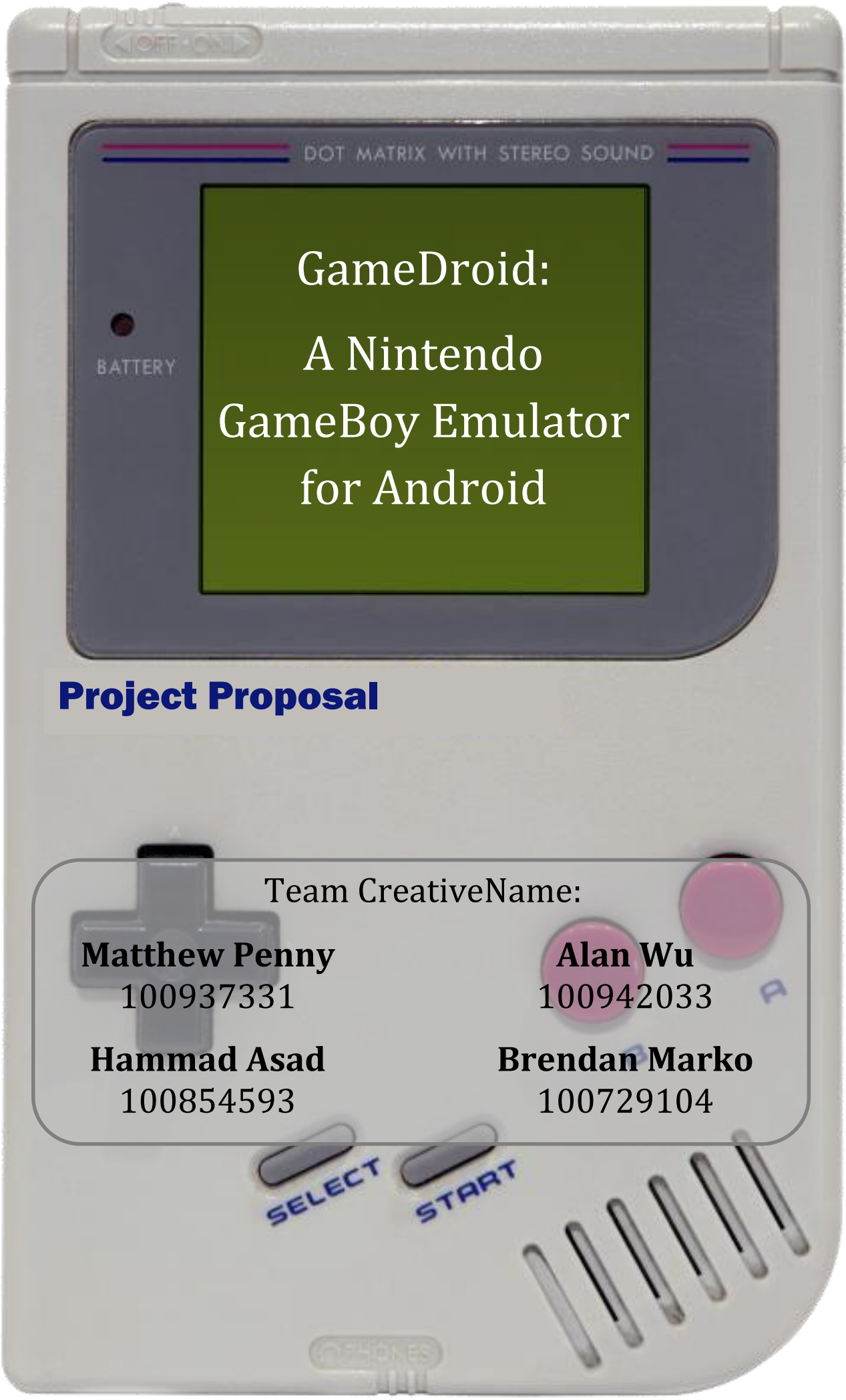

Project Proposal

Team CreativeName:

Matthew Penny
100937331

Alan Wu
100942033

Hammad Asad
100854593

Brendan Marko
100729104

Project Overview

Our project is a Nintendo GameBoy emulator that will provide users with the ability to play pre-packaged or user-provided ROM files on their mobile device. This will be a well-organized application based around simplicity in design, a tidy user interface, and some new custom features to make it unique. With the addition of rewinding gameplay and an intuitive save system, our emulator will enhance the user experience and distinguish itself from similar offerings.

