CMPT276 - Team 1’s Product Document - Release 4

*Instructions for automated tests, chip-8 usage,compiler usage, and tool usage are located in INSTRUCTIONS.md*

Administrative Details:

* We plan to have meetings every Wednesday from 12:30 - 2:30 after class, and are all also available Monday from 12:30 - 2:30 for an emergency meeting if need be.
* We will be using Slack for our online communication method, with separate channels on Slack for each part of the project to keep discussions organized.
* We have a Github repository setup for our version control system.
* The Emulator, Visualizer, and Tool (a Chip8 Sprite-Editor) will be created in Javascript, while the two games (Pong and Tetris) will be created using Assembly Language.

Role Breakdown:

Emulator: Josh T. & Adam

Visualizer: Adam

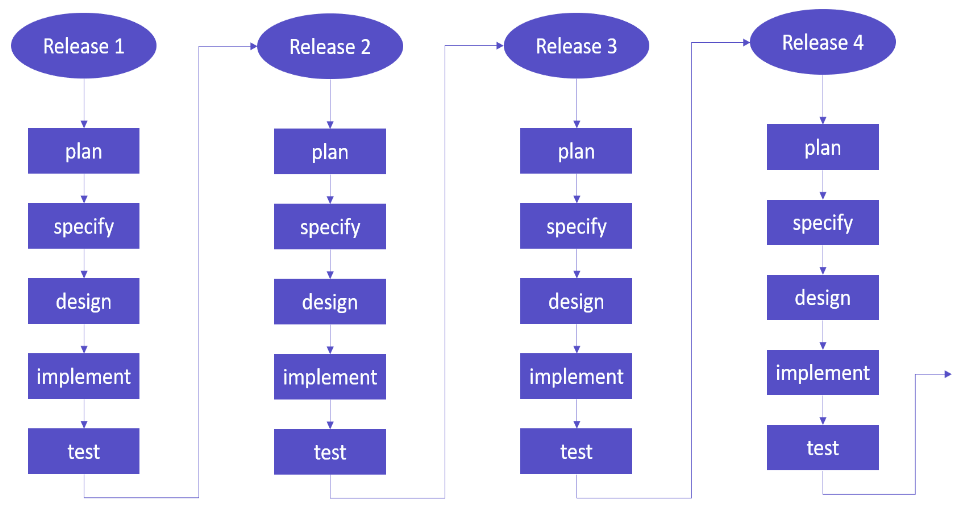
Tool: Sarb

Compiler : Sarb

Game 1: Josh H. & Sarb

Game 2: Brandon & Sarb

Software Methodology:



* We will be using an Incremental Development model.
* The planning and specifying phase will occur during meetings, to ensure that we are aware of and understand the requirements for the next release.
* The designing phase will occur in subgroups with those working on the emulator and visualizer in one group, and with those working on the game and tool in another group.
* Implementation will be largely done individually with the help of groupmates.
* The testing phase will mostly be done by unit testing, with one large automated test about a week before release.

Testing & Quality Assurance

* Instructions to run automated tests are in INSTRUCTIONS.MD
* Unit Testing:
  + For every function that we write for the emulator, visualizer, and tool, we will create a separate function to test that the original function does what it is intended to do, and add the test function to an automated testing file.
* System Testing:
  + Use system testing for simpler features such as the tool.
* Acceptance Testing:
  + Having peers test the emulator by playing the game later on in the development near releases 3 and 4

Detailed Use Cases for Release 5 (for the sections that could use it)

