Lights Out: Un Puzzle Renovado

[a]Deppeler Eric; [b]Schefer Mauricio Nicolás; [c]Arduña Zago Agustín Juan Luis; [d]Velazco Gez Schegtel Juan Ignacio.

[a]UTN, Facultad Regional Resistencia, B, ericdeppeler@hotmail.com
[b]UTN, Facultad Regional Resistencia, B, maurischefer24@gmail.com
[c]UTN, Facultad Regional Resistencia, B, agustinarduna@gmail.com
[d]UTN, Facultad Regional Resistencia, B, velazcoschegtel@ca.frre.utn.edu.ar

Resumen

El siguiente informe presentará el desarrollo del videojuego "Lights Out", consistente en apagar todas las luces dispuestas en un tablero de forma aleatoria. Para ello, se debe presionar en las mismas, lo que ocasiona que varíe el estado (encendido/apagado) tanto de la luz que se presionó, como así también las que se encuentran a su alrededor.

1. Introducción

El videojuego se desarrolló en la plataforma Pharo Smalltalk, en su versión 7.0. El paradigma de programación empleado para el diseño del software es el de la orientación a objetos, donde se planteó una clase encargada de la inicialización y control de ejecución de las distintas partes que componen el juego.

Hay cuatro modos de juego disponibles: *Original*, el modo de juego clásico de Lights Out, en el cual las luces que cambian de estado es la luz presionada más todas las adyacentes a la misma. *Filas y Columnas* donde, con respecto al anterior modo de juego, se alterna también el estado de toda la columna y fila de luces de la luz presionada. Está presente también el modo *Diagonal* donde las luces que varían su estado es la luz seleccionada en conjunto a las que estén inmediatamente dispuestas de forma diagonal a ésta. Por último, se encuentra disponible el modo de juego *"Aleatorio",* donde la variación de estado de las luces presenta un comportamiento errático, surgido como una combinación de los anteriores modos de juego ya nombrados.

El jugador puede además elegir la dimensión del tablero con la que desee iniciar la partida, siendo los tamaños disponibles 3x3, 5x5 y 7x7. Estos pueden verse como forma análoga a niveles de dificultad. Una vez iniciada la partida, se cargará una disposición aleatoria de luces encendidas y se completará exitosamente el juego una vez estén todas ellas apagadas.

2. Desarrollo

Se dio inicio al proyecto con el modelado de las clases aplicando conceptos relacionados al paradigma orientado a objetos, donde se definen las características de las clases y su relación entre ellas. Estas características serán necesarias para que la aplicación funcione conforme a lo deseado (ver Fig. 1).

Tabla 1. Objetos que componen al videojuego.

LightsOut
Jugador
Partida
Tablero
Luz

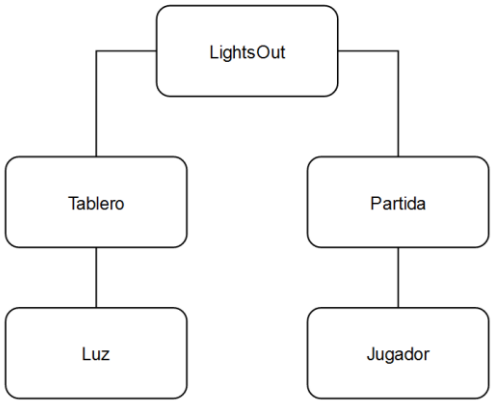

Fig. 1: Esquema básico de los objetos y sus relaciones.

La clase base es la denominada *"LightsOut"*, encargada de inicializar el programa comunicándose con las clases que lo componen (ver Fig. 2). Al ejecutarse el videojuego, esta se ocupa de exponer un menú de bienvenida donde se puede continuar con la inicialización del programa o terminar allí su ejecución (ver Fig. 3). Si se continua, se hace desde aquí un llamado a la clase *"Partida"* la cual realiza, a su vez, un llamado a la clase *"Jugador"* que permite el despliegue de una

ventana que solicita al usuario un alias (ver Fig. 4) para que más adelante pueda visualizar este nombre al finalizar la partida junto con la cantidad de "clics" empleados para ello. Continuando su ejecución, la clase *"Partida"* se encarga de listar cuatro opciones concernientes a los modos de juego disponibles (ver Fig. 5), como así también las posibles opciones para el tamaño del tablero (ver Fig. 6). Estos valores están denotados como las variables de instancia de la clase.

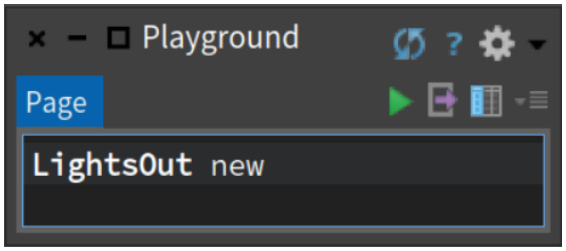

Fig. 2: Iniciando el videojuego.

