biittner [33] added that behavioral experiments were conducted using the II/64 software for the Apple IIe, the remaining digital artifact. A bug in its operation was eventually discovered and a hypothesis as to why this occurred was tested. Moreover, restrictions in the functionality available to the programmer were exposed, such as a read-only view to the Commodore 64's CPU registers.

III. Materials and Methods

To address the identified gap, the research will proceed as follows:

1) *Format Research, Identification, and Specification*. The game's file formats will be examined using a hex editor (ImHex) and reverse engineered to reveal how they should be read, parsed, and interpreted. On the other hand, the output files of the property management system used by the game engine will be reviewed to determine the various classes and structures attached to and used by the game. Game behavior will also be examined to determine what and how parts of the engine should be recreated.
2) *Implementation*. The recreated Velocity Engine will be built on top of SDL3, within which the previously created specifications will be used to read, parse, and interpret game files. The recreated engine will be the basis for re-implementing Ricochet Xtreme.
3) *Experimental Setup*. A standard Windows 11 PC will be used to simulate various in-game scenarios. The original game and the re-implemented game will both be run on the same machine.
4) *Testing and Evaluation*. Compare the re-implemented game against the original copy of the game. Participants who will test and evaluate the game will be recruited from RPDC. The participants will be asked to perform a set of tasks and several metrics related to user experience, performance, and compatibility will be evaluated using a survey instrument.
5) *Validation*. Analyze results to confirm improvements in game performance and compatibility with modern systems.

A. Development Tools

The re-implemented game will be developed on a machine with the following specifications:

- Operating System: Artix Linux 64-bit
- Processor: AMD Ryzen 4300U
- Memory: 16 GB RAM

The following software development tools will be used for developing the game:

- Environment
  - Visual Studio Code
  - ImHex
- Technologies
  - SDL 3
  - C++
  - PhysicsFS
  - zlib

B. Features

This section shows the features of the game that the user can access:

1) Main Menu
2) Statistics (High Scores)
3) Options
4) Level Selection
5) Main Game Screen
6) Pause Screen
7) Level Editor (This is absent from the original game)
8) Reactive Music

Some of these features are detailed in Figs. 1–4.

Fig. 1. Main Menu Screen

Fig. 2. Statistics (High Scores) Screen

C. Game Engine

Along with the game, its underlying game engine will be reverse engineered. It will be built on top of Simple Direct-Media Layer (SDL), which is a cross-platform development library that provides low-level access to various hardware

Fig. 3. Level Selection Screen

Fig. 4. Main Game Screen

and software interfaces [34]. It supports many mainstream platforms such as Windows, Linux, and macOS, as well as bindings for several programming languages. Although using existing game engines like Unity and Godot are more common in the literature [35], [36], SDL was selected because of its performance and platform versatility, as well as its widespread adoption in successful re-implementation projects, including OpenTTD and OpenRCT2.

D. Assets

James C. Smith, the Lead Programmer of Ricochet Xtreme, noted that all of the original art was created using 3D Studio Max and rendered to Truevision Targa (TGA) files, since at the time, that was the easiest way to get 32-bit images with alpha channels that can be read by a C++ program [37]. However, these TGA assets were eventually processed into the game's asset cache and compressed to save space, which was a necessity at the time due to limited storage constraints on the common specifications of computers at the time.

The re-implemented game will reuse the original game's assets. However, these resources will not be distributed or included with the re-implementation and must be provided by the user to avoid infringing on any copyrights. This approach has been used by most re-implementations of old video games, such as FreeSO [11] and OpenTTD, which allows them to exist without having to deal with the legal implications.

E. File Formats

Most of the game's file formats will have to be reverse engineered, since they are currently undocumented.

The game's asset cache contains two proprietary image formats referred to as "frame" and "sequence". The former is used for single, static images, while the latter is used for animated sprite sheets or texture atlases.

Likewise, most of the fonts included with the game use the "sequence" format and does not use the standard TrueType (TTF) and OpenType (OTF) font formats. The Velocity Engine does support the former, however, based on the existence of a single TTF (`Resources/Fonts/TRUETYPES/!default.ttf`) in the data package, which appears to be a bundled copy of Arial.

The Velocity Engine's property management system uses plain text files that can be modified using a regular text editor, aside from using the engine's built-in property editor [38]. The format uses key-value pairs and has structural similarities with the popular web data format, JSON.

The game uses the Ogg Vorbis audio codec for its background music (BGM) and sound effects (SFX). This audio format is a free, open-source, and non-proprietary audio format, with a standard encoder and decoder licensed under a BSD license [39].

