Lenguajes de Programación: Experiencias

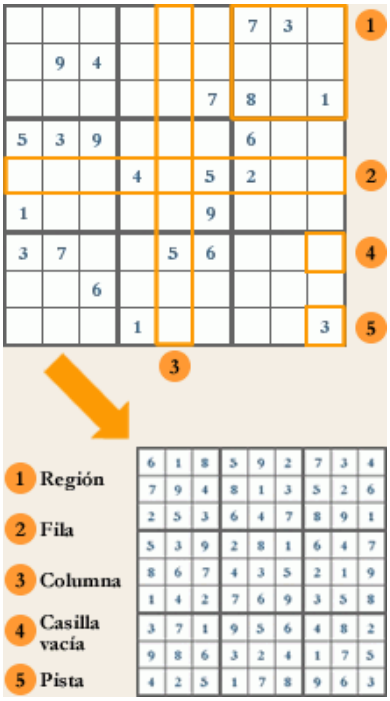
Iván Aveiga
Kevin Campuzano

September 4, 2013

C: Sudoku [1]

0.1 Información

Sudoku es un juego en donde en una cuadrícula de 9x9 no debe de tener números repetidos en filas, columnas o subcuadrícula de 3x3.



[2]

0.2 Aspectos Técnicos

Implementamos un juego de Sudoku en que el usuario debe resolver un tablero de sudoku según la difucultad elegida, entre estas tenemos.

- Fácil: 36-41 casillas llenas.
- Normal: 32-35 casillas llenas.
- Avanzado: 28-31 casillas llenas.
- Experto: 23-27 casillas llenas.

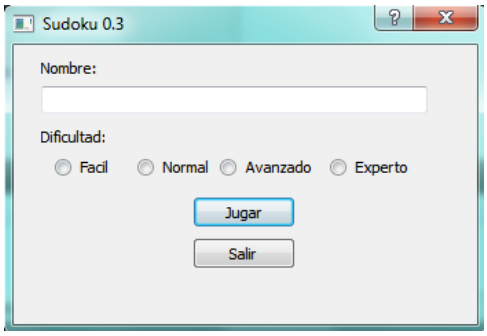
[1] Iván Aveiga, Kevin Campuzano
[2] http://www.sudokumania.com.ar/sm1/images3/reglas.png

0.2.1 Algoritmo

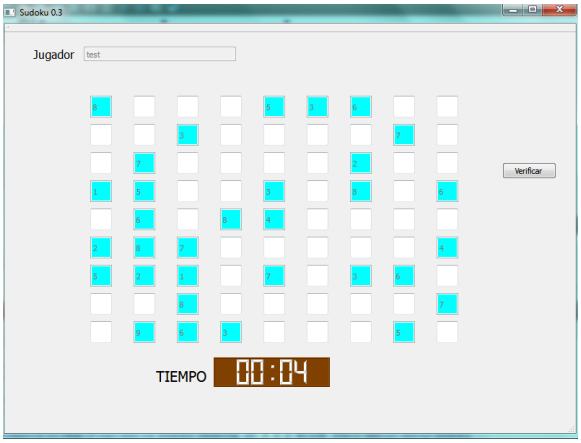
Se carga internamente un sudoku aleatoriamente entre cuatros sudokus almacenados en texto plano, dependiendo de la dificultad se seleccionan casillas aleatoriamente para usar en el tablero en que el usuario jugara. Se usa el concepto de random walks, para la selección de casillas se usó Programming Sudoku [Wei Meng]. El usuario puede verificar si las casillas estan correctamente colocadas dando clic en el botón verificar.

0.2.2 Interfaz Gráfica

Al iniciar la aplicación se muestra una ventana donde ingresar el nombre y la dificultad a jugar.

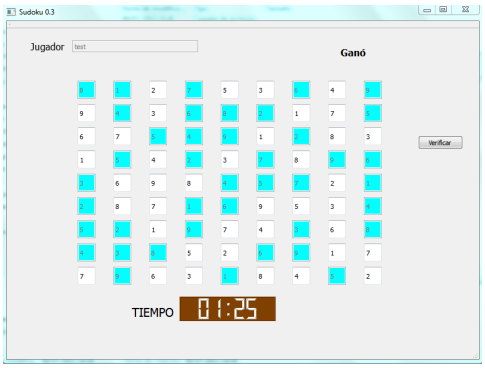


Una vez ingresado el nombre y seleccionada la dificultad aparece la ventana a jugar, mostrada a continuación.



