Porting Unreal Engine 5 to the Nintendo Wii

Sam Collier
BSc Computer Games Technology

Faculty of Design, Informatics and Business
Abertay University
DUNDEE, DD1 1HG, UK

ABSTRACT

Context: Unreal Engine 5 is intended for modern platforms and has no official support for legacy hardware.

Aim: Evaluate the engine's limitations on Wii hardware by porting an existing Unreal Engine 5 game.

Method: Unreal Engine 5 will be compiled using a C++ compiler compatible with Wii hardware. The artifact will deliver platform code necessary for operating system functionality, graphics, and cooking assets for the Wii, plus any engine optimizations made throughout the project.

Results: Quantitative performance metrics will be collected from profiling tools in order to compare and comprehend frame times, hitches, and memory usage. Unit tests will ensure features are ported correctly and measure progress. Visual fidelity will be measured both qualitatively and quantitatively to evaluate rendering techniques.

Conclusion: This project will enable other game developers to optimize their Unreal Engine 5 projects, and provide a feasible solution to porting their projects to the Wii.

Keywords
Unreal Engine, Wii, PowerPC, Fixed Function Graphics.

1. INTRODUCTION

Unreal Engine 5 is the leading industry standard game engine developed and released by Epic Games (2022). The engine has become an essential learning requirement for professionals across all disciplines looking to enter the games industry. Recently, the gaming community has been fatigued by Unreal Engine 5 titles releasing with poor performance, and the community is quick to blame Unreal Engine 5 for these issues (Fleischauer, 2025). Some argue that the blame lies with the game developers misusing the engine, which is certainly true in many cases, however there may also be flaws in the engine's core philosophy which this project aims to investigate.

The industry has already adopted Unreal Engine 5. Studios would much rather hire artists, designers, and programmers familiar with Unreal Engine 5, and skip the production cost of onboarding developers to a proprietary game engine. It'll take a long time for the industry to shift away from using Unreal Engine 5 if it is indeed the root cause of bad performance in games.

1.1 Aim

This project aims to retrofit Unreal Engine 5 onto the Wii platform, so that programmers, artists, and designers familiar with Unreal Engine 5 can see their work running on the Wii. The Wii is nearly 20 years old, and will provide a harsh runtime environment for the engine with severe hardware limitations. This project will investigate Unreal Engine 5's source code and evaluate its performance on Wii hardware.

2. BACKGROUND

Unreal Engine 5's codebase began with Unreal Engine 4. It was purpose-built to exclusively support and take advantage of the PlayStation 4 and Xbox One (Meer, 2008). The Wii was released seven years prior to those consoles. There's a small handful of Wii games made in Unreal Engine 3. Unreal Engine 3 was a completely different codebase built for the PlayStation 3 and Xbox 360, both PowerPC consoles like the Wii. What sets them apart from the Wii is that they have programmable graphics pipelines. Various stages of the pipeline can be written by the game developer. While on the Wii, all stages of the pipeline are configurable, but not programmable.

2.1 A Modern Graphics API on top of a Legacy Graphics API

Unreal Engine 5 has its own rendering interface which acts as an abstraction layer on top of various supported graphics APIs. Programmable graphics pipelines are here to stay, and both Unreal Engine 4 and Unreal Engine 5 were built with that in mind. The rendering abstraction assumes a modern graphics API with a programmable pipeline. The engine's material editor generates HLSL shader code under the hood.

Various aspects of a modern graphics API must be emulated in order to maintain the engine's modern abstraction layer. As described by Konrad and Göbel (2023), certain aspects are entirely handled by the driver in legacy APIs, such as data synchronization between CPU and GPU, which will be trivial to abstract since there's no implementation necessary. However they also mention features such as multithreaded command submission which will require more CPU overhead to abstract because commands on a legacy API are not thread-safe and must be deferred instead. There's also modern API features such as hardware ray tracing that simply cannot be implemented.

2.2 Wii Architecture

The Wii makes use of a 729MHz PowerPC processor from IBM codenamed Broadway (Copetti, 2020). The PowerPC architecture was purpose-built for balancing “between maximum performance, ease of manufacturability, and low price” (Paap and Silha, 1993). PowerPC has a reduced instruction set (RISC) design which enabled more instructions per CPU cycle, and was the most performant paradigm at the time.

Copetti (2020) describes the rest of the system components which are stored inside a system on chip (SoC) codenamed Hollywood. For system memory it has 24 MBs of “Splash” 1T-SRAM and an additional 64 MBs of GDDR3-SDRAM available outside the SoC. The GPU, named Flipper, has a clock speed of 243 MHz and 3 MBs of 1T-SRAM for texture memory. The GPU uses a fixed-function pipeline with no support for shader programming. The SoC also houses various I/O components which are controlled by a second CPU named Starlet. Starlet adds virtual memory to the system and can virtually remap I/O to be what the original GameCube expects in order to maintain backwards-compatibility. Starlet also provides a layer of security by officiating access to critical components such as the system’s 512 MB NAND flash storage which stores the operating system and user data.

