GameDroid: A GameBoy Emulator for Android

COMP 3004 – Deliverable 4

Team CreativeName:

Matthew Penny (100937331)
Alan Wu (100942033)
Hammad Asad (100854593)
Brendan Marko (100729104)

1.0 – Status report

The GameDroid application is fully functional at the time of writing. All requirements proposed at the start of the project have been met, both functional and non-functional. The project is also fairly polished, given the lack of public testing. This report will detail the completion status of requirements listed in the original proposal, and outline our final demo. The weekly development logs can be found at https://github.com/hammadasad/comp3004/tree/master/dev_logs and can be accessed using a Github account with appropriate credentials. The promotional video can be found at https://www.youtube.com/watch?v=SpdBk_0kcU4.

GameBoy emulation is the core of this project. As a quantitative measure, GameDroid passes all instruction and interrupt test suites written by the emulator development community. These test suites are widely used as benchmarks for the accuracy of emulators. Passing these tests provides strong evidence that GameDroid meets the requirement for CPU emulation. Many popular GameBoy emulators on the Google Play store do not pass all of these tests. As a qualitative measure, GamDroid is able to accurately emulate most GameBoy titles such as Pokemon Blue/Red/Yellow, Super Mario Land, Tetris and Ninja Gaiden Shadow. This is evidence that other components of emulation are also in working order. Although this evidence is not as strong as passing trusted test suites, since the end goal of the emulation requirements is to allow users to emulate games, it is reasonable to conclude that they have been met.

As for game ROM parsing, GameDroid is able to display the metadata of a given GameBoy ROM. This metadata includes game title, publisher, and region of origin. This information can be accessed by long pressing a ROM entry in the library menu. Games which supported saving on the original hardware are also able to save when emulated inside GameDroid. An example of such a title is Pokemon. A user is able save in-game while playing Pokemon, restart GameDroid, and then load that save in-game.

Gamedroid includes extra emulation-related functionalities not present on the original hardware that we think can improve a player's experience. GameDroid offers screen up-scaling and has a layout for playing in landscape mode, as promised in the original proposal. GameDroid's "save anywhere" functionality is also fully implemented. A user can hit the save-state button at any point during emulation and later use the load-state button to go back to that point. Loading a state means going back to a point in time that happened during emulation. A "state" in this context refers to a composition of all relevant states of the original hardware. As an extension of this capability, the "time rewind" feature is also fully implemented. Holding the rewind button causes the gameplay that happened just before the press to replay in reverse, and releasing the button causes GameDroid to resume emulating from the state corresponding to what is displayed on screen. This feature allows players to retry difficult portions of games without the requirement of remembering to hit the save-state button. For gaining an intuitive understanding of this functionality, the promotional video contains many clips of it being used to good effects.

GameBoy emulators on the Google Play store are often lackluster in terms of user-friendliness. Building a pleasing UI with great game library management functionalities was a major goal for this project. Using Android's material design, GameDroid has a look that is consistent with other modern applications. Since the first screen of the app is the list of most recently played games, a user is able to start playing with a single click. ROMs can be marked as favorites to help locate them, and the whole library can be searched. Users can also delete save files, saved states, and ROM files by long pressing on entries in the library. To add games to the library, a user simply need to put the ROM files at a specific path on the device's file system. This should be easy to do since transferring files to and from the device is a task that almost every smartphone and tablet owner knows by heart.

GameDroid's two non-functional requirements are good emulation performance and reasonable memory usage. Performance wise, GameDroid is able to emulate at full speed on the HTC One M9. This phone is a mid-range device from early 2015, which indicates that GameDroid can run at a good speed on many devices that are on the market today. Unfortunately, GameDroid is not able to run at full speed on devices that are less powerful. During development, we overestimated the speed of modern smart phones. Picking Java as the language of choice, unfortunately, prevents us from using optimization techniques present in lower level languages such as C++. The unnecessary overhead introduced by polymorphism slows down CPU emulation significantly on less powerful devices such as the Nexus 4. Despite this, it's still fair to say that GameDroid meets the performance non-functional requirements, since a large number of devices of reasonable prices are capable of running GameDroid at full speed. Memory usage wise, GameDroid performs respectably. During emulation, memory usage of the app ranges from 70MB to 120MB. This memory usage figure is common among popular Android apps. Twitter, for example, uses the same amount of memory.

In conclusion, the GameDroid project was successful in meeting all of its requirements and offering a great retro-gaming experience to its users. With accurate emulation, the ability to save-state and rewind, and a pleasing UI GameDroid is a feature-full emulator. As a team, we are proud that we were able to complete this ambitious project in the given amount of time and in the face of many difficult technical challenges. The team worked well together, helping out when anyone was stuck, and kept in good, frequent communication via Slack and weekly meetings. GameDroid will be kept in our memories long after COMP 3004 is over.

2.0 – Demo description

Since the last demo, we have been focusing on application usability and implementing the last few features outlined in the proposal. We will first present the new feature that was assigned to us, which is a help section. Next, we will run through a typical use case for the average user. This entails using the game library to find a game to play (e.g., using the "favorites" tab). Once the game is started, the user will demonstrate the new usability features of our main emulator view. These include the application automatically prompting the user to load a save file, confirmation to load and save, and the landscape UI. After playing for a short while to demonstrate full-speed emulation (non-functional requirement 1.0) the user will make a mistake and use the rewind feature to undo their error and try again (functional requirement 3.2). They will then exit the game (showing the newly added prompt to confirm the exit) and use the ROM info dialog in the library to delete a save file (functional requirements 5.2 and 5.3). Finally, the user will enter the settings menu, change the screen size, and play another game to demonstrate user-configurable settings (functional requirement 4).

As we present the application and show its features we will discuss technical limitations that had to be overcome. These include emulation speed and Java shortcomings. Everything is fully implemented. Nothing is simulated.