

Universidad Nacional de General Sarmiento



Programación 3 - Comisión 1

TP N°1 - Fuera luces!

Integrantes: Guillermo Dominguez; Tomás Vergalet

Introducción

En este presente informe se llevará a cabo el desarrollo del Trabajo Práctico N°1 de los alumnos Guillermo Domínguez y Tomás Vergalet, estudiantes de Programación III de la Universidad Nacional de General Sarmiento. A continuación, se explicarán las decisiones tomadas; como fueron aplicados los distintos conceptos y temas vistos durante la primera etapa de esta cursada; los distintos inconvenientes que se presentaron a lo largo de la realización del trabajo y las decisiones que se tomaron para poder solucionarlas.

Para finalizar el informe se hará una breve conclusión sobre el desarrollo del trabajo práctico.

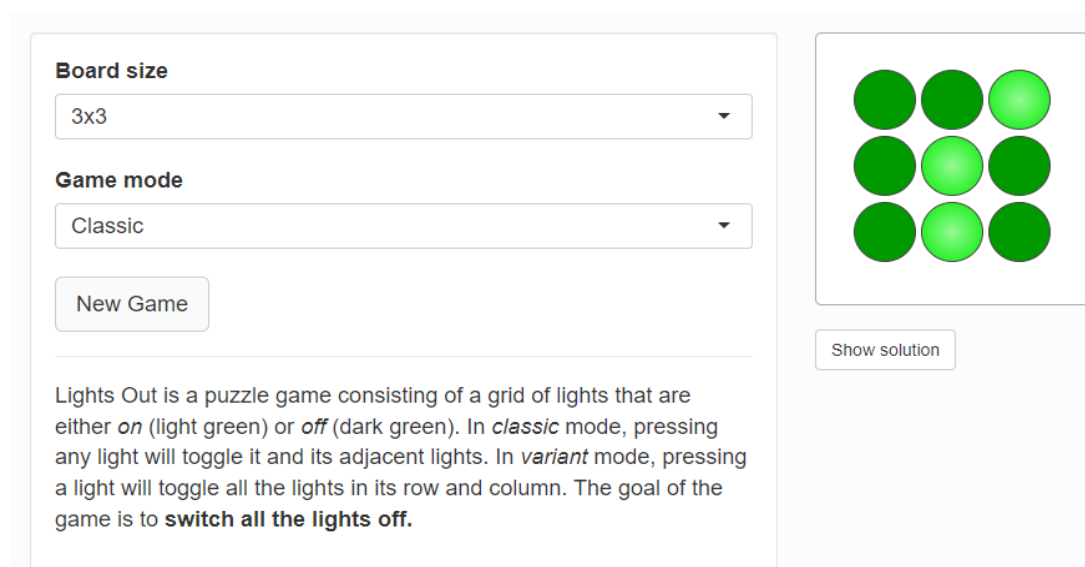
Desarrollo

En este TP debemos modelar y desarrollar el juego “Lights Off”.

Las reglas del juego son las siguientes:

“Se juega en una grilla de 4x4, y en cada posición se tiene una luz, que puede estar encendida o apagada. Inicialmente las luces tienen una combinación aleatoria de encendidos y apagados. En cada turno, el jugador hace clic sobre una luz, y este click tiene el efecto de cambiar el estado de la luz de la casilla y de las luces ubicadas en su fila y columna. El objetivo es lograr que todas las luces de la grilla terminen apagadas.”

Y además se nos brindó el siguiente ejemplo:



Link: <https://daattali.com/shiny/lightsout/>

Para ello, se modelaron las clases “**Main**”, “**Interface**” y “**Controller**”, para de esta manera poder tener un patrón **MVC** (*Model - View - Controller*), siendo la primera la encargada de llevar a cabo la parte lógica del juego, incluyendo las reglas y el estado actual del programa; la segunda se encarga de mostrar en pantalla lo que ocurre con el programa, interactuando así de esta manera con el usuario; la última clase se encarga de interconectar entre sí a la interfaz visual y al modelo mediante principalmente métodos de getters y setters, definiendo así un límite a los roles de cada clase.

Para lograr el objetivo de crear el juego, desde la clase “**Main**” decidimos implementar un tablero cuadrado generado a partir de una matriz mediante el método “**createBoard**”. El mismo a su vez genera una pila de pasos para llegar a la solución. La matriz trabaja con el tipo de datos *boolean*, lo cual permite manipular más fácilmente el

estado actual de la luz (*true* si está encendida o *false* si está apagada). Otra utilidad de este método, es que también ejecuta “**updateStackSolution**” el cual se ocupa de verificar si el usuario pulsó sobre la luz indicada como solución, y dependiendo del caso, agrega o quita elementos de la pila.