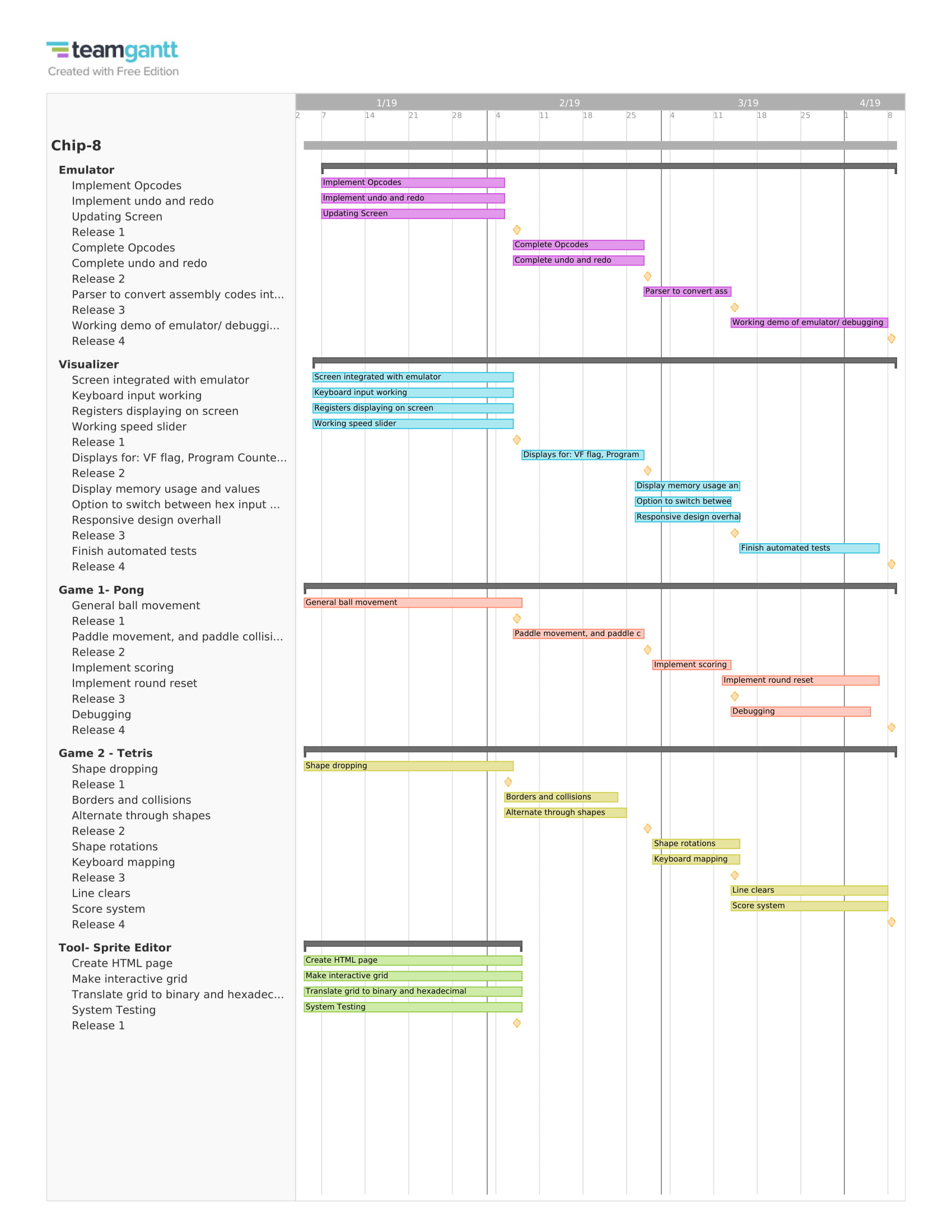
* Game 1: Pong
  + Ball moves differently based on where it hits the paddle to add variation to gameplay
  + Improve the visuals of the game, such that it looks less empty
* Game 2: Tetris
  + Add customized size of playing area
  + Add a power-up which allow player to choose which block comes next

