# **Group assignment: Puzzle game sudoku with game engine**

Group members

Ethan Drury: 23006382

Nicole Jang

Toby Gladding

Siddharth Gandhimathinathan,

# **Sudoku Game - Development Documentation**

## **Project Overview**

A classic Sudoku puzzle game built in Java using a custom game engine, featuring a clean visual design with multiple different backgrounds, in depth controls and sprites, and audio feedback to enhance the player experience.

## **Design Decisions**

A basic 9 by 9 grids with a 3 by 3 sub grid(Simple sudoku setup).

### **Grid Mathematics**

The game uses a 9x9 grid system with several mathematical calculations:

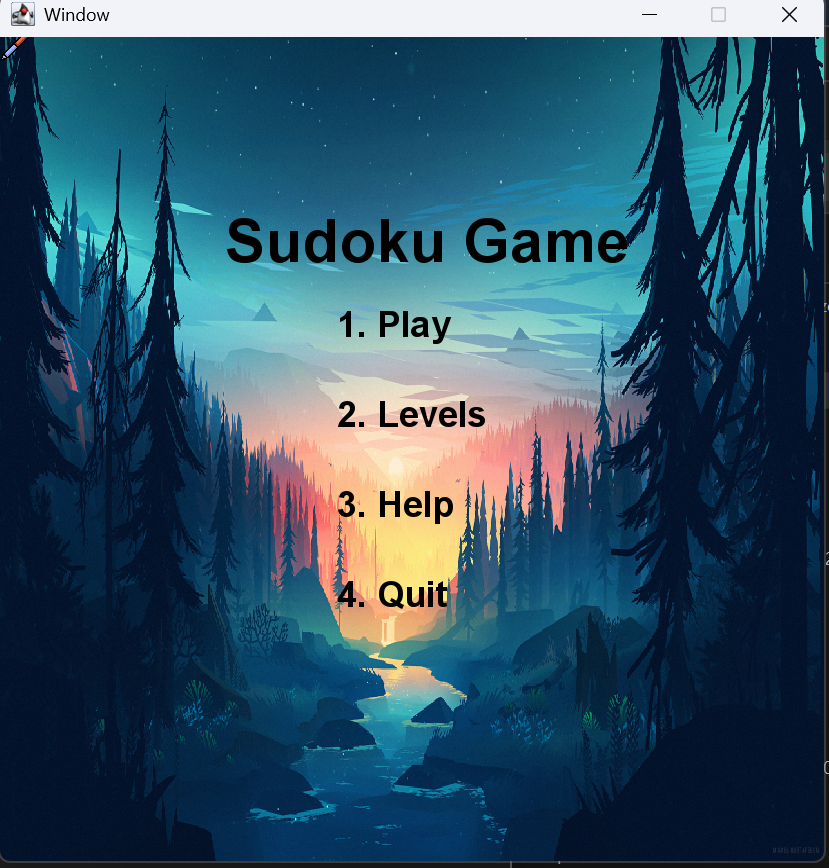
* **Cell Positioning**: cellX = boardOriginX + col \* CELL\_SIZE\_PIXELS
* **Mouse-to-Grid Conversion**: gridPos = (mousePos - BOARD\_PADDING) / CELL\_SIZE\_PIXELS
* **Subgrid Calculations**: subgridStart = (position / SUBGRID\_SIZE) \* SUBGRID\_SIZE
* **Boundary Detection**: Ensures clicks fall within valid grid coordinates (0-8)

### **Input:**

* Mouse Selection: Click any cell to select it
* Keyboard Input:

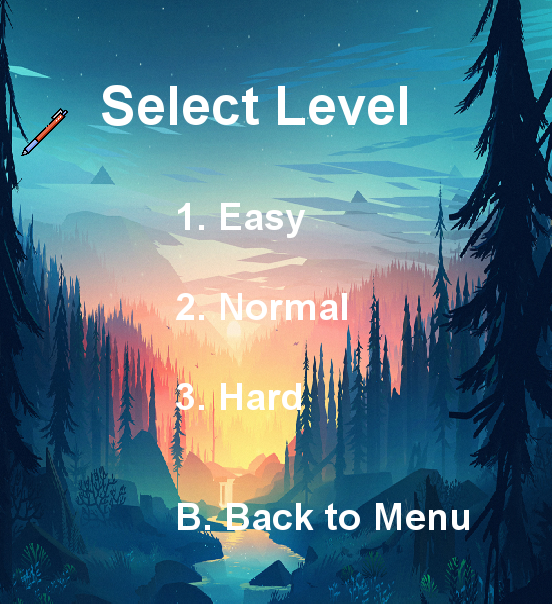
1. Numbers 1-9 to place values
2. 0/Backspace/Delete to clear cells
3. R to reset the puzzle
4. H for hints (highlights solvable cells)
5. ESC to clear selection
6. M to toggle background music
7. Press B to go back to menu
8. P allows you to return to menu when in a level.