El método “**updatePercentCompleted**” se encarga de verificar la cantidad de luces apagadas y actualizar el avance del juego en consecuencia.

Es también desde esta clase donde se actualizan sus valores mediante los métodos “**toggleValueRow**” y “**toggleValueColumn**”. Las mismas se ejecutan desde “**updateBoard**”, quien recibe un string con la posición y se encarga de validar si es posible ejecutar los métodos mencionados anteriormente. Otra de las funciones de la clase main es llevar el registro de cuántos movimientos hizo el usuario; y en la función “**readScoreFile**” se utilizan las clases *BufferedReader* y *FileReader* para poder acceder al registro de los 10 mejores jugadores, que se encuentra almacenado en un “.txt”. Con el método “**checkNewScore**” se verifica si el jugador está en condiciones de acceder a dicho grupo. En tal caso, con la función “**writeScoreFile**” se actualiza el archivo y se guarda el registro. Por último, se generaron distintas funciones con la finalidad de orientarnos a nosotros mismos y hacer pruebas de funcionamiento.

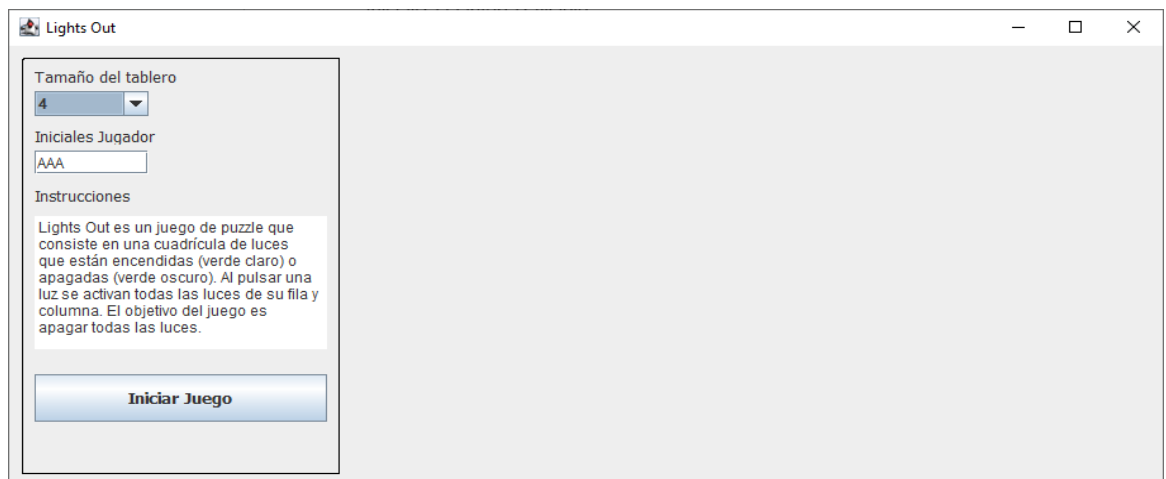
El método “gameCompleted” se encarga de recorrer la tabla y verificar que todos sus elementos estén en *false*. En tal caso, retorna *true*.

Todo cambio en el modelo durante la ejecución tiene que ser comunicado con la clase “**Controller**”, ya que el mismo será el encargado de pasarle dichas actualizaciones a la interfaz para que esta se actualice. Todo esto se hace mediante *getters* y *setters*.

Finalmente, la clase “**Interface**” contiene la interfaz que interactúa con el usuario. Se desarrolló utilizando la herramienta *WindowBuilder*. Con el método “**Main**” se inicializa la interfaz visible.

El método “**createAppWindow**” se usa para definir el título del programa y su tamaño.

El método “**createPanellInfo**” contiene componentes que cubren distintas funcionalidades: establecer el tamaño de la grilla, ingresar las iniciales del jugador, mostrar las instrucciones, y por último un botón asociado a un evento para iniciar el juego.



