Trabajo Práctico Especial

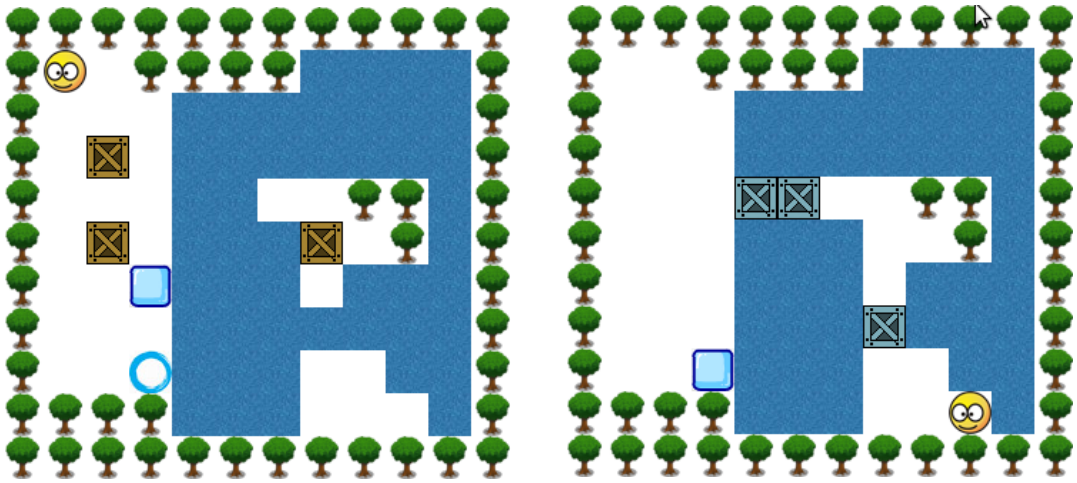
Objetivo

El objetivo del presente trabajo es implementar un juego en el lenguaje *Java*, con interfaz visual, aplicando los conceptos sobre diseño orientado a objetos adquiridos en la materia.

Descripción funcional

La aplicación a implementar es una variante del juego *"Silvesphere"*. El juego consiste en un tablero con diferentes elementos con los cuales el jugador debe interactuar con el objetivo de llegar a una celda destino.

El tablero puede contener zonas con agua en las cuales el jugador pierde si cae sobre ellas. Se pueden empujar cajas hacia el agua con el objetivo de formar un "piso" que le permita al jugador caminar sobre este. Inicialmente puede que la celda destino no aparezca visible. En dicho caso, se debe deslizar un cubo de hielo hacia una celda especial que actúa como interruptor, haciendo visible el destino. A continuación se muestra un ejemplo de un nivel:



(a) (b)

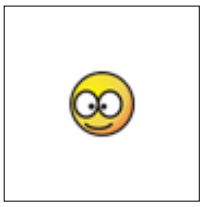
En la figura (a) se muestra el estado inicial del tablero, y en la (b) el estado final al resolverlo. La configuración del tablero está dada por un archivo de texto que especifica el contenido de cada una de las celdas ocupadas. La aplicación deberá procesar estos archivos y generar los tableros para cada nivel.

Elementos del juego

A continuación se detallan los distintos elementos que se pueden ubicar sobre el tablero:

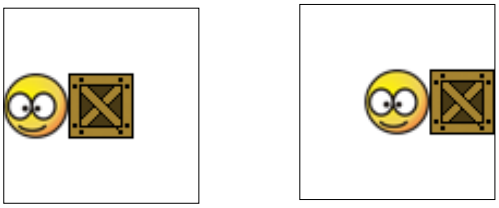
- **Jugador**

  Representa al jugador que se mueve en el tablero. Este se puede desplazar utilizando las flechas del teclado, en dirección vertical y horizontal. Se puede mover hacia una celda vecina vacía, empujar una caja adyacente, deslizar un cubo de hielo, caer sobre agua o posicionarse sobre el interruptor o el destino. El jugador no debe poder escaparse de los límites del tablero. Sólo puede haber un jugador en todo el tablero.