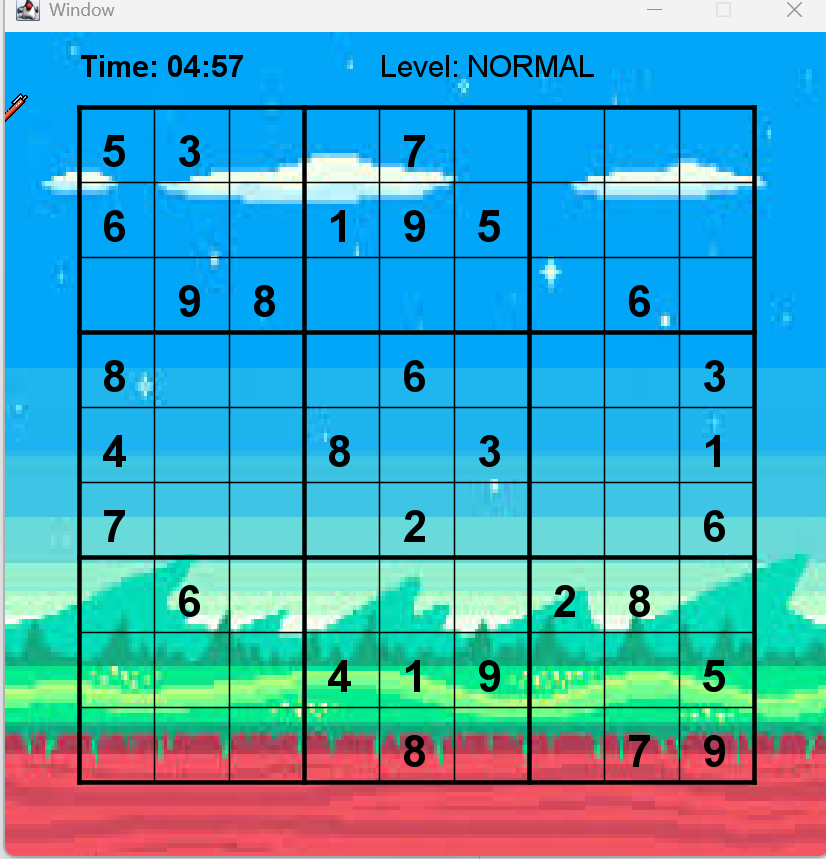
### 



**Menu:** Four options are provided: Play, Levels, Help, and Quit. Play just starts a game of sudoku at the standard level. Levels provide the alternatives of easy, hard, and normal. Help describes the basic game concepts, and quitting will cause the application to close.

**Game state:** When the timer starts in the game, you must complete the challenge before it reaches zero, otherwise you will have to restart (With R).

**Levels:** Our Sudoku game has three difficulty levels, which correspond to the number of initial numbers: easy, normal, and hard. Even using game features like hints is more difficult on the various levels because there could be more alternatives. Easy shows about two thirds of the board, whereas hard shows less than a third at the start.



**Scoring:** You must complete the puzzle in under five minutes and every wrong answer removes 5 seconds from the clock. When the game is complete you will se how long it took you to complete the puzzle you can either restart with r or press p to choose a different level.

## **Development Challenges**

### **1. Visual Feedback System**

**Challenge**: Initially, the game prevented players from entering invalid numbers, which was frustrating as users couldn't see what they were doing wrong.

**Solution**: Modified the code to allow incorrect entries but display them in red.

### **2. Grid Line Rendering**

**Challenge**: Ensure that selection highlights and hints did not obscure the grid lines, particularly the thicker subgrid bounds.

**Solution**:CELL\_BORDER\_OFFSET is used to draw highlights slightly offset from cell boundaries.

**3. Hint System Algorithm**

**Challenge**: Developing a hint system that finds cells with only one valid value without revealing too much.

**Solution**:Created the canSolveCell() method, which evaluates all possible values (1-9) for empty cells and returns true only when exactly one number is correct.

### **4. Audio Integration**

**Challenge**: Using context appropriate sound effects to enhance rather than distract from gameplay.

**Solution**:

* Select sound plays for valid moves.
* Error sound for rule errors
* Victory sound plays once upon finishing.
* Background music with a toggle control (M key).

### **5. Coordinate System Management**

**Challenge**: Translating pixel coordinates to grid positions while keeping the code tidy and clear.

**Solution**: Created dedicated conversion methods (getRowFromPixel, getColumnFromPixel) and used constants consistently for all sizing computations.

## **Technical Features**

### **Visual Design**

* **Color Coding**:

1. Black: Original clues (bold font)
2. Blue: Valid user input
3. Red: Invalid user input (rule violations)
4. Pale Yellow: Selected cell highlight
5. Light Purple: Hint highlight
6. Green: Victory flash effect

**Sprites/Animations:** A pen with a clicking animation to give feedback to the user when inputting answers into the grid.

### **Game Features**

* Real time constraint validation for rows, columns, and 3x3 sub grids
* Intelligent hint system that identifies uniquely solvable cells
* Victory detection with visual celebration (Victory screech)
* Timer countdown changes color when the clock is almost zero.

## **Future Enhancements**

* Puzzle generation algorithm
* Save/load game state

## **Work done**

### **Toby -** Handled the majority of the front-end work, including considerable amount of work into the implementation of the menu, help, and play choices, as well as some level design. Making the project seem appealing by determining values for the lines and offsets while importing a majority of the key’s code functions.

**Ethan**- Implemented the majority of the main game elements, including the hint function and timer, as well as the conflict code.

### 

### **Development Notes**

Achieves the specifications set.

**Final Answers for normal difficulty puzzles.**

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

**Final Answers for Hard difficulty puzzle.**

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