CMPT276 - Team 1’s Product Document - Release 3

*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

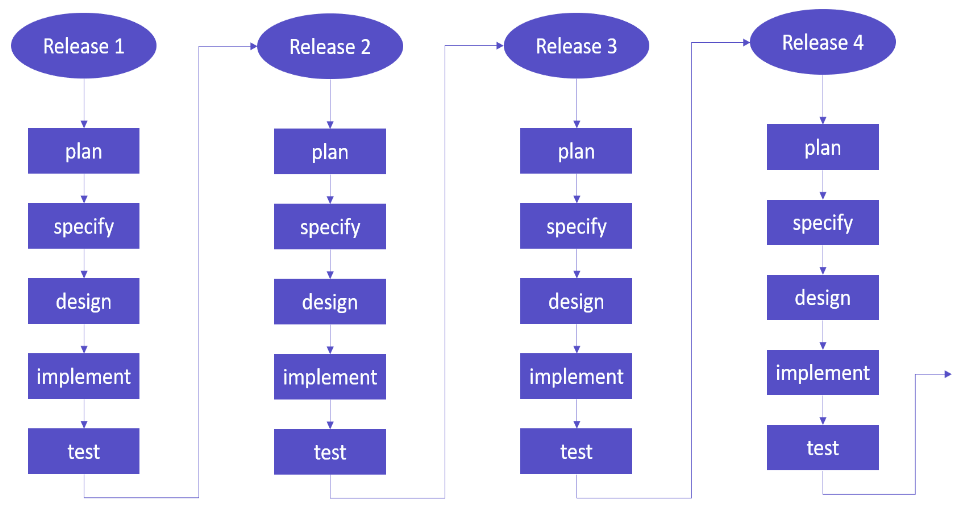
Tool: Sarb

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 4

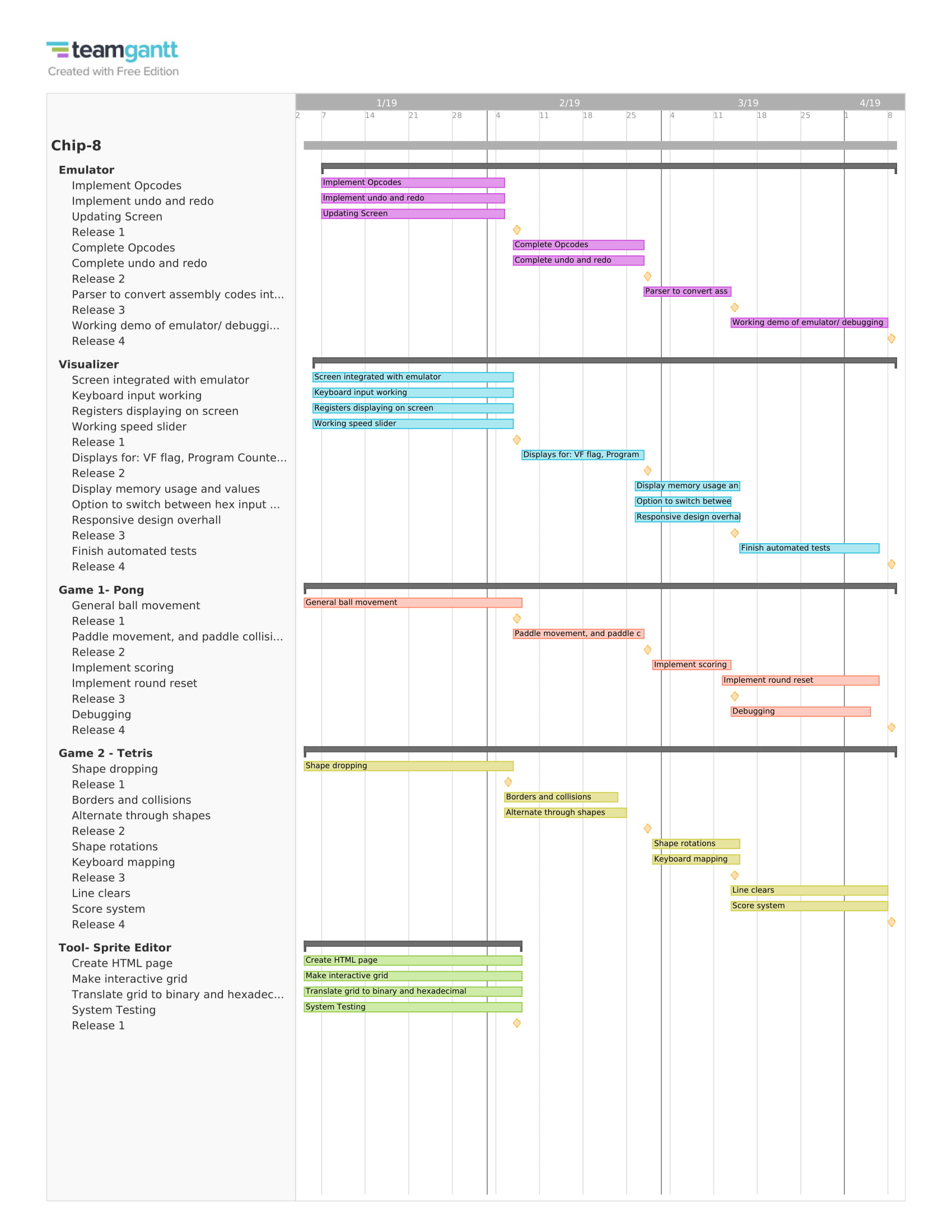
* Emulator
  + Automated testing showing all instructions working properly when turn on the program.
  + Fix all the bugs in the emulator.
  + Improve emulator performance.
  + Can import a binary file for the emulator to translate file to hexadecimal, result is displayed inside the text box.
  + Codes are put into the emulator text field, written in hexadecimal, work correctly.
  + Compiler can translate assembly to hexadecimal correctly.
* Visualizer
  + Release 4 introduces legacy mode for instructions: 8xy6 and 8xyE
  + The visualizer is for anyone who wants to analyze memory management, usually for those writing a Chip8 program or ROM
  + Text based input that integrates with emulator
  + UI for playing, pausing, stepping forwards, and stepping backwards in the code
  + File importing
  + Remapped keyboard input for convenience
  + Program execution speed control
  + Foreground and background colour options
* Game 1: Pong
  + For release 4, the game will be able to keep track of points
  + For release 4, the game will be completely finished, with 2 users able to game of pong against each other
* Game 2: Tetris
  + Rotation of blocks by the use of a key press
  + Full line detection
  + Line deletions
* Compiler
  + Compile function calls, sprites, and jumps
  + Compile most common opcodes
  + Load files from computer
  + Output compilation to text area
* Tool
  + Interact with grid to change boxes in order to create a sprite
  + Reset the grid to create another sprite
  + Translate button working. Displays the sprite translated into binary and hexadecimal.