The data package which contains all of the game's assets is a standard ZIP archive. However, it has a fairly minor modification that makes file archiving utilities like 7-Zip and WinRAR treat the ZIP file as a broken archive. Reflexive Entertainment changed the ZIP archive's end of central directory record signature to begin with the two ASCII characters (`RE`) instead of (`PK`) [40].

F. Reverse Engineering Process

While tools such as Ghidra exist, it cannot be used to produce a clean-room reverse engineering implementation because of legal issues surrounding its use, especially with the provided executable of the game. Although that is the case, a basic dynamic analysis will be used to examine the game. This process involves running the game and examining its behavior and its effects [19]. On the other hand, basic static analysis will be used on the game's data package and the rest of the files (excluding the executable). This involves extracting important information or characteristics from these files and evaluating their formats [19], using a text editor (for the property files) and a hex editor (for binary files).

It was originally thought that the release versions of the game did not include the ability to create new "frame" and "sequence" files from TGA images, and that this ability was only present in a special internal version of the game [37]. However, it was eventually discovered by one of RPDC's members that this is possible by creating dummy, empty TGA images with the identical base names for all frame and sequence files to trick the game into creating new "frame" and "sequence" files. Smith believed that this was not possible [37], but was proven otherwise by testing around with the files. Along with basic dynamic analysis, this process will be used to reverse engineer the "cached" image file formats of "frame" and "sequence".

G. Testing and Evaluation

This study will employ a non-probability, purposive sampling technique to select participants. The participant must possess prior experience playing the original Ricochet Xtreme game. Each participant will be asked to do the following tasks in both the original and the re-implemented game, using the prescribed experimental setup:

1) Explore the main menu and its connected screens.
2) Play at least three levels.
3) Test the new level editor.

After testing, the participant will be asked to answer a survey questionnaire containing questions that evaluate and tackle the following metrics: (a) usability/playability, (b) personal gratification, (c) game behavior, (d) performance, (e) stability, and (f) platform and hardware compatibility. The first two metrics were adapted from Factor 1 and Factor 7 of the Game User Experience Satisfaction Scale, composed of nine (9) subscales, which was designed for measuring video game satisfaction [41]. Gabriel & Lapitan [35] adapted the GUESS instrument to fit the requirements of their research. Likewise, a similar approach will be done for this study, since a review of the relevant literature indicates that no existing framework fully accommodates all of these metrics in the context of game re-implementation.

APPENDIX A
SURVEY QUESTIONNAIRE

The survey items were formulated as statements, which participants were asked to rate on a seven-point Likert scale to quantify their perceptions. This scale provided a range of responses as follows:

1 – Strongly Disagree
2 – Disagree
3 – Somewhat Disagree
4 – Neither Agree nor Disagree
5 – Somewhat Agree
6 – Agree
7 – Strongly Agree

TABLE I: Usability/Playability

#	Statement	1	2	3	4	5
1	I think it is easy to learn how to play the game.					
2	I find the controls of the game to be straightforward.					
3	I always know how to achieve my goals/objectives in the game.					
4	I find the game's interface to be easy to navigate.					

Continued on next page

TABLE I: Usability/Playability (Continued)

#	Statement	1	2	3	4	5
5	I do not need to go through a lengthy tutorial or read a manual to play the game.					
6	I find the game's menus to be user friendly.					
7	I feel the game trains me well in all of the controls.					
8	I always know my next goal when I finish an event in the game.					
9	I feel the game provides me the necessary information to accomplish a goal within the game.					
10	I think the information provided in the game (e.g., onscreen messages, help) is clear.					
11	I feel very confident while playing the game.					

TABLE II: Personal Gratification

#	Statement	1	2	3	4	5
1	I am in suspense about whether I will succeed in the game.					
2	I feel successful when I overcome the obstacles in the game.					
3	I want to do as well as possible during the game.					
4	I am very focused on my own performance while playing the game.					
5	I feel the game constantly motivates me to proceed further to the next stage or level.					
6	I find my skills gradually improve through the course of overcoming the challenges in the game.					

TABLE III: Game Behavior

#	Statement	1	2	3	4	5
1	The re-implemented game reproduces all core gameplay mechanics accurately.					
2	Visual effects, animations, and timing are consistent with the original game.					
3	Sound effects and background music match the tone and timing of the original.					
4	All levels, menus, and progression systems function as expected.					
5	Gameplay outcomes (scores, physics, interactions) are consistent with the original.					

TABLE IV: Performance

#	Statement	1	2	3	4	5
1	The game maintains a smooth and stable frame rate during normal play.					
2	Load times are comparable to or faster than the original version.					
3	The game runs efficiently without overusing CPU or GPU resources.					
4	Performance remains consistent during high-action or complex scenes.					
5	Animations and transitions appear without lag or stuttering.					