Podemos ver el ingreso de números validados a un solo dígito y que por motivos de ilustración podemos ver que las celdas erróneas luego de dar clic en el botón verificar se tornan de color magenta.

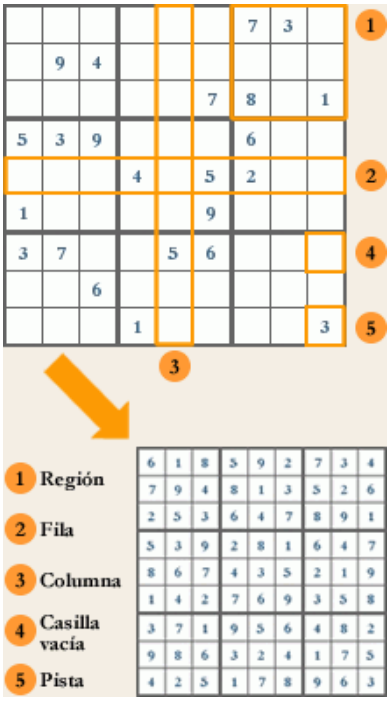
Al resolver correctamente el tablero el cronómetro se paraliza y se muestra el mensaje de que el usuario ganó.



Python: PySudoku 0.9 [3]

0.3 Información

Sudoku es un juego en donde en una cuadrícula de 9x9 no debe de tener números repetidos en filas, columnas o subcuadrícula de 3x3.



[4]

0.4 Aspectos Técnicos

Implementamos un juego de Sudoku en que el usuario debe resolver un tablero de sudoku según la difucultad elegida, entre estas tenemos.

0.4.1 Herramientas de Desarrollo:

- **Lenguaje de Programación:** Python 2.7 64 bits.
- **Libreria Gráfica:** PyQt 4 64 bits.
- **Sistema Operativo:** Windows 7 64 bits.
- **Sistema de Versionamiento:** Github.

[3] Iván Aveiga, Kevin Campuzano
[4] http://www.sudokumania.com.ar/sm1/images3/reglas.png

- **Herramienta de Documentación:** Doxygen.
- Fácil: 36-41 casillas llenas.
- Normal: 32-35 casillas llenas.
- Avanzado: 28-31 casillas llenas.
- Experto: 23-27 casillas llenas.

0.4.2 Algoritmo

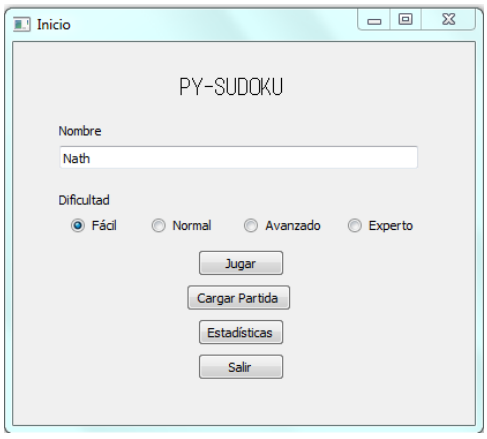
La forma de generar un tablero es usando un algoritmo que genere un tablero de 9x9 completo, una vez que se ha generado el tablero se procede a eliminar celdas aleatoriamente, el número depende de la dificultad. Para el juego se manejan dos tableros de sudoku, uno completo que se mantiene de fondo para la comparación y las ayudas, y el tablero en que el usuario jugará.

