Sudoku Generator

Thanks for purchasing Sudoku Generator by ExaGames!

Forget about maintaining databases of fixed puzzles in your Sudoku games. ExaGames' Sudoku Generator is an algorithm capable of creating up to 6,670,903,752,021,072,936,960 unique puzzles. Your players will never play the same puzzle twice!

Overview

1. Import Sudoku Generator to your project.
2. Create a new SudokuBoard: *SudokuBoard sudokuBoard = new SudokuBoard();*
3. Get a random puzzle from your board: *int[,] puzzle = sudokuBoard.GetPuzzle();*
4. Let the user play.
5. Compare user inputs against the board.

Detailed Instructions

Create a new SudokuBoard

Initialize a new SudokuBoard instance anywhere in your code (for example, on Start() or some "New Game" button handler) using this code:

```
SudokuBoard sudokuBoard = new SudokuBoard();
```

You may want to add ExaGames.SudokuGenerator namespace to your using list on top of your script.

Your newly created board is stored in the *Board* property of the *sudokuBoard* object as a 9x9 integer matrix.

Get a random puzzle from your board

Puzzles are SudokuBoards with empty fields for the player to fill. Each puzzle contains just three values pre-filled for each row, column and square.

Puzzles are also generated as 9x9 integer matrices, with zero's in hidden fields.

For example: for a board like this…

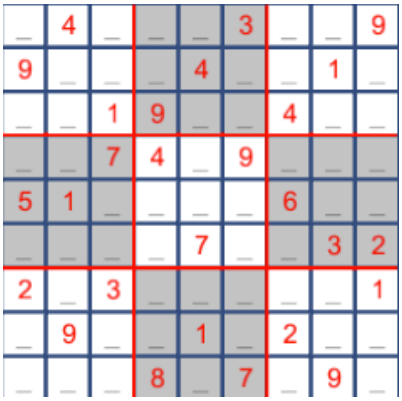
Sudoku Board

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

```
sudokuBoard.Board = { // int[9,9]
{6, 4, 2, 1, 5, 3, 7, 8, 9},
{9, 3, 8, 7, 4, 2, 5, 1, 6},
{7, 5, 1, 9, 8, 6, 4, 2, 3},
{3, 2, 7, 4, 6, 9, 1, 5, 8},
{5, 1, 9, 2, 3, 8, 6, 7, 4},
{4, 8, 6, 5, 7, 1, 9, 3, 2},
{2, 7, 3, 6, 9, 5, 8, 4, 1},
{8, 9, 5, 3, 1, 4, 2, 6, 7},
{1, 6, 4, 8, 2, 7, 3, 9, 5}
}
```

… a puzzle can be something like this:

Puzzle



```
sudokuBoard.GetPuzzle() = { // int[9,9]
{0, 4, 0, 0, 0, 3, 0, 0, 9},
{9, 0, 0, 0, 4, 0, 0, 1, 0},
{0, 0, 1, 9, 0, 0, 4, 0, 0},
{0, 0, 7, 4, 0, 9, 0, 0, 0},
{5, 1, 0, 0, 0, 0, 6, 0, 0},
{0, 0, 0, 0, 7, 0, 0, 3, 2},
{2, 0, 3, 0, 0, 0, 0, 0, 1},
{0, 9, 0, 0, 1, 0, 2, 0, 0},
{0, 0, 0, 8, 0, 7, 0, 9, 0}
}
```

You create puzzles for your board calling sudokuBoard.GetPuzzle(), which returns a 9x9 int matrix with your puzzle.

```
int[,] puzzle = sudokuBoard.GetPuzzle();
```

Every call to GetPuzzle will create a different puzzle for the same board. To change the board, simply call new SudokuBoard() again.

Compare user inputs against the board

How you implement this step depends on your specific game implementation; the most common way to do this is with a nested for loop.

For example, suppose you stored your user inputs directly in the *puzzle* matrix declared in the above step. You can check if the puzzle is solved with a method like this:

```
public void CheckSolution() {
      bool solved = true;
      for(int i=0; i<9; i++) {
            for(int j = 0; j<9; j++) {
                  if(puzzle[i][j] != sudokuBoard.Board[i, j]) {
                        solved = false;
                        break;
                  }
            }
            if(!solved)
                  break;
      }
      if(solved) {
            // The puzzle is solved! Congratulate the player!
      }  else {
            // There are some errors, let the player keep playing...
      }
}
```

Demo scene

This package includes a demo scene you can use as starting point for your development. The *GameController* script recreates the above instructions, and it's the only thing you need to start creating random Sudoku boards right away.

When deploying your game, you can safely remove the ExaGames/SudokuGenerator/Demo folder to save some disk space if you don't need it.

One more thing…

Check out other great assets at the ExaGames site in the Asset Store.

And don't forget to visit our website to stay up to date with ExaGames' new assets and games. You may also find some free games to play! Go to:

www.exagames-studio.com

Thank you from the ExaGames team!