TABLE V: Stability

#	Statement	1	2	3	4	5
1	The game does not crash or freeze during play sessions.					
2	Memory usage remains stable over time without leaks or excessive growth.					
3	The game recovers properly after being paused or minimized.					

Continued on next page

TABLE V: Stability (Continued)

#	Statement	1	2	3	4	5
4	Saved data remains intact and loads correctly between sessions.					
5	The game handles unexpected inputs without crashing.					

TABLE VI: Platform and Hardware Compatibility

#	Statement	1	2	3	4	5
1	The game runs correctly on the user's current operating system.					
2	Performance is consistent across different hardware configurations.					
3	Hardware inputs (e.g., mouse, keyboard) work properly on all supported platforms.					
4	The game performs without noticeable input lag on the computer.					
5	No platform-specific issues (e.g., driver or sound problems) were observed.					

REFERENCES

[1] N. D. Bowman and T. Wulf, "Nostalgia in video games," *Current Opinion in Psychology*, vol. 49, p. 101544, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352250X22002652

[2] P. Salvador, "Survey of the video game reissue market in the united states," 2023.

[3] C. Ponsard, "Meeting digital preservation requirements for software through an emulation strategy," *ICSOFT*, pp. 318–323, 2021.

[4] D. Guay-Bélanger, "Assembling auras: Towards a methodology for the preservation and study of video games as cultural heritage artefacts," *Games and Culture*, vol. 17, no. 5, pp. 659–678, 2022.

[5] M. Guttenbrunner, C. Becker, and A. Rauber, "Keeping the game alive: Evaluating strategies for the preservation of console video games," *International Journal of Digital Curation*, vol. 5, no. 1, pp. 64–90, 2010.

[6] J. McDonough, R. Olendorf, M. Kirschenbaum, K. M. Kraus, D. Reside, R. Donahue, A. Phelps, C. Egert, H. Lowood, S. Rojo *et al.*, "Preserving virtual worlds final report," 2010.

[7] M. Guttenbrunner, C. Kehrberg, A. Rauber, and C. Becker, "Evaluating strategies for the preservation of console video games," in *iPRES 2008*, 2008.

[8] M. A. Winget, "Videogame preservation and massively multiplayer online role-playing games: A review of the literature," *Journal of the American Society for Information Science and Technology*, vol. 62, no. 10, pp. 1869–1883, 2011.

[9] E. Kaltman, W. Schwaid-Lindner, D. Jonathan, A. Borman, A. Garnett, and L. Masinter, "An overview of emulation as a preservation method," 2025.

[10] Interkarma, "Daggerfall Unity 1.0 Release," 2023. [Online]. Available: https://www.dfworkshop.net/daggerfall-unity-1-0-release/

[11] R. Simpson, "Volcanic: an open source simantics ide," School of Computing Science, University of Glasgow, Glasgow, Level 4 Project, mar 2016.

[12] OpenTTD Team, "OpenTTD — About," 2025. [Online]. Available: https://www.openttd.org/about

