**Release 4**

**What’s Changed?**

**Emulator:**

The Emulator has now been fully completed and thoroughly tested for bugs and errors. With the Emulator, users can choose from a selection of ready-made Chip-8 games and load them into the Emulator. Users are then able to play their selected games in the Emulator and can change the current Chip-8 game to a new, different game. The Emulator is also well-suited for students, programmers, and other enthusiasts. It provides a couple of tools to assist users, who might be interested in learning how Chip-8 works or how the programs run.

**Visualizer:**

The Emulator has an in-built Visualizer installed. Users are able to see most of the Emulator's key variables, including the values in the registers, the memory, and the timers. In addition, users are able to speed up or pause a Chip-8 game at any time. While paused, users can advance the program one opcode at a time, and more advanced users will be able to follow along by looking at the Visualizer. In this way, users can keep track and see how the program works.

**Tools:**

A tool that we made allows advance users to write lines of opcodes in the text box. By clicking “Compile and Run!”, users can compile and see their program run live on the display.

**Games:**

We made two games for our Chip-8 Emulator.

***Duel***

Duel is a simple, competitive shooting game for two players. The game starts with a scoreboard in the middle of the display and two cowboy characters, one on each side of the screen. These cowboys are staring down at each other. At the start of each round, a bell will ring once after a random time interval has passed. At the sound of the bell, the two players will compete against each other to see who is the first to fire his/her gun. If a player shoots before the alarm, he/she will receive a penalty(the other player gets a point). The character on the left side of the screen is “Player 1.” His/her shooting key is “Q” on the keyboard. The character on the right side of the screen is “Player 2.” His/her shooting key is “R.” The player who shoots first wins the match and earns a point. The first player to score 5 points wins the game.

***Snake***

This is the classic game Snake. When the player starts the game, the player can control a snake using the “E,” “S,” “D,” “F” keys. Food will appear on the screen. When the snake eats the food, the snake will become longer, and food will appear in new position. The player will lose if the snake runs into itself or into the borders of the screen.

**What’s Next?**

This is the final release. We have no more updates planned for this project, and all aspects and features of this project are in their final states. We are done!

**Instructions for Running the Emulator**

***To load games and play,***

1. Double-click on “Chip8.html” (Firefox works the best because other browsers like chrome have difficulty loading some of our games)
2. Click on “Browse” to choose the desired game to play from your local hard drive
3. Click on “Load and Run!” button
4. Wait for the game to load
5. Enjoy
6. *Press F12 to open the web console to see the Emulator’s details*

***To pause or speed up the current running program,***

1. Press the tilde key to pause/resume the Chip-8 program. Alternatively, click on the "Pause" button
2. When paused, press F2 to execute the next opcode in order to advance the program line by line. Alternatively, click on the “>” button
3. If not paused, click on “Run Faster!” in order to make the Emulator run faster. It is only able to run up to four times as fast as the normal speed
4. Close the web browser or the tab in order to stop the Emulator

***To run the mini-compiler,***

1. Type some opcodes (for example, 00E0, 600A, etc.) into the textbox, one opcode per line
2. Click on “Compile and Run!” in order to run the opcodes.
3. The Emulator will compile and run the set of instructions given by the opcodes.

**Instructions for Automated Testing**

1. Double-click on “Chip8.html”.
2. Click on the “Run Automated Testing” button.
3. See the results in the newly opened tab.

**Project Post Mortem**

**What Didn’t Worked Well**

When we first planned our deadlines at the beginning of the semester, we were too optimistic. As we will find out later on in the semester, the factors and variables that were not taken into consideration caused delays and some missed deadlines. Learning from the experience, we realized it is important to take into account as many variables and factors as possible when planning deadlines. Our team should lay out the variables, and come up with ideas on how to handle each situations. Should the unfortunate occurs, the team will be prepared to handle the situation.

We also planned to implement the “Agile Method” for the development of the project, where features are implemented simultaneously and improved overtime. However, as we begin working on the project, we shifted more and more towards the “Waterfall Method” of development where features are implemented one after another. Since the scale of the project and team size is small, our team found that it is easier and more efficient to focus our energy on one important feature at a time. When we used the “Agile Method” at the beginning, it was hard for us to understand what the each team members are working on. Insufficient knowledge of the topic and lack of quality communication between team members can also contribute to this confusion. When we started to shift towards the “Waterfall Method”, we were forced to be more cooperative and competent. Since our team focused on one feature at a time, it required us to be better communicators in order to coordinate with each other. Using the “Waterfall Method” meant that we often were working and researching about the same topics. As a result, it allowed us to share ideas and knowledge with each other, making us more competent as a team. Seeing the improvements and advantages of the “Waterfall Method”, we decided to adopt the methodology for the development of our project.

**What Worked Well**

One thing that worked really well for the team is our group meetings on Fridays. These meetings serves a great purpose in keeping the team members updated on each others progress. We sit together and talk about what we did in the previous week and what we are going to get done in the coming week. The meetings kept us accountable for our progresses individually. When we are stuck or faced with unexpected problems, this is where we come together and brainstorm ideas and solutions as team. No technology will be able to replace face to face human interaction and communication. Talking to each other in person improves the effectiveness of communication and it also helps build team synergy.

**Lessons and Insights**

Although we have made consistent progresses throughout the semester and we have handled problems and unexpected situations very well, we feel that there is always room for improvement. One of the things we could have done better was planning deadlines. Taking into consideration some of the unexpected situations and human conditions would have helped a great deal in reducing stress and anxiety on release dates. Our automated testing component was not completed in time for release 1 was the result of the underestimation of the effort needed to complete the emulator and overestimation of the resource we have towards completing the automated testing component. This could have been avoided if these factors were taken into account in our initial planning phase.

Having the experience and knowledge we have now, we have one good advice for future classes. We advise that each team member to be flexible and adaptable. When thing are not going according to plan, members getting sick, or a member missing in action, it is important for each individual in the team to adjust and act according to the situation. Rather it is having to step up and take the lead or taking up the extra responsibility, flexible and adaptive individuals make a sturdy team that can ride out to any storm.