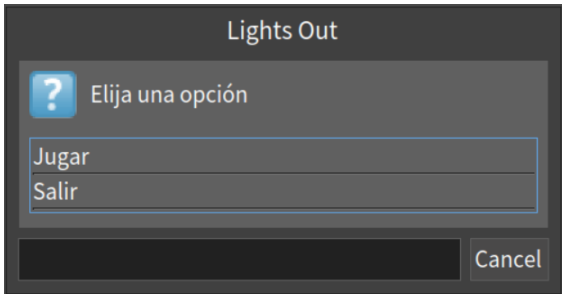

Fig. 3: Pantalla de bienvenida.

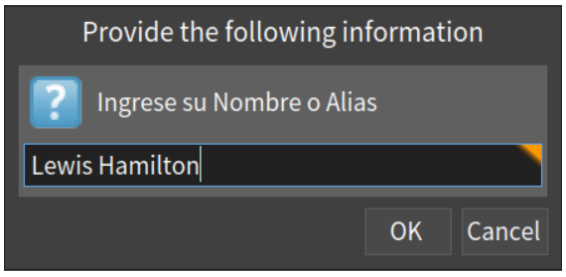

Fig. 4: Pantalla de bienvenida.

La clase base continúa su ejecución delegando la creación del tablero del juego (ver Fig. 7) haciendo un llamado a la clase *"Tablero"*, que es una subclase de *"BorderedMorph"*. El motivo detrás de la decisión de recurrir a una herencia de esta clase es que la misma facilita la creación de elementos gráficos, en este caso, un cuadro que a posteriori nos servirá como base para una matriz de celdas, que representan las luces del videojuego. En esta clase también se definen los métodos encargados de modelar qué luces, al ser presionadas sobre ellas, modificarán su estado. Estos patrones de comportamiento son finalmente los distintos modos de juego disponibles para el jugador. Por último, hay dos métodos relacionados a la finalización de la partida. Estos son *"recorrerLuces"* y *"contadorDeClicks"*. El primero es un método que analiza si se han apagado todas las luces, y el segundo lleva a cabo el control del número de clicks empleados para conseguirlo.

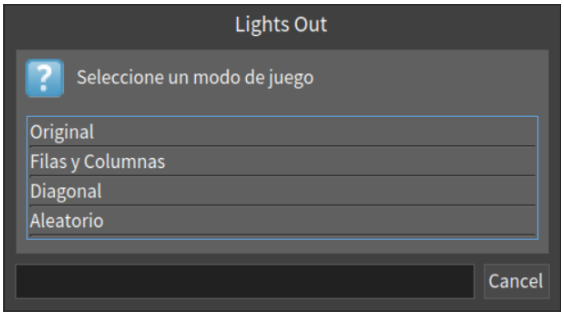

Fig. 5: Despliegue de los modos de juego disponibles.

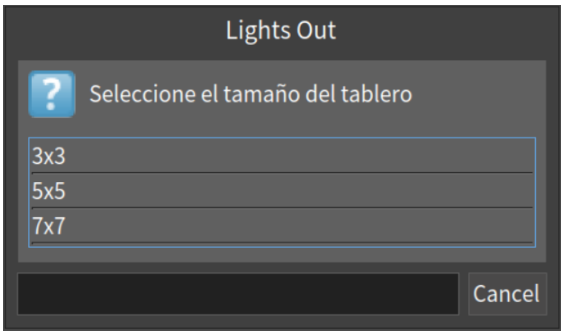

Fig. 6: Las tres dimensiones posibles para el tablero.

Las luces están definidas en términos de la clase *"Luz"*, subclase de *"SimpleSwitchMorph"*, por la cual hereda métodos que permiten crear objetos que pueden encontrarse en uno de dos estados posibles, provocando un cambio de estado al contrario del actual mediante la pulsación sobre el objeto. Estos métodos resultaron sumamente apropiados para modelar las luces que requería el videojuego. Además, cuestiones referentes a la estética también se pudieron solucionar fácilmente ya que se heredaron métodos para modificar el color y la forma del objeto.

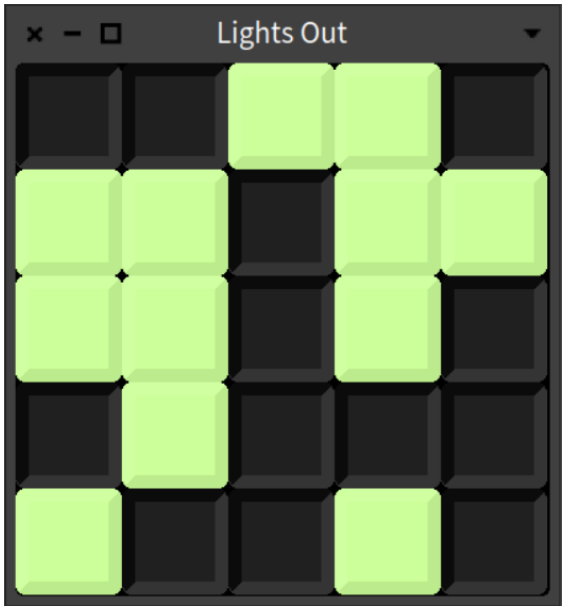

Fig. 7: Un tablero de dimensión 5x5.