Generación de un tablero lleno

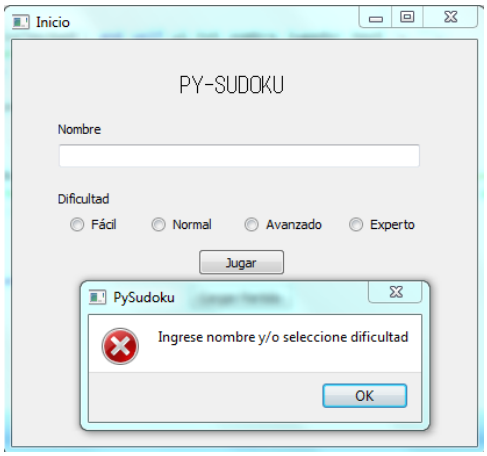
```
proc poblar_matriz( )
    matriz = [9][9]
    col = 0
    mientras  col < 9:
        posibles = [1,2,3,4,5,6,7,8,9]
        mezclar(posibles)
        fila = 0
        prueba = 0
        mientras fila < 9 y prueba < 200
            malas = 0
            parar = 0
            para cada elemento i en posibles
                si posibles[i] está en columna
                    malas += 1
            si malas == 9
                limpiar matriz
                parar = 1
                fila = 0
                prueba += 1
            si parar == 0
                numero = aleatorio (1,9)
                mientras conflicto(numero)
                    numero = aleatorio(1,9)
                matriz[fila][col] = numero
                remover numero de posibles
                filas + = 1
        col += 1
        si prueba == 20
            limpiar matriz
            col = 0
    retornar matriz
```

0.4.3 Interfaz Gráfica

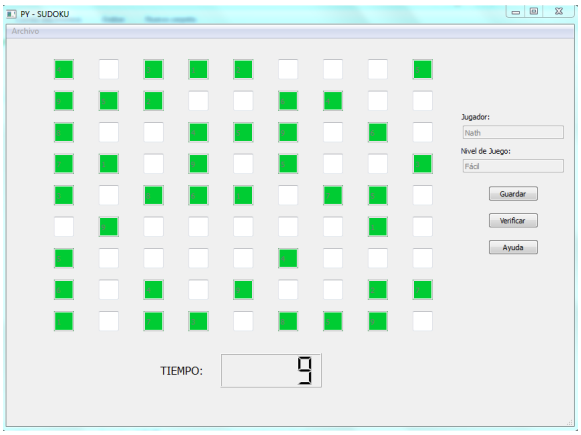
Al iniciar la aplicación se muestra una ventana donde ingresar el nombre y la dificultad a jugar.



Mientras no se ingrese los datos de nivel y el nombre no se podrá iniciar el juego, saltando un mensaje de error.



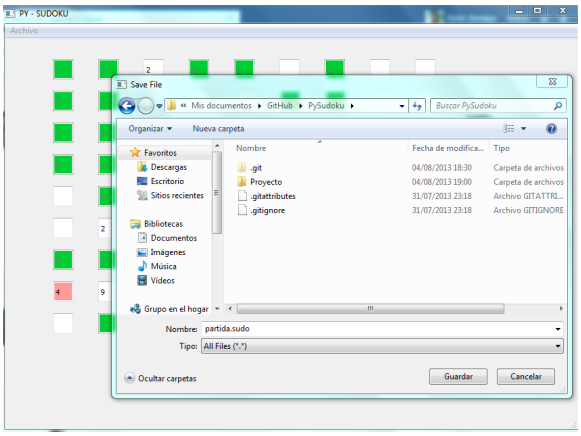
Una vez ingresado el nombre y seleccionada la dificultad aparece la ventana a jugar, mostrada a continuación.