- **Caja**

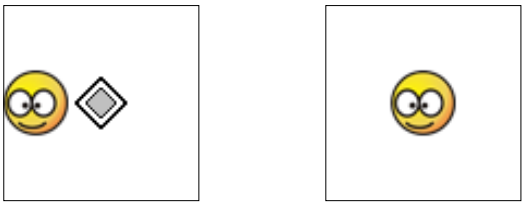
Las cajas pueden ser "empujadas" por el jugador en sentido horizontal y vertical. Sólo pueden ser empujadas si la celda adyacente en la dirección del movimiento está libre, hay agua, un destino o un interruptor (una caja no puede empujar otra caja, un árbol, ni un cubo de hielo).



Dado un tablero como el que se muestra en la figura (a), si el jugador se desplaza a la derecha, empuja la caja en dicha dirección y pasa a ocupar el lugar que ocupaba la caja, como se muestra en la figura (b).

- **Destino**

El jugador debe ubicarse sobre el destino para ganar cada nivel. El destino debe actuar como una celda libre (debe ser posible empujar sobre ella cajas y cubos de hielo). Un nivel no puede contar con más de un destino.

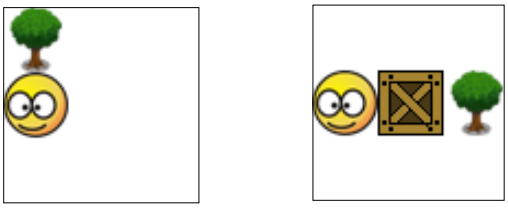


(a) (b)

En el tablero de la figura (a) se muestra al jugador ubicado en una celda adyacente al destino. Si el jugador se desplaza hacia la derecha quedando sobre el destino, gana el nivel como se muestra en la figura (b).

- **Árbol**

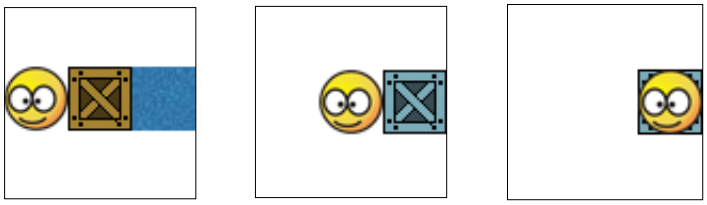
Es un elemento del tablero que no puede ser atravesado por ningún tipo de elemento. En el tablero de la figura (a), el jugador no puede desplazarse hacia arriba porque la celda adyacente en dicha dirección es un árbol. Tampoco en el tablero de la figura (b) puede empujar la caja hacia la derecha ya que la celda adyacente a la caja en dicho sentido es un árbol.



(a) (b)

- **Agua**

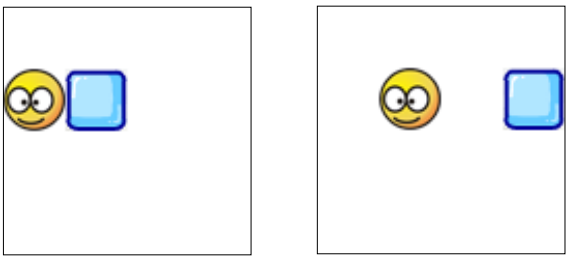
Es una celda sobre la que se hunden los elementos que caen en ella. Si el jugador se ubica sobre la misma, pierde el nivel. Si se empuja una caja, el agua pasa a actuar como una celda vacía permitiendo que el jugador pase por encima de ella, empuje cajas, cubos de hielo. Dado un tablero como el que se muestra en la figura (a), si el jugador empuja la caja esta caerá al agua como se muestra en la figura (b). Luego el jugador podrá volver a moverse hacia la derecha (sin caer al agua y perder), resultando como se muestra en la figura (c).



