Nintendo Entertainment System Emulator

Tooling Selection and Justification

Kajetan Lach

December 10, 2024

1 Introduction

This document contains a detailed overview of the tools selection for the Nintendo Entertainment System emulator project. The selection process, justification for each tool and their contribution to the project's development are the topics covered.

2 Programming Languages

2.1 C++

C++ is a general-purpose programming language created as an extension of the C programming language. It is widely used for systems programming, embedded systems, and game development. Statically typed, compiled, multi-paradigm language that supports procedural, object-oriented, and generic programming. It is known for its performance, efficiency, and flexibility.

2.1.1 Purpose

Core implementation of the emulator. It will be used to implement the CPU, PPU, APU and other parts of the NES system.

2.1.2 Justification

C++ provides **high performance** and **low-level memory control**, which are critical for emulating hardware accurately. Its **rich ecosystem of libraries** supports tasks like file handling, input processing and graphics rendering.

2.1.3 Integration

C++ works seamlessly with graphics libraries like **SDL2** and **debugging tools**, providing a robust foundation for emulator development.

2.2 Python

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It is widely used for web development, data analysis, and automation. Dynamically typed, garbage-collected, multi-paradigm language that supports procedural, object-oriented, and functional programming.

2.2.1 Purpose

Scripting and automation for **testing** and **supplemental tooling**, for example, generating test data or analyzing ROMs.

2.2.2 Justification

Python's **ease of use** and **vast library ecosystem** make it ideal for quick automation tasks.

2.2.3 Integration

Python complements the C++ codebase by **handling non-performance-critical tasks** and providing a **flexible environment** for testing and analysis.

3 Frameworks and Libraries

3.1 SDL2 (Simple DirectMedia Layer)

SDL2 is a cross-platform development library designed to provide low-level access to audio, keyboard, mouse, and graphics hardware. It is widely used in game development, multimedia applications, and emulators.

3.1.1 Purpose

Graphics rendering, **input handling**, and **audio output** for the emulator.

3.1.2 Justification

SDL2 is **cross-platform**, making it easier to ensure portability across Windows, macOS and Linux. It is **lightweight**, yet **powerful**, providing the necessary capabilities for rendering NES graphics and processing inputs.

3.1.3 Integration

Integrates directly with C++ and serves as the interface for displayng emulated visuals and audio outputs.

3.2 Google Test

Google Test is a C++ testing framework developed by Google. It provides a rich set of assertion macros and test fixtures for writing unit tests.

3.2.1 Purpose

Unit testing the emulator's core components such as the CPU emulation logic.

3.2.2 Justification

Google Test is **mature** and **widely-used**, offering many features like assertions and parameterized tests.

3.2.3 Integration

Works with C++ to validate the accuracy of hardware emulation.

3.3 ImGui

ImGui is a C++ library for immediate mode graphical user interfaces. It is designed to be easy to integrate and use for creating debug tools and visualizations.

3.3.1 Purpose

Building the emulator's **graphical user interface**.

3.3.2 Justification

ImGui is **fast, portable and flexible**. It's ideal for building debugging interfaces or minimalistic GUIs for tools.

3.3.3 Integration

Directly integrates with **SDL2** for rendering and complements the emulator's debugging tools by providing real time visual feedback.

4 Development Tools

4.1 Visual Studio Code

Visual Studio Code is a lightweight, open-source code editor developed by Microsoft. It supports a wide range of programming languages and extensions.

4.1.1 Purpose

Main **Integrated Development Environment** for writing and debugging code.

4.1.2 Justification

VS Code is **lightweight, extensible** and provides a rich set of extensions for supporting both C++ and Python development.

4.1.3 Integration

Works well with CMake for build automation and integrates with Git for version control.

4.2 CMake

CMake is an open-source build system designed to build, test, and package software projects. It generates build files for various platforms and build tools.

4.2.1 Purpose

Build system and **project configuration** for the emulator.

4.2.2 Justification

CMake simplifies **cross-platform** builds, **automation** of generation of project files and **dependency management**.

4.2.3 Integration

Works seamlessly with compilers, IDEs and CI/CD systems to manage builds efficiently.

4.3 Git

Git is a distributed version control system used for tracking changes in source code during software development.

4.3.1 Purpose

Version control for source code and documentation.

4.3.2 Justification

Git provides a **reliable** way to track changes, versions, collaborate with others and revert code when necessary.

4.3.3 Integration

Integrates with Github for hosting remote repositories and managing project collaboration. Works well with VS Code for version control during development.

5 Project Management and Collaboration Tools

5.1 Github, Github Pages, Github Actions

Github is a web-based platform for hosting Git repositories and managing software development projects.

5.1.1 Purpose

Repository hosting and collaboration. **Github Pages** for hosting project documentation and additional educational resources for future users. **Github Actions** for CI/CD.

5.1.2 Justification

Github offers **issue tracking**, **pull request workflows**, **CI/CD** systems and **documentation hosting** in one platform.

5.1.3 Integration

Facilitates collaboration, code reviews and CI/CD integration making it a central hub for development.

5.2 Discord

Discord is a communication platform designed for communities, gaming, and team collaboration.

5.2.1 Purpose

Team communication and collaboration.

5.2.2 Justification

Discord provides **text chat**, **voice chat**, and **screen sharing** features, making it ideal for real-time communication and collaboration. It's free and easy to use.

5.2.3 Integration

Enables team members to communicate effectively, share progress, and discuss project-related topics.

6 Design and Documentation Tools

6.1 Draw.io

Draw.io is a free online diagramming tool for creating flowcharts, process diagrams, and other visual representations.

6.1.1 Purpose

Creating **system architecture diagrams** and **design documentation**.

6.1.2 Justification

Provides an **intuitive** interface for drawing UML diagrams, flowcharts and system overviews.

6.1.3 Integration

Enables the team to visualize and communicate the system architecture effectively.

6.2 LaTeX

LaTeX is a typesetting system used for creating high-quality documents, reports, and scientific papers.

6.2.1 Purpose

Writing **technical documentation** and **reports**.

6.2.2 Justification

LaTeX provides **professional** and **consistent** formatting for technical documents, making it ideal for writing detailed reports and documentation.

6.2.3 Integration

Works well with version control systems like Git and provides a clean, structured way to write technical documents.