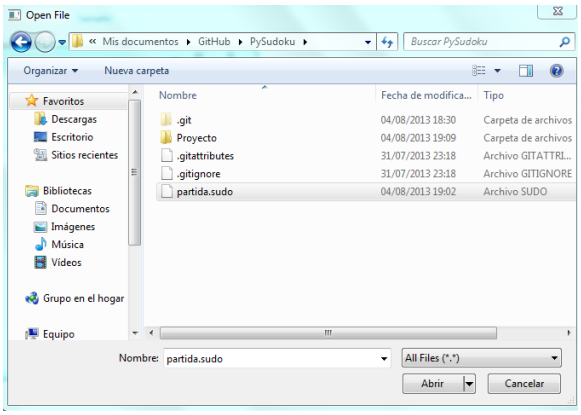
0.4.4 Funcionalidad

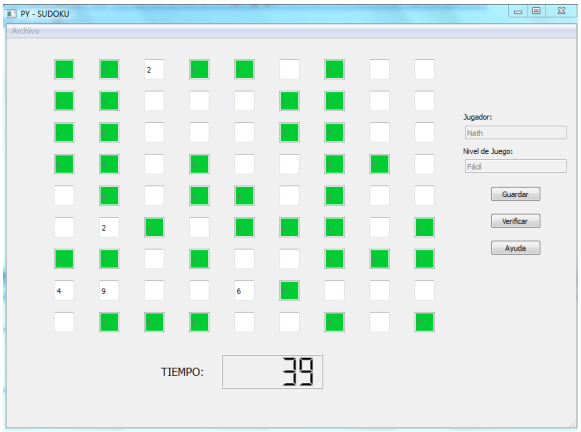
El jugador cuenta con un botón de ayuda, son 5 ayudas, una vez usada las ayudas se deshabilita este botón, si el jugador usa por lo menos una ayuda queda eliminada la opción de participar en el ranking de mejores jugadores. El juego genera una celda aleatoria y muestra el valor correcto, además se deshabilita la edición de dicha celda y su color se torna turquesa.

El jugador puede guardar la partida en cualquier momento y a su vez cargarla.

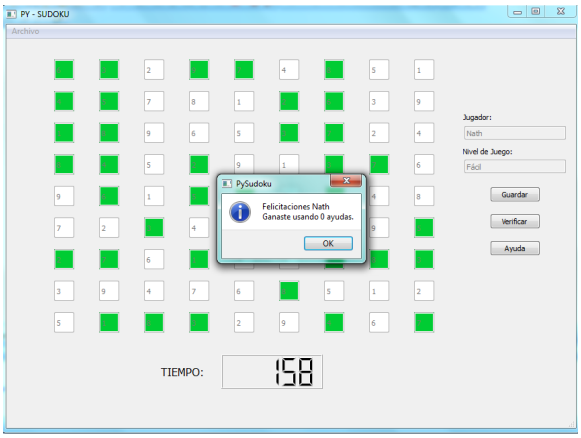


Se puede ver el proceso de carga de una partida guardada:

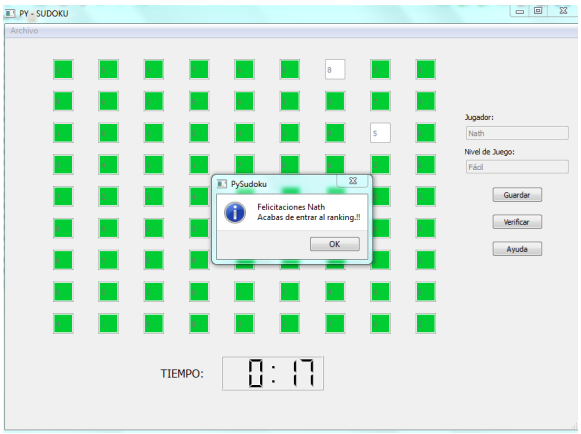




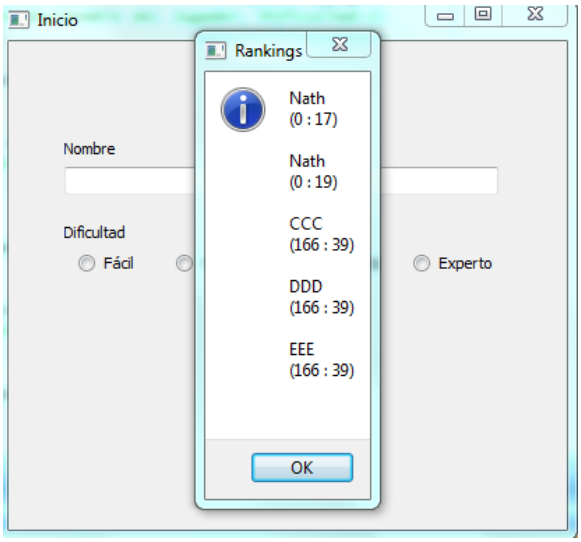
Al resolver correctamente el tablero el cronómetro se paraliza y se muestra el mensaje de que el usuario ganó.



El puntaje se maneja en base al número de segundos que toma resolver el Sudoku, si es un buen puntaje entra al ranking, en esta imagen podemos ver el mensaje de ingreso al ranking.



