

The grid is generated by code.

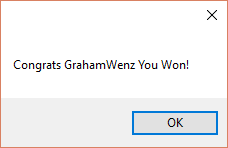
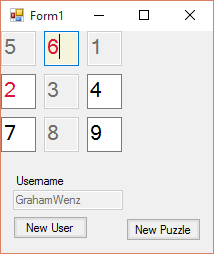
Each grid textbox only allows integers and backspaces.

Boxes with pre-generated numbers are unchangeable

Focused TextBox turns Beige;

Saving UserName gets rid of trailing spaces before and after the name

Entering a duplicate number will cause a warning message when Solve is clicked, then the textbox containing the error is selected



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Sudoku

{

public partial class Form1 : Form

{

string userName = "";

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

createBoxes();

newPuzzle();

}

private void btnNew\_Click(object sender, EventArgs e)

{

btnNew.Visible = false;

btnNew.Enabled = false;

btnSolve.Enabled = true;

btnSolve.Visible = true;

newPuzzle();

}

private void btnSolve\_Click(object sender, EventArgs e)

{

int[] puzzleVals = new int[9];

int puzzleValue = 0;

int i = 0;

foreach (Control x in this.Controls)

{

if ((x is TextBox) && (x.Name != "txtUser"))

{

int.TryParse(x.Text, out puzzleValue);

puzzleVals[i] = puzzleValue;

i++;

}

}

// If any of the numbers are duplicates, show a warning

if ((isDuped(puzzleVals, puzzleVals[0])) || (isDuped(puzzleVals, puzzleVals[1])) || (isDuped(puzzleVals, puzzleVals[2])||(isDuped(puzzleVals, puzzleVals[3]))||(isDuped(puzzleVals, puzzleVals[4]))||(isDuped(puzzleVals, puzzleVals[5]))|| (isDuped(puzzleVals, puzzleVals[6]))|| (isDuped(puzzleVals, puzzleVals[7]))|| (isDuped(puzzleVals, puzzleVals[8]))))

{

MessageBox.Show(userName + " You have errors here.");

foreach (Control x in this.Controls)

{

if ((x is TextBox) && (x.Name != "txtUser"))

{

int.TryParse(x.Text, out puzzleValue);

if (isDuped(puzzleVals, puzzleValue))

{

x.Select();

x.Focus();

x.ForeColor = Color.Crimson;

}

}

}

}

else

{

MessageBox.Show("Congrats " + userName + " You Won!");

btnSolve.Visible = false;

btnSolve.Enabled = false;

btnNew.Enabled = true;

btnNew.Visible = true;

}

}

private void createBoxes()

{

// Create 9 textboxes

TextBox[] txt = new TextBox[9];

// Format the boxes

for (int i = 0; i < 9; i++)

{

int spacer = 43;

txt[i] = new TextBox();

if (i < 3) // Create the first row

{

txt[i].Left = i \* spacer;

txt[i].Top = 0;

}

else if (i < 6) // Create the second row

{

txt[i].Left = (i - 3) \* spacer;

txt[i].Top = spacer;

}

else // Create the last row

{

txt[i].Left = (i - 6) \* spacer;

txt[i].Top = spacer \* 2;

}

txt[i].Multiline = true;

txt[i].Font = new Font(txt[i].Font.FontFamily, 16);

txt[i].Size = new System.Drawing.Size(35, 35);

txt[i].Name = "txt" + (i + 1);

this.Controls.Add(txt[i]);

}

// Add Focus and keystroke based events to each generated textbox

foreach (Control x in this.Controls)

{

if (x is TextBox && x.Name != "txtUser")

{

x.Enter += ControlReceivedFocus;

x.KeyPress += LimitInput;

}

}

}

private void newPuzzle()

{

// Generate random numbers for the puzzle

int[] vals = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

int[] randomVals = Shuffle(vals);

int[] randomCells = Shuffle(vals);

randomCells = randomCells.Take(4).ToArray();

int getVal = 0;

int i = 1;

foreach (Control x in this.Controls)

{

if ((x is TextBox) && (x.Name != "txtUser"))

{

x.Text = ""; // Reset the textboxes for the new puzzle

x.Enabled = true; // Enable all previously disabled boxes

if (x.Name == ("txt" + i))

{

if (randomCells.Contains(i))

{

x.Text = randomVals[getVal].ToString();

x.Enabled = false; // Disallow changes to this box

getVal++;

}

}

i++;

}

}

}

private int[] Shuffle(int[] array)

{

Random r = new Random();

for (int i = array.Length; i > 0; i--)

{

int j = r.Next(i);

int k = array[j];

array[j] = array[i - 1];

array[i - 1] = k;

}

return array;

}

private void ControlReceivedFocus(object sender, EventArgs e)

{

// Reset the BackColor for each form control

foreach (Control x in this.Controls)

{

x.BackColor = default(Color);

}

// Set the active control BackColor to beige

this.ActiveControl.BackColor = System.Drawing.Color.Beige;

}

private void LimitInput(object sender, KeyPressEventArgs e)

{

// Disallow non-numerical keystrokes, except backspace

if (!(Char.IsDigit(e.KeyChar) || (e.KeyChar == (char)Keys.Back)))

e.Handled = true;

}

private void btnSaveUser\_Click(object sender, EventArgs e)

{

txtUser.Text = txtUser.Text.Trim();

txtUser.Enabled = false;

btnSaveUser.Visible = false;

btnSaveUser.Enabled = false;

btnNewUser.Visible = true;

btnNewUser.Enabled = true;

userName = txtUser.Text;

}

private void btnNewUser\_Click(object sender, EventArgs e)

{

txtUser.Text = "";

txtUser.Enabled = true;

btnNewUser.Visible = false;

btnNewUser.Enabled = false;

btnSaveUser.Visible = true;

btnSaveUser.Enabled = true;

userName = "";

}

private bool isDuped(int[] a, int x)

{

int dupes = 0;

for (int i = 0; i < a.Length; i++)

{

if (a[i] == x)

dupes++;

if (dupes > 1)

return true;

}

return false;

}

}

}