Manual de Juego Sudoku

Manuel Suárez-Veronica Pozo-José Salas

8 de julio de 2013

Índice general

1. **Introducción** **2**
2. **El Juego** **3**
3. **Historia** **4**
4. **Aplicacion** **5**
5. **Funcionalidad** **7**
6. **Colaboraciones** **9**
7. **Conclusiones** **10**

Capítulo 1

Introducción

El libro a continuación es creado como una herramienta para el desarrollo de habilidades de edición colaborativa de documentos de texto plano. La herramienta que habilita dicha colaboración, en este taller, es Git pero podría ser reemplazada por otros sistemas de versionamiento.

Capítulo 2

El Juego

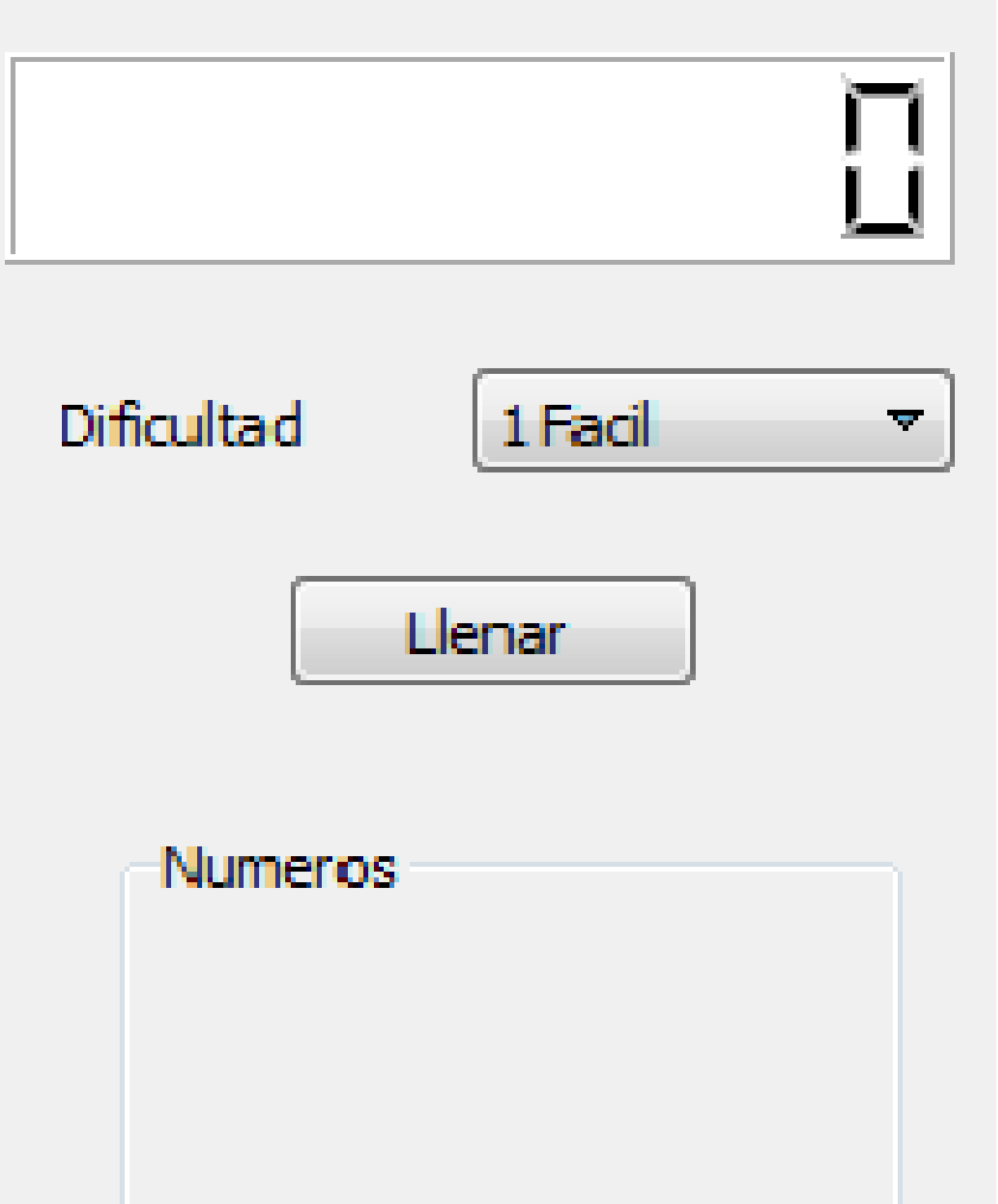
Capítulo 3

Historia

El Sudoku puede haberse ideado a partir de los trabajos del matemático suizo Leonhard Euler (1707-1783), quién utilizó los cuadrados latinos1 para realizar cálculos probabilísticos, aunque su aparición más obvia se dio en el año 1979, cuando el norteamericano Howard Garns publica en Nueva York, a través de la empresa Dell Magazines, un juego muy similar conocido bajo el nombre de "Number Place". En el año 1984 una editorial japonesa lo exportó hacia ese país, divulgándolo en el periódico "Monthly Nikolist" bajo el nombre de "Los números deben estar solos" (cuyo apócope en japonés es SuDoku). La editorial introdujo con el correr de los años algunas innovaciones que hicieron que el juego ganara gran popularidad entre los lectores. Sus inicios reales como Sudoku en Occidente se remontan a los años 2004/2005, gracias a su publicación en los diarios "The Times" y "The Daily Mail". Desde aquel entonces, el juego ha alcanzado una gran popularidad en el mundo moderno: en 2005 se publicó el primer libro acerca de Sudokus en el mundo ("Los mejores Sudokus", con 200 tableros agrupados en 4 niveles de dificultad) y se lo incluyó entre los 9 problemas del ICPC (sigla de International Collegiate Programming Contest).

Capítulo 4

Aplicacion



Capítulo 5

Funcionalidad

Generacion de tableros:

- Algoritmo implementado.
  - Se genera un tablero con numeros random hasta que este sea resolvible.
  - Se verifica si es resolvible.
  - Se trata de resolver por arreglos de posibildades de numeros en casillas.
  - Si no se puede resolver así, se utiliza la técnica del Backtracking.
  - Utizando Backtracking se busca solucionar cambiando los numeros de-

```
void Generador::generarSolucionRando
    srand((unsigned int) time(NULL))
    int x, y, num_de_Random,v;