El método **“createPanelStatus”** se encarga de mostrar la cantidad de movimientos que realizó el usuario (turnos), y, además, mostrar el progreso porcentual respecto a las luces apagadas en el tablero. Dispone además un botón para mostrar una posible solución. Por último, los *records* registrados hasta el momento, utilizando un método **“createScorePanel”**

Turno actual			
0			
Avance tablero			
0 %			
Siguiente paso			
Mostrar solución			
Records			
Posicion	Jugador	Tamaño	Turno
1	AAA	04	005
2	AAA	04	006
3	AAA	04	006
4	AAA	04	007
5	AAA	06	007
6	AAA	05	008
7	AAA	04	009
8	AAA	04	010
9	AAA	04	011
10	AAA	04	012

El método **“createPanelLights”** crea un panel para contener la grilla de luces.

El método **“addLightsToPanel”** agrega las luces al panel mencionado anteriormente, y las asocia a los eventos generados por los clics del usuario.



Con los métodos asociados anteriormente se logra finalizar con la interfaz. A partir de esto, el usuario puede interactuar con la aplicación, y la misma ejecutará los siguientes métodos.

El método “**updateLightsToPanel**” se encarga de actualizar las luces por cada evento previamente mencionado.

El método “**updateTurn**” se utiliza para actualizar en pantalla la cantidad de movimientos que hace el usuario.

El método “**showSoluctionToPanel**” se encarga de cambiar el color de fondo de la luz sugerida que el usuario puede presionar para llegar hasta la solución.

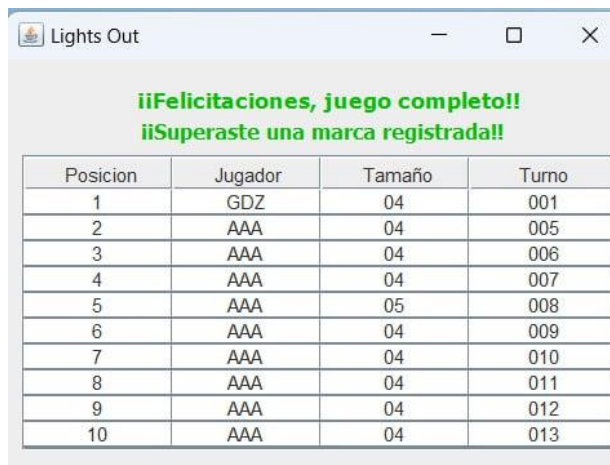
En el caso de que se apaguen todas las luces del tablero, se ejecuta el método “**endGame**”, el cual se encarga de “limpiar” la pantalla. Posteriormente, agrega una alerta felicitando al usuario, y mostrando una tabla con los mejores jugadores.



The screenshot shows a window titled "Lights Out" with a congratulatory message in green text: "¡¡Felicitaciones, juego completo!!". Below the message is a table with 4 columns: Posicion, Jugador, Tamaño, and Turno. The table lists 10 players, all named "AAA", with various scores and turn numbers.

Posicion	Jugador	Tamaño	Turno
1	AAA	04	005
2	AAA	04	006
3	AAA	04	006
4	AAA	04	007
5	AAA	06	007
6	AAA	05	008
7	AAA	04	009
8	AAA	04	010
9	AAA	04	011
10	AAA	04	012

Si el jugador tiene un puntaje récord, el usuario también es notificado y su record se registra.



The screenshot shows a window titled "Lights Out" with two congratulatory messages in green text: "¡¡Felicitaciones, juego completo!!" and "¡¡Superaste una marca registrada!!". Below the messages is a table with 4 columns: Posicion, Jugador, Tamaño, and Turno. The table lists 10 players, with the first player "GDZ" having a score of 04 and turn 001, which is the highest score in the list.

Posicion	Jugador	Tamaño	Turno
1	GDZ	04	001
2	AAA	04	005
3	AAA	04	006
4	AAA	04	007
5	AAA	05	008
6	AAA	04	009
7	AAA	04	010
8	AAA	04	011
9	AAA	04	012
10	AAA	04	013

Conclusión

Podemos concluir resumiendo que “Lights Off” es un juego cuyo objetivo es hacer que el usuario apague todas las luces de un tablero.

En nuestra implementación usamos el modelo MVC con 3 clases. Una para modelar las reglas y llevar el control en tiempo de ejecución; otra para un diseño estético e interactuar con el usuario y por último una que comunique ambas clases. Tuvimos dificultades a lo largo del proceso típicas de programadores que están dando sus primeros pasos con la herramienta. La principal fue que cada vez que teníamos que limpiar la pantalla porque cambiaba el estado del juego, se nos borraban los paneles de la misma y no se generaban los nuevos. Esto se debía a que en muchas ocasiones, nos faltaba el “*repaint*”. Pero más allá de eso nos sentimos bastante conformes con lo aprendido.