This project is interesting from a development standpoint due to the challenge of properly interpreting game software data for old hardware and outputting accurately emulated gameplay using Android APIs. The application will be interesting for users due to the sense of nostalgia afforded by the ability to play games from an older generation on a mobile device. Additionally, the application will provide the ability to rewind games to a previous state (up to thirty seconds prior) in a manner similar to rewinding a VHS tape. This feature will create a more enjoyable gaming environment, easing the difficulty level and frustration of many notoriously hard games.

This emulator is well-suited to a mobile form factor due to the ease of accessibility of the application to users who might be looking to pass time while commuting or during short periods of downtime. An added benefit of mobile emulation over an original GameBoy is local storage of ROM files. This removes the need for physical cartridges and results in a compact, all-in-one gaming application. With the popularity of mobile devices and their associated app stores, updates can be applied easily to add new features or patches making post-development management and maintenance seamless and simple.

Functional Requirements

1.0 Emulation requirements

1.1 Memory layout emulation: Game code interacts with the various pieces of GameBoy hardware through memory-mapped I/O, and as such, the application must provide this communication mechanism in order for game software to function properly.

1.1.1	The virtual GameBoy must have at least 8KB of RAM.
1.1.2	The virtual GameBoy must have at least 8KB of video RAM (VRAM).
1.1.3	The emulated GameBoy's memory map must mimic that of original GameBoy hardware (i.e., emulated hardware mapped to the same memory addresses as the real thing).

1.2 CPU emulation: The application must emulate a Sharp LR35902 core @ at least 4.295454MHz. This is the CPU found in the Nintendo GameBoy and its emulation is required to run game code.

1.2.1	The emulated CPU must be able to interpret Sharp LR35902 machine code and update registers accordingly.
1.2.2	The emulated CPU must be able to handle interrupts.
1.2.3	The emulated CPU must be able to read and write to and from virtual GameBoy memory.

1.3 PPU emulation: The application must emulate the Nintendo GameBoy's proprietary PPU (picture processing unit). Emulation of this module is required for game graphics output.

1.3.1	The emulated PPU must be able to read and use the virtual GameBoy's memory to draw the correct game background tiles and sprites on the screen at the correct positions.
1.3.2	The emulated PPU must be able to read and use the virtual GameBoy's memory apply the correct palette to the sprite and background layers.
1.3.3	The emulated PPU must be able to render the contents of the virtual GameBoy's video memory at a resolution of at least 160x144 at the refresh rate of the original Nintendo GameBoy (59.73Hz).

1.4 Gamepad emulation: The application must emulate the Nintendo GameBoy's gamepad buttons (directional pad, A, B, start, and select). Emulation of this module is required to allow users to interact with running games.

1.4.1	The application must provide a virtual gamepad for users to interact with.
1.4.2	Emulated GameBoy game code must be able to poll the state of the virtual gamepad.

2.0 Game ROM file requirements

2.1 Game file parsing: The application must be able to correctly parse Nintendo GameBoy ROM files. These files contain game code/data and are required to play games.

2.1.1	The application must be able to parse Nintendo GameBoy ROM file headers and read the game metadata they contain.
2.1.2	The application must be able to use metadata read from ROM file headers to load the game data into memory and properly initialize the emulator state.

2.2 Game save functionality: Some GameBoy games are able to save progress. It is important to emulate this feature, or users will need to restart games from the beginning every time (which is very frustrating). If a ROM file's header contains metadata indicating it supports this feature the application must be able to detect when game code attempts to save progress and write that save file to disk.

3.0 Emulator enhancements

The emulator will have full access to the state information of the virtual GameBoy, allowing the implementation of desirable features which would not have been feasible on the original hardware.

3.1	The application must provide the ability for users to save and load the state of the emulator to/from disk at any point while playing a game (enabling "save anywhere" functionality).
3.2	The application must store at least 30 seconds of prior gameplay and provide a button to "rewind" emulation while depressed, using this data (allowing users to undo gameplay).