(a) (b) (c)

- **Cubo de hielo**

Es un elemento que al ser empujado por el jugador se desliza en la dirección del movimiento (horizontal o vertical) hasta tanto se encuentre con algún obstáculo (un árbol, una caja, otro cubo de hielo, etc). Cuando el jugador empuja un cubo de hielo pese a que el cubo pueda desplazarse más de un casillero, el jugador como máximo avanza una celda. Si se tiene un tablero como el que se muestra en la figura (a), si el jugador se mueve hacia la derecha, provocará que el hielo se deslice 2 casilleros, resultando el tablero que se muestra en la figura (b).



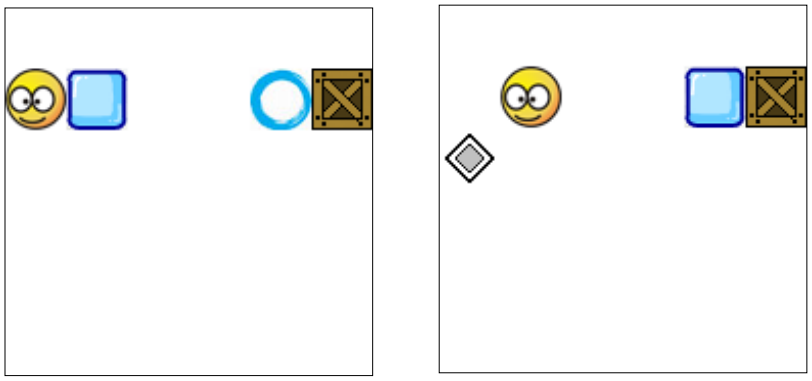
(a) (b)

A diferencia de lo que ocurre con las cajas, si un cubo de hielo cae al agua este se derrite convirtiéndose en agua (y si luego el jugador se mueve hacia el agua pierde el nivel).

- **Interruptor**

Es un elemento que hace visible e invisible el destino del nivel, dependiendo de si haya o no un cubo de hielo sobre este. Si no hubiera un cubo de hielo sobre el interruptor, y en consecuencia el destino estuviera invisible, si el jugador consigue llegar hasta donde está ubicado el destino (invisible), no debe ganar el nivel. Un nivel no puede contar con más de un interruptor.

En el tablero de la figura (a), cuando el jugador hace deslizar el cubo de hielo hacia el interruptor, este hace visible el destino del nivel resultando el tablero de la figura (b).



(a) (b)

Desarrollo del juego

Al ingresar a la aplicación, se le debe mostrar al usuario un menú principal con las siguientes opciones:

1. **Nuevo juego**
2. **Cargar juego guardado**
3. **Salir**

Al seleccionar la opción *"Nuevo juego"* el usuario debe elegir un archivo de tablero para jugar. Los tableros se almacenan en archivos de texto con extensión *".txt"*.

Una vez que se le muestra el tablero cargado al usuario, el mismo puede comenzar a jugar, desplazando con las flechas del teclado al jugador.

Si completa satisfactoriamente el nivel, se le avisa al usuario que ganó y se le vuelve a mostrar el menú principal. En caso de perder, se muestra un cartel indicando esto, y se vuelve al menú principal.

Formato de los archivos de niveles

El archivo del nivel es un archivo de texto que contiene tantas líneas como filas tenga el nivel y cada fila contiene tantos caracteres como columnas tenga también el nivel. Cada carácter determina qué elemento de tablero se encuentra ubicado en cada posición según el siguiente listado:

- "@": Jugador
- "G": Destino
- "#": Agua
- "B": Caja
- "C" Cubo de hielo
- "K": Interruptor
- "T": Árbol

A continuación se muestra un ejemplo de un archivo de nivel, que corresponde al ejemplo mostrado en la sección "Descripción funcional":

```
TTTTTTTTTTT
T@ TTTT####T
T   ########T
T B ########T
T   ##  TT#T
T B ###B T#T
T  C### ###T
T   ########T
T  K###  ##T
TTTT###  G#T
TTTTTTTTTTT
```