Detailed Work Breakdown

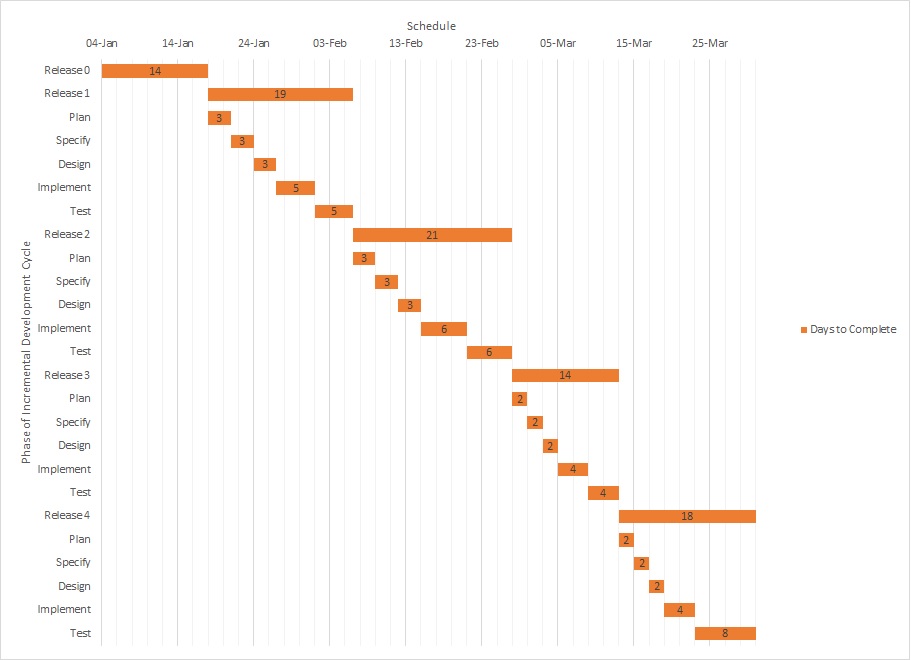
* Emulator (Total estimate: 60 hours (split roughly 5 hours/week over 12 weeks))
  + Release 1 (est. 20 hours)
    - A buggy working emulator.
  + Release 2 (est 15 hours)
    - Expect to have most of the bugs fixed from a lot of testings.
  + Release 3 (est. 15 hours)
    - Focus on polishing and optimizing 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
* 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
* Compiler - (Total estimate: 40 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
* 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 2:

* Adjusted testing and quality assurance section. Decided to work in JavaScript instead of Jest
* Specified detailed use cases for release 4 for each major component.
* Adjusted detailed work breakdown section
  + For each major component of the project (Emulator, visualizer, game 1: pong, game 2: tetris, tool):
    - Add details about what will be completed for each release
    - Add time estimates for how long it will take to complete each part of the releases
  + For Tetris:
    - Moved line clears to release 4
    - Moved “instant drop” feature to release 3
  + For Pong:
    - Moved scoring system from release 3 to release 4
* Update section “Features Completed/Not Completed for Release 3” for all major components (Emulator, visualizer, game 1: pong, game 2: tetris, tool)
* Added CHIP8 language compiler as part of project

Features Completed/Not Completed for Release 3

* Emulator
  + Features specified up to release 3 that are completed:
    - Implement all the instructions.
    - Implement undo and redo features.
    - Implement screen function features.
    - Provide automated testing for most instructions except the ones involved in pressing key.
    - Provide automated testing for default fonts from 0 to F.
    - Have almost all instructions working.
    - Provide Legacy mode for shifting instructions.
  + Planned features not completed for release 3:
    - Verify all instructions working properly.
    - Automated testing for all instructions.
    - Good emulator framerate.
    - Player control in game.
* Visualizer
  + Planned features not completed for Release 3:
    - Toggle between hex and assembly input

**Reason**: Assembly compiler not ready for integration yet

* + No other features to add, only general debugging and visual improvements
* Game 1 - Pong
  + Features specified for release 3 that are completed:
    - Round resets when the ball goes past a paddle
  + Unplanned features for release 3 that are completed:
    - None
  + Features specified for release 3 that were not completed:
    - Scoring system
* Game 2 - Tetris
  + Features specified for release 3 are complete and include:
    - Shape rotations
    - “Instant” drop feature
  + Features planned for release 3 that are not completed:
    - Line clears - moved to release 4
* Compiler
  + Features specified for release 3 are complete and include:
    - HTML page with text area and import button
    - File loading working
    - Compilation code for address specific opcodes like JP, CALL, ect
    - Compilation code for opcodes used in gameplay
    - Error catching for invalid opcodes
  + Features planned for release 2 that are not completed:
    - Complete code for all opcodes
    - Error handling with accurate line specifier