            num_de_Random = 25;
       do {
           for(int i=0;i<81;i++) tab
            for (int i = 0; i < num_
                x = rand() % 9;
                y = rand() % 9;
                v = rand() % 9 + 1;
                if (MovimientoValido
                    Set(x, y, v);
                }
                else{
                    i--;
                }
                x=y=v=0;
            }


    } while (!resolverTablero());
}
```

tras del vacio, y utilizando recursion de la misma.

- Dificultad.
  - Se procede a quitar numeros del arreglo, y comprobar si se puede resolver y tienen única solución.
  - Se agregan en un pila los numeros y posiciones que fueron quitadas y se podia obtener única solución.
  - Segun el numero de dificultad se obtienen posiciones random repetidamente tantas casillas hay, y se ve las veces que puede estar ese numero.

```
bool stillFoundOne = true;
pila= QStack<Posicion>();
while (Insertar > 0 && stillFoundOne) {
    stillFoundOne = false;

    int yoffset = rand() % 9;
    int xoffset = rand() % 9;

    for (int y = 0; y < 9 && Insertar > 0 && !lista_op.empty(); ++y)
        for (int x = 0; x < 9 && Insertar > 0 && !lista_op.empty(); ++x) {
            int realx = (x + xoffset) % 9; // The real x and y positions
            int realy = (y + yoffset) % 9;
            Posicion pe((realx*9)+realy, tablero[(realx*9)+realy]);
            pila.push(pe);
            tablero[(realx*9)+realy]=0; // Remove the element from the board and try to solve it again

            bool foundOne = false;
            for (QList<Posicion>::iterator it = lista_op.begin(); it != lista_op.end() && !foundOne;) {
                tablero[it->GetPos()]=it->GetValor();
                if (VerificarResolucion(true)) { // Board is solvable by ScanSolve()
                    Posicion addpos(it->GetPos(), it->GetValor());
                    pila.push(addpos);

                    it = lista_op.erase(it); // Remove from the undoList and add to the pila

                    --Insertar;
                    foundOne = true;
```

- Guardar y cargar tableros cifrados .
- Jugar y finalizar juego (2).
- Puntaje (3).

Capítulo 6

Colaboraciones

Correcta documentación de colaboración otorga puntaje. • Si la documentación es inconsistente con una distribución adecuada de trabajo se considerará descontar puntos. • Contabilicen número de commits por integrante. • ¿En qué fechas hay picos?

Capítulo 7

Conclusiones

Se concluye