Almacenamiento de partidas

En cualquier momento del juego, el usuario puede guardar el tablero actual para luego continuar jugando en otro momento. Al seleccionar esta opción, se le debe consultar el nombre y la ubicación del archivo a guardar (puede tener cualquier nombre y cualquier extensión). **Para almacenar el estado del tablero se debe utilizar serialización.**

Luego, cuando el usuario restaura una jugada, deberá seleccionar el archivo que previamente guardó. La aplicación deberá cargar el tablero especificado en el archivo y continuar el juego desde allí.

Consideraciones de implementación

La cátedra proporcionará un conjunto de clases para simplificar las operaciones visuales relacionadas con el tablero y el manejo de imágenes. **La aplicación debe funcionar con estas clases.**

Adicionalmente se proporcionarán imágenes para los tipos de celdas. Cada grupo puede optar por utilizar estas imágenes o crear las suyas.

El código fuente debe contener comentarios en formato ***Javadoc***.

Se deberán implementar casos de prueba de ***JUnit*** para verificar el correcto funcionamiento de las clases de *backend*. No es necesario testear clases de *frontend*.

Se debe proporcionar un *buildfile* de ***Ant*** en donde el target default genere el *jar* ejecutable de la aplicación (archivo que debe llamarse tpe.jar), la documentación (***Javadoc***) y ejecute los tests. La cátedrá proporcionará un apunte y un proyecto de ejemplo sobre esta herramienta.

Material a entregar

Se deberá entregar un informe que contenga: explicación de la jerarquía diseñada, problemas encontrados durante el desarrollo y decisiones de diseño tomadas.

El código fuente de la aplicación se deberá entregar a través del repositorio de SVN, el cual como mínimo deberá contener los siguientes directorios y archivos:

- ***src***: directorio con los códigos fuente
- ***resources***: directorio con las imágenes utilizadas y cualquier otro archivo necesario (propiedades, configuración, etc.)
- ***levels***: directorio de niveles con archivos de prueba utilizados durante la implementación
- ***build.xml***: buildfile de Ant para poder compilar el proyecto

El repositorio **no** debe contener el proyecto compilado (no entregar archivos ***.class*** ni ***.jar***).

Grupos

El trabajo deberá ser realizado por grupos de hasta 3 (tres) alumnos. **Cada grupo deberá enviar un correo electrónico a la cátedra informado quiénes son los integrantes.**

Se habilitará un repositorio de SVN para cada grupo, que puede ser utilizado durante el desarrollo del trabajo. La entrega final del trabajo se realizará a través de este repositorio (se considerará el último commit antes de la fecha y hora de entrega).

Fecha de entrega

La fecha de entrega es el día jueves 8 de noviembre antes de las 13hs. El informe se podrá entregar hasta el viernes 9 de noviembre a las 13 hs, también mediante el repositorio SVN. Si el trabajo no estuviera aprobado se cuenta con una instancia de recuperación.

Existe la posibilidad de entregar en fecha tardía, el jueves 15 de noviembre. En este caso se aplicará una penalización de 2 (dos) puntos en la nota final del trabajo y no se contará con una instancia de recuperación.

Evaluación

Para la evaluación se tendrá en cuenta la funcionalidad, el estilo del código, el diseño de las clases implementadas, los testeos de unidad implementados y el contenido del informe. Se debe obtener una nota mayor o igual a 4 (cuatro) para aprobar.

Si se entrega un trabajo en fecha tardía que está aprobado, pero luego de aplicar la penalización pasa a estar desaprobado, el trabajo queda finalmente desaprobado.

Consultas

Las consultas se realizarán en el horario del laboratorio, o mediante correo electrónico a la dirección **poo@it.itba.edu.ar**