Detailed Work Breakdown

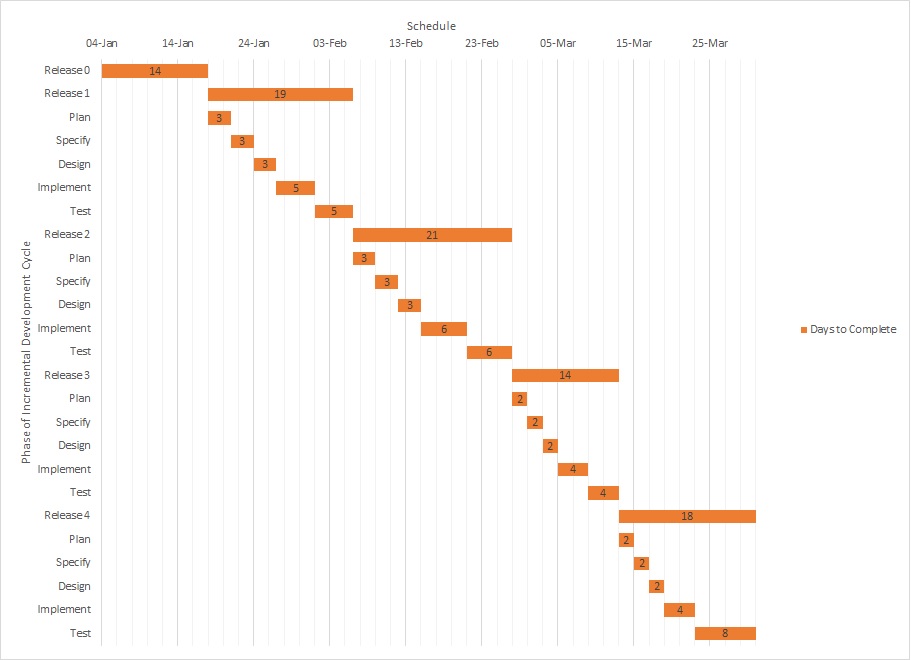
* Emulator (Total estimate: 60 hours (split roughly 5 hours/week over 12 weeks))
  + Release 1 (est. 20 hours)
    - A buggy working emulator.
  + Release 2 (est 15 hours)
    - Expect to have most of the bugs fixed from a lot of testings.
  + Release 3 (est. 15 hours)
    - Focus on polishing and optimizing the emulator.
  + Release 4 (est. 10 hours)
    - Fully functional emulator.
* Visualizer (Total estimate: 24 hours (split roughly 2 hours/week over 12 weeks))
  + Release 1(est. 8 hours)
    - All the registers and memory displayed alongside the chip-8 screen.
  + Release 2 (est. 6 hours)
    - Option to load program from hex or from mnemonic.
    - File selection button to load programs from file
  + Release 3 (est. 5 hours)
    - Controls menu to customize input keys.
  + Release 4 (est. 5 hours)
    - Rigorously tested and debugged Visualizer.
* Game 1 - Pong (Total estimate: 60 hours (split roughly 5 hours/week over 12 weeks))
  + Release 1 (est. 10 hours):
    - The ball is drawn to the screen, and is able to bounce off the walls and not go “out-of-bounds”
  + Release 2 (est. 15 hours):
    - The user is able move the paddles through keyboard input
    - Ball has proper collision with the paddle (bounces off).
  + Release 3 (est. 20 hours):
    - Implement round resets whenever the ball gets past a paddle
  + Release 4 (est. 15 hours):
    - Implement scoring system
    - Additional Testing and Optimization
  + Release 5
    - Use Binary Coded Decimal to move all the scoring into one register
    - Improve the overall appearance of the game
    - Have the ball behave differently based on where it hits the paddles to add more variation
* Game 2 - Tetris (Total estimate: 70 hours (split roughly 5 hours/week over 12 weeks))
* Release 1 (est. 20 hours)
  + Basic shape movement/physics
  + Collisions between borders and blocks
* Release 2 (est. 20 hours)
  + Rotating through different shapes (when spawned)
  + Stacking shapes
  + Shape movements side to side (user input)
* Release 3 (est. 20 hours)
  + Shape rotations
    - Shape “instant drop” feature
* Release 4 (est. 10 hours)
  + Scoring system
    - Line clears
  + Release 5 (est. 10 hours)
    - Customizable size of playing area
    - Power-ups
* Compiler - (Total estimate: 50 hours (split roughly 5 hours/week over 8 weeks))
  + Release 2 (est. 20 hours)
    - Webpage setup and file loading
  + Release 3 (est. 20 hours)
    - Implement compilation for most common opcodes
  + Release 4 (est. 10 hours)
    - Implement compilation for all opcodes
    - Debug and improve error handling
* Release 5 (est 20 hours)
  + - Implemented syntax error handler
    - Debugged sprite compilation
* Tool - Sprite Editor (est. 15 hours to complete)
* Release 1 (est. 20 hours)
  + Interacting with grid to change boxes in order to create a sprite
  + Reset the grid to create another sprite
  + System Testing
  + Release 2
    - Translate button working. Displays the sprite translated into binary hexadecimal
    - Ignoring empty lines of sprite

Detailed Work Breakdown:

We implemented the detailed work breakdown in a gantt chart. JPG image is provided or access the pdf with this link: <https://drive.google.com/file/d/1NmXA6pGnwslELsDF75-gQvDHEhW4qD5N/view?usp=sharing>



Detailed Schedule (with due dates)

* The detailed work breakdown shows what specific pieces of the project will be done for each release, the following schedule breaks down each release based on our software methodology (incremental development), to plan how we will complete each part:
* For each release, we have scheduled out how much time will be spent on the Plan, Specify, Design, Implement, and Test aspects of the Incremental Development Cycle.

Changes since Release 3:

* Adjusted testing and quality assurance section. Decided to work in JavaScript instead of Jest
* Specified detailed use cases for release 5 for each major component if there is something to be added.
* Adjusted detailed work breakdown section
  + For each major component of the project (Emulator, visualizer, game 1: pong, game 2: tetris, tool):
    - Added a release 5 section if there is something that is to be implemented.
* Update section “Features Completed/Not Completed for Release 4” for all major components (Emulator, visualizer, game 1: pong, game 2: tetris, tool)
* Integrated compiler into the emulator.
* Updated INSTRUCTIONS.md as well as the word documents in the games file.
* Got rid of unnecessary files such as the duplicate compiler folder.