El usuario puede consultar el ranking, donde se muestra los mejores 5 tiempos.



0.5 Conclusiones

Python es un lenguaje que presta muchas facilidades, una de ellas es la facilidad de escribir código de manera casi natural, el tipado dinámico ayuada en muchas cosas, PyQt el binding de Qt para Python da la facilidad del diseño de interfaces con el Qt Designer. A Diferencia del proyecto pasado (implementación en C++) se logró cumplir las especificaciones del proyecto.

La especificación de guardar / cargar una partida cifrada se resolvió usando serialización y guardando el archivo en binario, esto garantiza que el usuario no pueda entender el contenido al abrir la partida con un editor de texto plano, además que para ver el contenido debe conocer la estructura del objeto serializado.

Haskell: Lexer 0.1 [5]

0.6 Introducción

Implementamos la idea básica de un analizador léxico, fue escrito "a mano", hay herramientas que podían usarse como Alex para generar análisis léxico y Parsec para *parsear*. Partimos definiendo los tokens que vamos a reconocer siendo los siguientes:

- Operator: +, -, /, *, ; /, %
- Reserved Word: auto, break, case, const, continue, default, do, else, enum, extern, for, goto, if, register, return, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while.
- Operadores de Incremento y Decremento.
- Operadores lógicos.
- Operadores relacionales.
- Primitve types: int, float, double, char, long
- Integers.
- Reals.
- Strings.
- Todo lexeme que no entre en la categoría de los tokens anteriores entra a la categoría de UNKNOWN TOKEN.

El lexer opera bajo las siguientes condiciones:

1. Solo funciona con espacios en blanco, es decir, no detecta la siguiente expresión en sus respectivos lexemas

$$(3 * 2) + 3$$

2. No Detecta punteros.

El algoritmo básico para generar la lista de (lexeme, tokens) es el siguiente:

1. Leer el archivo fuente .c .
2. Generar la lista de lexemas usando el espacio en blanco como delimitador.

[5] Iván Aveiga, Kevin Campuzano

3. Asignar a cada lexema en la lista su respectivo token.
4. Guardar en un archivo .csv la lista de (lexema, token).

0.7 Git

Fue agradable trabajar con sistema de versionamiento, una gran experiencia que servirá para el mundo profesional además del académico. Sacar el máximo provecho de sus ventajas además de tener los gráficos colaborativos de cada integrante del proyecto. Aprendimos a resolver la mayoría de conflictos que teníamos cuando se dañaba la rama (branch) de trabajo. Utilizamos algunas veces la línea de comandos ante que el cliente gráfico.

0.8 LaTeX

La curva de aprendizaje de LaTeXes un poco compleja, muchas veces es difícil el cambio de paradigma WYSIWYG a WYSISWYM. Pero una vez dominada la sintáxis básica se puede ver las ventajas. En nuestro caso no nos preocupamos por los márgenes, identaciones, justificaciones, tipo de letra y demás. Nos enfocábamos en lo que queríamos escribir, el contenido. Al momento de crear el libro nos ayudó bastante la inclusión de archivos, teniendo lo único que hacer es indicar la ruta de los archivos .tex que ya teníamos creados anteriormente.

[6]Ivan Aveiga,Kevin Campuzano

Conclusiones

0.9 Observaciones

Aprendimos la ventaja de trabajar con sistemas de versionamiento, documentación usando la herramienta Doxygen, escritura de manuales en LaTeXademás de usar herramientas para creaciones de aplicaciones gráficas (Qt) y programación funcional (bases).

Pensamos que Python es un lenguaje que tiene muchas ventajas, tanto por su simplicidad como las librerias disponibles tanto para tareas de desarrollo (web) como para computación científica (scypy, matplotlib, pyml). Así que pensamos usar mucho ese lenguaje en proyectos a futuros.

El curso fue muy provechoso en todo lo aprendido, respecto a programación funcional logramos notar el gran poder que tienen unas pocas líneas de código y la expresividad que tienen.

Seria interesante aprender a más profundidar Haskell por el gran poder que tiene y por ser funcional puro además de ser de evaluación perezosa. Profundizar en Python.