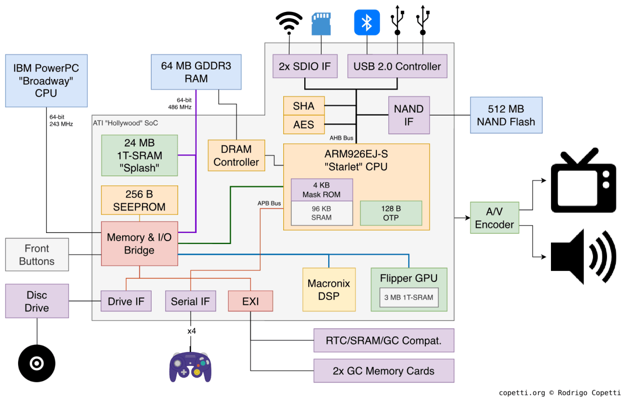

Figure 1 – Diagram of the Wii’s internal components (Copetti, 2020)

3. PROJECT ARTIFACT

This project will deliver core engine code necessary to port Left Upon Read by Triple Seven Studios (2024), a game built in Unreal Engine 5, to the Wii. Left Upon Read is a first-person action-adventure game which involves an eclectic mix of sword combat and cell phone texting. The game has a unique painterly art style which was achieved by a shader post-processing technique known as the Kuwahara filter exhibited in Figure 2.

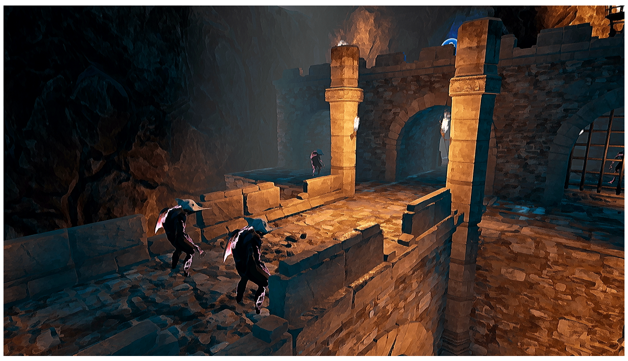

Figure 2 – Screen capture of Left Upon Read showcasing the Kuwahara filter (Triple Seven Studios, 2024)

The Wii port will need to faithfully keep to the game’s visual style, and select the most appropriate compromises between visual fidelity and performance. The game should also take advantage of the Wiimote’s button layout and motion sensors while also supporting a more traditional control scheme for GameCube controllers.

3.1 Adopting a new Compiler

Since the Wii is on a completely different architecture to Unreal Engine’s target platforms, a compiler capable of converting C++ source code to PowerPC CPU instructions must be used. Unreal Engine’s build system will need to be extended to use this compiler to build the engine for the Wii.

3.2 Core Engine Technology

The Wii has a very limited amount of system memory available, 88 megabytes in total. Unreal’s garbage collector and memory allocator systems will need to be adapted and better optimized to handle this limitation.

Traditionally, Wii games were published onto proprietary optical discs. However, the technology to reproduce these discs is not available to the public, so the game will be loaded onto a modern solid state drive connected via the Wii’s USB 2.0 interface. USB 2.0 will deliver significantly higher transfer speeds than an optical disc, but modifications to Unreal’s asset streaming system may be necessary if it relies on faster transfer speeds and causes stutters during gameplay.

An implementation of Unreal’s rendering interface will be made to map to the Wii’s graphics API. In the absence of shaders, creative uses of the Wii’s fixed-function graphics API will be required.

Unreal’s asset cooking pipeline will need to be extended to support the Wii. Cooking is the process of converting development assets to a format that can be packaged and shipped on a target platform. Assets will need to undergo an endianness conversion. PowerPC is big-endian which means the Wii reads bytes left to right instead of little-endian which is right to left on modern desktops and consoles. And since the Wii has a significantly smaller amount of memory, textures will need to be compressed into formats supported by the Wii. Geometry will also undergo optimizations during this stage to both take up less space, and perform faster for the GPU.

4. METHOD

4.1 Emulating Wii Hardware

Dolphin Emulator (no date) is a program capable of reliably emulating Wii hardware on other platforms. Dolphin provides several benefits to this project. It speeds up iteration times by removing the need to constantly flash changes onto an SD card to test a code change on the Wii. It can easily be containerized and deployed to continuous integration pipelines running on x64 Linux machines for immediate feedback on engine and game unit tests. Dolphin also has a debugger capable of breakpoints, stepping, loading symbols, exploring the callstack, and viewing CPU instructions. And a log window which can be used to view engine and game log messages. Dolphin will be a crucial tool for both developing the artifact and collecting results.