Features Completed/Not Completed for Release 4

* Emulator
  + Features specified for release 4 that are completed:
    - Fixed VF flag
    - Fixed key input
    - Improved framerate
    - New testing
* Visualizer
  + No other features to add, only general debugging and visual improvements
* Game 1 - Pong
  + Features specified for release 4 that are completed:
    - Scoring System was implemented
    - Optimizations were made to improve performance
  + Unplanned features for release 4 that are completed:
    - None
  + Features specified for release 4 that were not completed:
    - None
* Game 2 - Tetris
  + Features specified for release 4 are complete and include:
    - Scoring system
    - Line clears
  + Unplanned features for release 4 that are completed:
    - None
  + Features specified for release 4 that were not completed:
    - None
* Compiler
  + Features specified for release 4 are complete and include:
    - Debugging sprite compilation
    - Integrating compiler into emulator
    - Improved assembly syntax error handling
  + Features planned for release 4 that are not completed:
    - Extensive testing with more games
* A major feature not completed for release 4 is getting our Tetris to work properly on our emulator. Reason: Emulator/Visualizer may have missed some test cases that need to succeed in order for Tetris to be run properly.

Project Post-Mortem Discussion:

* What worked well?

Our team decided that there were four main points that allowed us to work well on this project.

* + The first one is the breakdown of workload. Due to understanding each other’s weaknesses and strengths, we were able to break down our workload effectively. For example, those who had experience working in JavaScript would work on the emulator, visualizer, and tool and those without would work on the games.
  + Due to how we broke down the work, this lead to our project becoming fairly modular. This helped us in the long run as none of us were interfering with each other’s code. Thus there was no case of conflicting versions of the project, which was expected to happen initially.
  + We also learned the Chip8 system fairly quickly due to us learning and teaching each other about it. This left us more time for the developmental phase of the project.
  + Our team felt that because we prioritized communication, that allowed us to understand the responsibilities that comes with each role and how it affects the other ones. This also kept us in check. If any of us were having trouble, it would be fine as long as we voiced our complications as we could help.
* What didn’t work well?
  + Understanding of system components. Since our team worked somewhat independently, we did not know how about the details in the other components of the project. This led to some confusion for those who worked on the emulator and visualizer about how a certain opcode worked in the games.
  + Testing. Due to our inexperience with automated testing, we thought that using our own unit testing and system testing was enough. However, when it came to transition from the external emulator over to our own emulator, we found many problems that could not be easily fixed.
  + Time management. We had trouble in the early stages of our project estimating how long components would take, as well as how much time we would have to spend on this project relative to our other courses and commitments.
* What would you have done differently knowing what you know now?
  + Our team would have definitely focused much more on testing, whether it is system testing, acceptance testing, or especially automated testing.
  + Plan for the unexpected to happen. During the early stages of our project, things seemed to be going smooth, but later on, we realized there was a lot of work that needed to be done that we had not accounted for in our initial planning.
* What was the biggest unexpected challenge?
  + Time management was more difficult than expected in many ways. One way that it was difficult was estimating how long each component would take to be completed. Early on in the project, we had thought the emulator would be fully done somewhere between release one or release two. While we had made good progress on the emulator by then, there were many bugs that we had not yet found. This is in part due to the second way that time management was difficult for us, which would be managing our other commitments while working on this project. Many of us had midterms or other assignments either due close to release dates, or just constantly throughout the semester. This caused our progress on this project to be slower than we had initially thought, which required us to reevaluate our timeline for the project.

* Was all the testing worth it?
  + Unit testing proved to be extremely helpful as it is necessary to make sure everything was working, whether it was on the emulator side or the gaming side.
  + System testing was definitely worth the time, as it allowed us to find bugs in our emulator that would have been difficult to detect had they not been tested using one of our games (for instance).

* What advice would you give to the next students who take 276?

Our team has four pieces of advice to give to future 276 students.

* Everyone should work on the emulator at the start. This will allow for a much better understanding of how the Chip8 system works and serve as a good foundation when moving on the making the games.
* Be consistent with references. Since there are many references on the Chip8 system online, it could possibly be difficult to settle on one implementation.
* Keep your system modular. This will ensure that a problem will be contained within its own section. If you do not keep the system modular and end up trying to fix a bug, it could be found in any line of code in any of the files.
* Start system testing earlier. This will allow for the debugging of the emulator to be fairly easier for those working on it.