CS 106
Muir, Thomass
Instructor Kevin Harrigan
04/09/20

CS106 Final Project Write-Up

Description: What is your project?

My project is the classic puzzle game of *Sudoku. Sudoku* is a logic-based, number-placement puzzle. The objective of the game is to fill a 9x9 grid board with digits so that every 3x3 box (subgrid), column, and row in the grid contain all of the digits from 1 - 9. Each digit can only appear once in each subdivision (column, row, or box). My program is able to auto-solve and regenerate new *Sudoku* boards. Additionally, my program gives players the option to save and load their game states (current grids) so that they can enjoy the game at their own leisure.

Inspiration: What was your inspiration?

Growing up, my dad really loved to play Sudoku. He would carry a book of Sudoku puzzles around and we would work on them together while travelling. Since my dad loves to see my work and is always asking me to make things for him, I decided that Sudoku would be the perfect project to attempt!

Instructions: Explain how to run your sketch, and how to use its features.

The game opens on the splash screen. The player can choose to immediately play, read the rules, or read the about page (which lists my contact information).

The classic *Sudoku* game involves a grid of 81 squares. The grid is divided into nine blocks, each containing nine squares. The rules of the game are simple: Each of the nine blocks has to contain the numbers 1 to 9 within its squares. Each number can only appear once in a row, column or box. Players can select their numbers by pressing any one of the 9 number buttons in the side panel. They can also select the “erase” button in order to erase any mistakes they may have (but not any pre-generated numbers on the grid). If a player so chooses, they can also just replace the number directly without erasing it.

To give the players a hint, incorrect numbers will be red when a player places them on an incorrect square. To return back to the “home” (splash) screen, players can click the house icon or the “return to home” button in the side plan. To auto-complete the puzzle, players can press the auto complete button in the side panel. When a player finishes the puzzle or auto-completes it, they can then press the New Puzzle button to generate a new puzzle! Using the buttons beneath the canvas, players can save their game states and load it back in whenever they are ready to play again.

Topics: List which of the topics from the menu above you used in your sketch.

1. JSON Files - In order to load and save the board states, I created a JSON object that was stored within the gameStorageJSON variable. When the “Save Button” was pressed, I then set the values

of the JSON properties to those of the current game state arrays. This included the current board (current state), initial board (so that players could still erase their mistakes), and the solved board (for auto-completion and game-over purposes). When the player presses “Load Button”, the loadJSON() function is called and the states are restored to the stored game state.

2. Containers - I used the built-in array functions splice() and includes(), as well as the dot functions indexOf() and substring() in order to locate and remove possible “Number Candidates” for each row and column. I also used the built in array function push() to add (push) different elements to the end of various arrays throughout the program.

3. User Interfaces and the DOM - In order to implement the “Load” and “Save” buttons, I used HTML DOM element objects to represent the HTML button elements. If the player clicks the “Load” button, the loadData() function is called. Similarly, if the player clicks the “Save” button, the saveData() function is called.

4. Advanced Shapes: I used the createVector() function to create a variety of two dimensional vectors. I used these vectors in order to keep track of different grid (x, y) positions. I used the function in the findLowestCandidates(), drawGame(), and selectedSquare() functions.

5. Randomness and Noise: I used the random() function in order to choose random squares that will become empty for the player to fill out. I also used the random() function in order to randomly select squares on the board to fill out with viable “Number Candidates”.

6. Topic 10 (Extension) - For my project, I used recursion to generate the *Sudoku* boards. I did this by creating a recursive function called fillBoard(). I used recursion to repeatedly fill in different boxes in the grid until the grid was full. When one square was filled, the function would be called again and the process would repeat. Moreover, if there was an error at any time throughout the process, my array would be reset and the fillBoard() function would be called again.

   Another aspect of my project that I expanded on was the communication with users, in which I utilized the alert() and confirm() functions in order to display warnings, messages, and confirmation messages to the user.

   I also used the createFileInput() function to enable users to select local game state files. This was used in combination with JSON objects and my HTML button elements.

   Finally, I used try/catch statements in order to test the code for errors and throw exceptions if needed in order for the game not to crash.