4.2 Unit Testing

Unreal Engine 5 includes a gauntlet of unit tests to verify various aspects of the engine work as expected on its supported platforms. The game code for Left Upon Read (Triple Seven Studios, 2024) also has a series of unit tests to ensure game bugs don't repeat themselves.

These tests will immediately pinpoint Wii-specific behaviors, bugs, and crashes. These results will accelerate the development of the artifact, but they can also be analyzed in many ways. Overall the tests can serve as quantitative results to exhibit how many engine features have been ported to the Wii. Collecting results for tests that pass, tests that could pass, and tests that will never pass, enables comparison between Unreal Engine's expected requirements and the Wii's capabilities in order to understand which Unreal Engine projects are feasible for the Wii and which are not.

4.3 Frame Profiling

There are several tasks that go into generating a frame in Unreal Engine 5. Knowing how long these tasks take is crucial for optimizing the engine. Profiling can also provide an opportunity for comparison against a modern x64 Windows machine. Obviously the actual timings cannot be compared directly, the Wii is almost two decades old. But ranking those timings from slowest to quickest to understand what percentage of the frame each statistic occupies is a comparable metric between both platforms. This comparison can highlight specific performance pain points on the Wii which may or may not be present on modern platforms. Profiling will also be used to evaluate the performance of legacy fixed-function workarounds.

4.4 Screen Captures

While developing a rendering back-end for the Wii, screen captures will aid as a qualitative metric to evaluate legacy fixed-function workarounds for the Wii. These captures will be collected through the use of Unreal's automation framework in order to reproduce the same captures. Visual fidelity comparisons can be made between the methods themselves and against a modern platform. Visual fidelity will be measured both from a still image and from a series of frames in motion. Visual evaluation is necessary in order to decide if a potential performance tradeoff of a technique is worthwhile. If necessary, these decisions can be validated with a small user study.

Quantitative methods for measuring visual fidelity such as the structural similarity (SSIM) index described by Wang and Bovik (2009) can be added to this project's automation tests to receive quick feedback on visual comparisons and aid qualitative assessment.

5. SUMMARY

Although the Wii is discontinued and no longer accepts new titles, it can still provide a useful performance benchmark for Unreal Engine 5 games of similar scope to Wii games. If a game can be optimized for the Wii, it is going to run well on a plethora of hardware configurations.

The Wii also has a large and passionate homebrew scene. This project would make Wii homebrewing more accessible to designers and artists who have less programming experience, and inspire more projects for the Wii.

Developers can also use this project as a reference on how to add a new game console to Unreal Engine. There are no existing examples available to the public due to non disclosure agreements on every supported console.

6. REFERENCES

Copetti, R. (2020) *Wii Architecture - A Practical Analysis*, *The Copetti site*. Available at: https://classic.copetti.org/writings/consoles/wii/ (Accessed: September 17, 2025).

"Dolphin" (no date). Available at: https://github.com/dolphin-emu/dolphin (Accessed: October 12, 2025).

Fleischauer, M. (2025) "Unreal Engine 5 Performance Issues Explored," *GameFromScratch.com*, 15 September. Available at: https://gamefromscratch.com/unreal-engine-5-performance-issues-explored/ (Accessed: October 12, 2025).

Konrad, R. and Göbel, S. (2023) "A Cross-Platform Graphics API Solution for Modern and Legacy Development Styles," in M. Haahr, A. Rojas-Salazar, and S. Göbel (eds.) *Serious Games*. Cham: Springer Nature Switzerland, pp. 50–62. Available at: https://doi.org/10.1007/978-3-031-44751-8_4.

Meer, A. (2008) "Unreal Engine 4 to 'exclusively target console,'" *Rock, Paper, Shotgun*, 13 March. Available at: https://www.rockpapershotgun.com/epic-unreal-engine-4-console-exclusive (Accessed: October 11, 2025).

Paap, G. and Silha, E. (1993) "PowerPC: a performance architecture," in *Digest of Papers. Compcon Spring. Digest of Papers. Compcon Spring*, pp. 104–108. Available at: https://doi.org/10.1109/CMPCON.1993.289645.

Triple Seven Studios (2024) "Left Upon Read." Available at: https://triple7studios.itch.io/left-upon-read (Accessed: October 11, 2025).

Epic Games (2022) "Unreal Engine 5". Available at: https://unrealengine.com (Accessed: October 14, 2025).

Wang, Z. and Bovik, A.C. (2009) "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Processing Magazine*, 26(1), pp. 98–117. Available at: https://doi.org/10.1109/MSP.2008.930649.