Fig. 8: Mensaje que indica que la partida ha terminado exitosamente.

Se tuvieron que sobrescribir métodos para el comportamiento de la pulsación del cursor, con el fin de evitar *bugs* donde las luces no se comportaban conforme a lo diseñado.

Para la aleatoriedad de las luces se recurrió a una función que permite seleccionar valores al azar.

Finalmente, una vez se detecta que el estado de todas las luces que componen el tablero son el de "apagado", se despliega una ventana indicando que el juego ha finalizado mediante el uso de la herramienta *UIManager*. Se continúa felicitando al usuario por haber completado la partida y luego, tras aceptar el cuadro de diálogo, se da por finalizada la partida (ver Fig. 8).

3. Conclusiones

El desarrollo del videojuego nos resultó sumamente desafiante, al no estar nuestro equipo de desarrollo completamente interiorizado con las herramientas que nos provee Pharo Smalltalk. Además, coincidió para todos nosotros que este fue nuestro primer proyecto con una programación orientada a objetos.

Recurrimos a la bibliografía proporcionada por la página oficial de Pharo, donde adquirimos conocimientos complementarios a los dictados en la cátedra, tanto sobre los fundamentos del paradigma de orientación a objetos, como así también de cuestiones referentes a la programación en Pharo Smalltalk. Estos recursos aunados a diversas pruebas de carácter empírico constituyeron la base para la creación del videojuego. El ecosistema de Pharo que permite en tiempo real hacer un seguimiento de la ejecución del programa hicieron que dichas pruebas fueran realmente provechosas.

Bibliografía

Ducasse, S. (2021). *Pharo with Style*.
Ducasse, S. D. (2022). *Pharo 9 by Example*. Paris: BoD - Books on Demand.
Pollet, S. D. (2018). *Learning Object-Oriented Programming, Design and TDD with Pharo*.

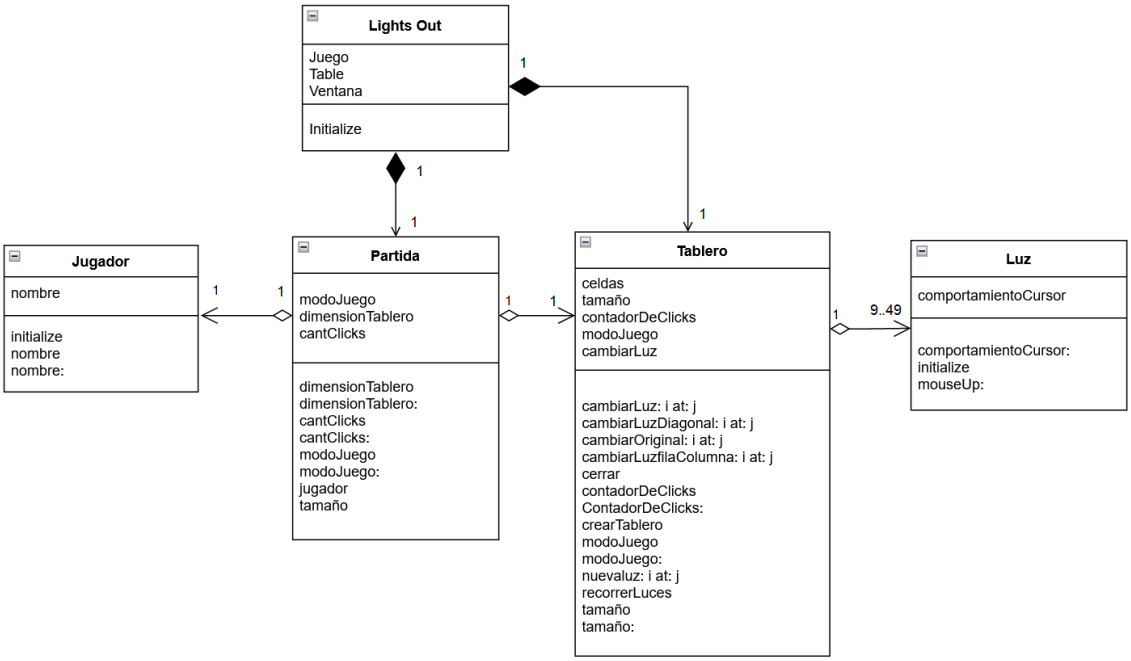

Fig. 9: Diagrama de clases completo.