[13] Reflexive Entertainment, "Press/news releases," 2006. [Online]. Available: https://web.archive.org/web/20060825213444/http://www.reflexive-inc.com/pressnews.shtml
[14] Ricochet Wiki, "Ricochet Xtreme," 2025. [Online]. Available: https://wiki.ricochetuniverse.com/wiki/Ricochet_Xtreme
[15] J. C. Smith, 2005. [Online]. Available: https://web.archive.org/web/20161031080816/http://forums.indiegamer.com/threads/building-an-editor-around-an-engine.2326/#post-29534
[16] M. Alrammal, M. Naveed, S. Sallam, and G. Tsaramirsis, "Malware analysis: Reverse engineering tools using santuko linux," *Materials Today: Proceedings*, vol. 60, pp. 1367–1378, 2022, International Conference on Latest Developments in Materials and Manufacturing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221478532106750X
[17] G. Canfora, M. Di Penta, and L. Cerulo, "Achievements and challenges in software reverse engineering," *Communications of the ACM*, vol. 54, no. 4, pp. 142–151, 2011.
[18] M. Ashawa, R. McGregor, N. P. Owoh, J. Osamor, and J. Adejoh, "Static and dynamic malware analysis using cyclegan data augmentation and deep learning techniques," *Applied Sciences*, vol. 15, no. 17, p. 9830, 2025.
[19] M. F. Ismael and K. H. Thanoon, "Investigation malware analysis depend on reverse engineering using idapro," in *2022 8th International Conference on Contemporary Information Technology and Mathematics (ICCITM)*. IEEE, 2022, pp. 227–231.
[20] B. Cui, F. Wang, Y. Hao, and L. Wang, "A taint based approach for automatic reverse engineering of gray-box file formats," *Soft Computing*, vol. 20, no. 9, pp. 3563–3578, 2016.
[21] Z. Lin, X. Zhang, and D. Xu, "Automatic reverse engineering of data structures from binary execution," in *Proceedings of the 11th Annual Information Security Symposium*, 2010, pp. 1–1.
[22] A. Reinhard, *Archaeogaming: An introduction to archaeology in and of video games*. Berghahn Books, 2018.
[23] radek sparta, "Awesome game remakes," 2025. [Online]. Available: https://github.com/radek-sprta/awesome-game-remakes
[24] D. Brčić, "Uloga arhiva u dugoročnom očuvanju napuštenog softvera s posebnim osvrtom na videoigre," Ph.D. dissertation, University of Zagreb, Faculty of Humanities and Social Sciences, Department of Information and Communication Sciences, 2025.
[25] ScummVM Wiki, "About – ScummVM," 2021. [Online]. Available: https://wiki.scummvm.org/index.php?title=About
[26] B. Dym, E. Simpson, O. Fong *et al.*, "The internet is not forever: Challenges and sustainability in video game archiving and preservation," *Journal of Electronic Gaming and Esports*, vol. 1, no. 1, 2023.
[27] M. Yong Wong, M. Landen, M. Antonakakis, D. M. Blough, E. M. Redmiles, and M. Ahamad, "An inside look into the practice of malware analysis," in *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, 2021, pp. 3053–3069.
[28] J. Aycock and T. Copplestone, "Entombed: An archaeological examination of an atari 2600 game," *The Art, Science, and Engineering of Programming*, vol. 3, no. 2, 2019.
[29] R. Bettivia, "Where does significance lie: Locating the significant properties of video games in preserving virtual worlds ii data," *International Journal of Digital Curation*, vol. 11, no. 1, pp. 17–32, 2016.
[30] Z. Gao, Y. Cui, H. Wang, S. Qin, Y. Wang, Z. Bolun, and C. Zhang, "Decompilebench: A comprehensive benchmark for evaluating decompilers in real-world scenarios," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 23 250–23 267.
[31] Y. Cao, R. Zhang, R. Liang, and K. Chen, "Evaluating the effectiveness of decompilers," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2024, pp. 491–502.
[32] M. Fereli, N. D. W. Cahyani, and M. Irsan, "File signature identification of flipper zero device to support file recovery," in *2025 International Conference on Information and Communication Technology (ICoICT)*. IEEE, 2025, pp. 1–6.
[33] J. Aycock and K. Biittner, "Experimental archaeogaming: A case study," *Advances in Archaeological Practice*, vol. 12, no. 2, pp. 75–85, 2024.
[34] "SDL3/FrontPage – SDL Wiki." [Online]. Available: https://wiki.libsdl.org/SDL3/FrontPage
[35] E. M. Gabriel and F. R. G. Lapitan, "Growing together: A co-operative farming game with ai agents using adaptive utility ai," Institute of Computer Science, University of the Philippines Los Baños, Technical Reports, 2025.
[36] J. E. G. Teknomo and F. R. G. Lapitan, "Soulbound: A networked persuasive game to empower mental health awareness and action," Institute of Computer Science, University of the Philippines Los Baños, Technical Reports, 2022.
[37] J. C. Smith, personal communication.
[38] ——, "Reflexive's "property system" and property editor," 2005. [Online]. Available: https://web.archive.org/web/20161002175938/http://forums.indiegamer.com/threads/reflexives-property-system-and-property-editor.2099/
[39] Xiph.Org, "Vorbis audio compression," 2016. [Online]. Available: https://www.xiph.org/vorbis/
[40] F. D. O. Fajardo, "Implement support for ZIP archives with Reflexive's custom EOCD signature," 2024. [Online]. Available: https://github.com/frankwilco/NuVelocity.SharpZipLib/commit/b24e445d15388bf08ae377d4fc1a6a64f787928b
[41] M. H. Phan, J. R. Keebler, and B. S. Chaparro, "The development and validation of the game user experience satisfaction scale (guess)," *Human factors*, vol. 58, no. 8, pp. 1217–1247, 2016.

Francis Dominic O. Fajardo is a BS Computer Science student at the University of the Philippines Los Baños. He was a member of the Junior Philippine Computer Society, where he served as the Director of Marketing and Promotions of one of its chapters from A.Y. 2021–2022. He has contributed to many free and open source software (FOSS) projects. His research interests include web browser development, information security, reverse engineering, game development, and FOSS.