4.0 Emulator options

Since user devices and preferences vary, the application must provide a way of modifying emulation and application parameters (i.e., via a settings screen).

4.1	The application must allow users to upscale the resolution of emulated games.
4.2	The application must allow users to switch screen orientation.

5.0 Game library

Users may have multiple games on their device and a "library" screen can help curate them and facilitate quick selection/loading.

5.1	The application must provide a list of GameBoy ROM files previously loaded and allow users to select and load games to play.
5.2	For each game in the library, the application must allow users to manage emulator save states and game save files (if the game natively supports saving).
5.3	The application must allow users to add and remove games from the library from disk.
5.4	The application must allow users to mark games in their library as favorites (for finding them more easily).
5.5	The application must allow the library to be searched.

User Scenarios

Scenario 1.0: The user loads a game and starts gameplay

Tom has not played his GameBoy for years and wants to relive the experience on-the-go with his Android smartphone. Tom opens the GameDroid emulator. The game library screen for the emulator is displayed (**5.1**) and there are no ROMs available to choose from. Tom clicks on the '+' and a file manager screen pops up. The user then navigates to the directory of the ROM and chooses it. The ROM loaded is 'Pokemon Gold', which is now available in the game library screen (**5.3**); Tom clicks on it in the game library screen. Functional requirements **1.1**, **1.2**, and **1.3** come into effect and the screen changes to mimic a GameBoy. The game is loaded (**2.1**) and the gamepad is viewable and responsive (**1.4**). A 'Start Game' text appears on the Gameboy screen and Tom presses the 'Start' button until the gameplay begins.

Scenario 2.0: The user starts a game, saves the state and quits to the game library screen

Joe is on the bus and wants to occupy himself by playing a GameBoy game on his Android smartphone. Joe opens the GameDroid emulator. The game library screen for the emulator is displayed and the 'Pokemon Gold' ROM is available in the 'Recent' pane (**5.1**). Joe clicks on 'Pokemon Gold' and the game loads. Functional requirements **1.1**, **1.2**, **1.3** and **2.1** come into effect – the Gameboy view is now seen with the responsive gamepad (**1.4**). The game is loaded and now playable – Joe continues through the startup screens by pressing the 'Start' button. Joe continues playing the game for 10 minutes, until he realizes that the bus is arriving to his destination. Joe saves the current state of the game (**3.1**) by pressing the save icon so that the state can later be loaded. Afterwards, the menu button is pressed at the bottom left of the screen and the 'Quit' option is chosen to exit to the game library screen.

Non-functional Requirements

1.0 Performance:

The emulator should be able to operate at least as fast as the actual hardware. Processors in modern phone are orders of magnitude faster than the 8-bit processor in the GameBoy in terms of clock speed. Our software emulation should be able to match the performance of the original hardware even in the presence of the overhead of Android. Matching the performance of the GameBoy helps the emulator deliver an experience that is true to the original.

2.0 Efficiency:

The emulator should be efficient in terms of memory usage. The original hardware has 8kB of RAM and video RAM, and supports ROM up to 1MB. While it's unrealistic to expect the memory usage of the emulator to be the sum of these sizes, the emulator should not be memory intensive. Keeping the RAM usage reasonable for the emulator leaves space for other apps running on the Android device during usage of the emulator. This helps with keeping app switching quick even on devices that have a low amount of physical RAM.

Mockups

The virtual gamepad will be displayed directly below the video output, mimicking the layout of an original Nintendo GameBoy. This allows users who are familiar with the GameBoy to more easily use the application.

Conventional icons for the menu and rewind buttons are used (see bottom-left and bottom-right, respectively) so that users will immediately know their function.

The game library screen will allow users to easily organize their GameBoy game ROM files. The library will be able to list recently played games, games marked as favorites, and also show games in alphabetical order. This will ensure that users will always be able to find what they want to play quickly.

Since game ROM files are small (typically on the order of kilobytes) it is not unreasonable to believe that users will add many games to their library. Providing a search function allows users to quickly locate a given game even if it is one they have not played recently (and regardless of the size of the game library).