

CS 101 Project

SRS Document

Introduction :

Sudoku is a puzzle game in which a 9x9 grid with a few numbers already filled in is given to the user and the user is expected to solve it by filling in all numbers from 1 to 9 in every row, column and certain 3x3 squares. Because of the dimensions of the grid, no number is repeated along a row, column or 3x3 square.

	7	1	6			9		
4						3		1
			8		2		7	
	5					4		
2			1		5			3
		3					9	
	1		2		4			
6		8						2
		4			9	1	5	

A Sample Sudoku highlighting the rows and columns and the 3x3 box.

Work has been divided as-

1) Code - Hersh Manek

Ayush Lakhotia

Harshit Agrawal

Ajeet Singh

2)Graphics – Alekhya Audi

Ruhee D'Cunha

Archita Dungdung

Problem statement :

a) Giving the user a solvable sudoku puzzle having a unique solution, checking his gameplay against the rules of conventional sudoku and finally checking the correctness of the solution. Calculation of the player's score will be done. High scores of each level of difficulty will be displayed.

b) Solving a sudoku puzzle given by the user.

Approach :

In the second part of the program, the user is given a blank 9x9 grid. He can input any number of values at any positions. The validity of the input values will be checked according to the rules of Sudoku and the grid will be solved by the program. In the first part, we will store the sudoku puzzles in files according to the level of difficulty. And according to the level of difficulty chosen by the user, a random puzzle from the files will be given to him. The player will be allowed to input values (to play the game) and the program checks whether the user is playing according to the rules. Hints will be available to the user when required. The final answer is verified with the already solved Sudoku. Score will be calculated by taking into account the time taken to solve the puzzle. The score will be checked against the high scores after every game and the high scores will be altered accordingly.

Functional Specifications :

We are working on a Sudoku game with two aspects: Part one, the game itself, and Part two, a customizable solver. Initially the user will have to choose between the two modes of the game.

Function used- choicewindow

The window will have two buttons, one to play the game and one to solve a custom grid.

Part 1: Sudoku Game

A window opens when the user starts the game.

Function used - menuwindow

It has various buttons for New Game, Leaderboard, How to Play, Exit.

In New Game, one more frame appears which asks the user to choose a level and give his username.

Function used - checkerwindow

Depending on the level the user chooses, the program will display one of the Sudoku grids already generated for that difficulty. There will be a 9X9 Sudoku grid in the frame of the game. There will also be tabs for hints, 'Clear Screen', which will restart the game and for starting a new game. The user will be able to click on each cell individually, choose a digit between 1 and 9 using buttons at the bottom of the screen and automatically check its validity i.e., whether that digit repeats in its column, row or 3X3 box. If he puts in a number that isn't valid, that number together with the number it is clashing with will be highlighted. Thus the game continues.

	5					4		
2			1		5			3
		3				3	9	

Clashing Digits in the same row and 3x3 box

A set of hints will also be provided.

Function used - hint

If the user clicks on a cell and then clicks on the hints tab he will be given the digit that appears in that cell.

We will include the concept of high scores.

Function used - clock

We will start a digital clock and keep time. Depending upon the time and number of hints the user has requested, the score will be calculated and the name of the user together with the score will be displayed in the High Scores window.

Part 2: Sudoku Solver

In Part 2, a blank 9X9 grid is given to the user.

Function used - solverwindow

The user can input as many values as he wants. There will be a 'Finished' tab which he will click when he is done inputting. The completed grid will then be displayed.

Description Of Data (Input/Output):

Part 1:

Input: User input will mainly be in the form of mouse clicks, with very few keyboard responses recorded.

Output: The grid will be shown onscreen throughout the game, with numbers shown as they are filled in.

Part 2:

Input: User input will be the sudoku he wants solved.

Output: The solved Sudoku.

User Interface Requirements:

The basic aim is to display windows, buttons, the grid and to record mouse clicks for input and output.

Interfaces to other systems:

We will be using the EzWindows graphics library.

Hours Worked by each member :

Alekhya Audi - 111030021 - 16 hrs
Ruhee D'cunha - 111030005 - 12 hrs
Archita Dungdung - 111030019 - 13
Hersh Manek - 111030003 - 16 hrs
Harshit Agrawal - 111030013 - 12 hrs
Ayush Lakhota - 111030012 - 13.5 hrs
Ajeet Singh - 111030027 - 8 hrs

Appendix:

Part 1: The Game Sudoku

This is quite straightforward with the user solving a Sudoku.

Logic-

We just need to check whether the value the user input is in the same column, row or corresponding 3x3 box.

Part 2: Sudoku Solver

Logic-

The user can input as many values as he wants. We will store this in a file. We are using brute force mechanism to solve this Sudoku.

In the very start, we first check if the values given by the user follow the rules of Sudoku. Cells are numbered from 0-80, like in a 2D array in C++. In this mechanism, if we take a cell and it is empty, we put a value in it starting from 1 and going upto 9. Using a function, we check its legality whether it follows the rules of conventional Sudoku. If there is already some input given by the user, we move to the next cell. If 1 is not suitable, then we increment it and try with 2 and check the legality again. We then move on to the next cell. If there is no legal possibility (none of the values from 1-9 are possible), then we go the previous cell and if it contains a user input then go to its previous cell and increment the value stored in it by 1 and again check the legality. This continues till we get a solution for all the empty cells. We have to be certain that we don't change the user input while solving.