Práctica 1

Iniciación a la programación en Android y Java

Objetivo: Implementar un juego de nonogramas para móvil Android y escritorio usando el lenguaje de programación Java

1. Nonogramas:

El nonograma, también conocido como Hanjie, Picross o Griddlers en el Reino Unido, es un rompecabezas que consiste en colorear las celdas correctas de una cuadrícula, de acuerdo con los números a los lados de la misma, con el fin de revelar una imagen oculta. En este tipo de rompecabezas, los números miden cuántos cuadros rellenos contiguos hay en una fila o columna dada. Por ejemplo, una pista de "4 8 3" significa que hay grupos de cuatro, ocho, y tres cuadros rellenos, en ese orden, con al menos un espacio en blanco entre los grupos sucesivos.

Estos rompecabezas son a menudo en blanco y negro, y describen una imagen binaria, aunque también pueden ser a color. En caso de serlo, las pistas numéricas también son coloreadas para indicar el color de los cuadros. Dos números de diferentes colores pueden tener o no un espacio entre ellos. Por ejemplo, un cuatro negro seguido por un dos rojo podría significar cuatro cuadros negros, algunos espacios en blanco, y dos cuadros rojos, o podría ser simplemente cuatro cuadros negros seguidos inmediatamente por dos rojos. Los nonogramas no tienen un límite teórico en su tamaño, y no están restringidos a diseños cuadrados.

Algunos ejemplos:

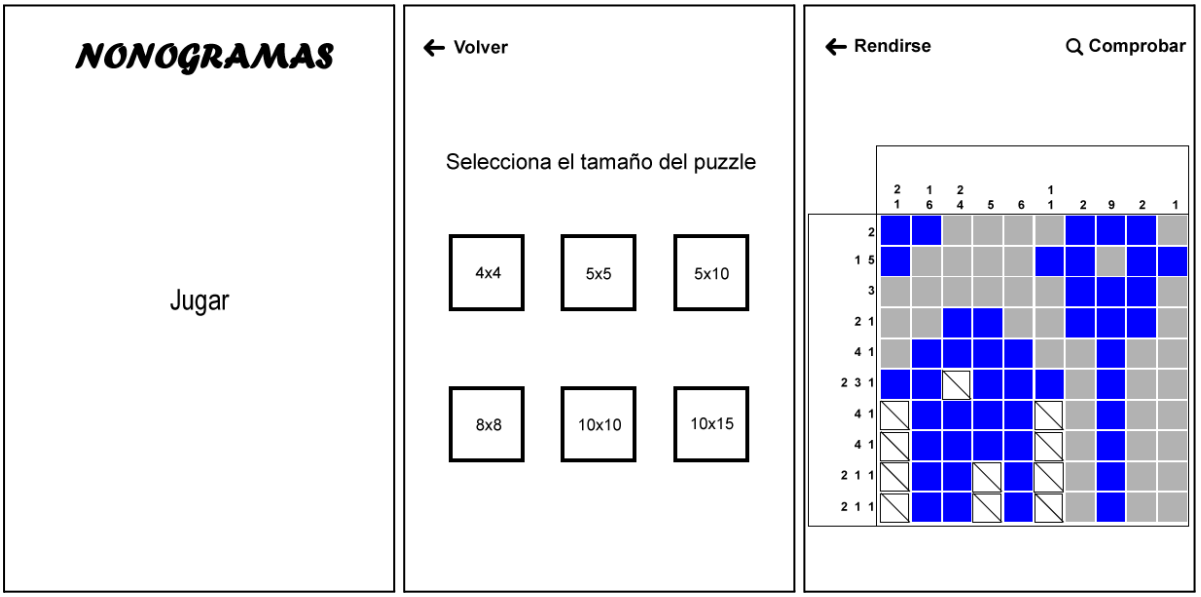
- Nonogram - Android
- Nonogram - IOS
- Nonogram - Switch
- Nonogram - Web

2. Descripción del juego a implementar:

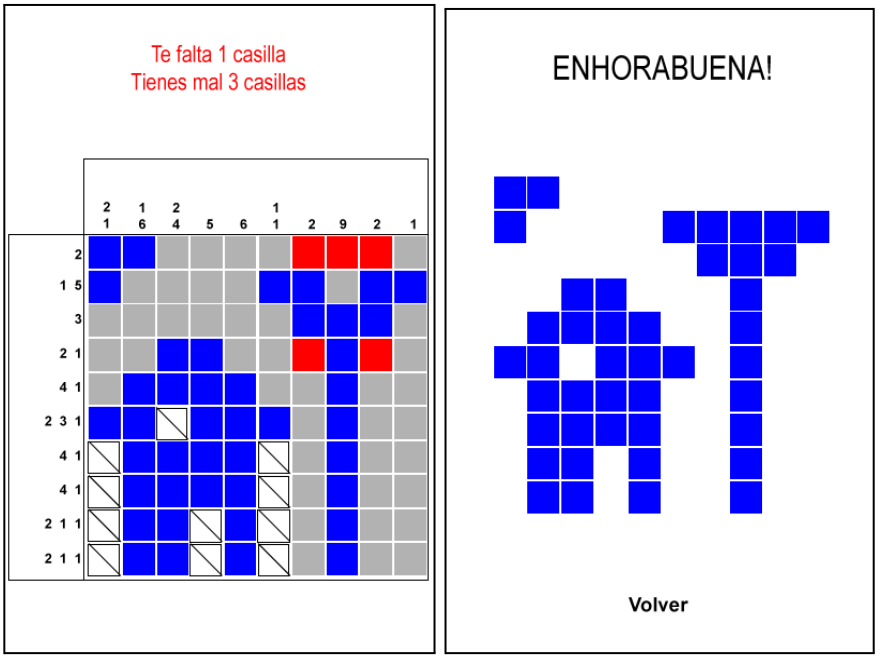
Pantallas de juego

Al ejecutarse, el juego mostrará una pantalla de bienvenida con el logotipo de la aplicación y un “menú” con una única opción de “Jugar”. Tras el menú, se permite seleccionar el tamaño del tablero. Una vez escogido el tamaño se muestra finalmente el

estado inicial del tablero que el usuario deberá completar. Si el jugador rellena todas las casillas correctamente, un instante después de completar el tablero, el juego felicita al usuario y termina la partida. Si el jugador tiene problemas podrá pulsar el botón “comprobar” con lo que el juego resaltará durante un periodo de tiempo las casillas que están mal indicadas por el usuario como parte de la solución y también mostrará un mensaje con el número de casillas que faltan para llegar a la solución y con el número de casillas mal indicadas.



De izquierda a derecha: Pantalla de título; Pantalla de selección de nivel; Pantalla de juego



De izquierda a derecha: Pantalla de juego al pulsar “Comprobar”; Pantalla al superar el nivel.

Tamaño de puzzles

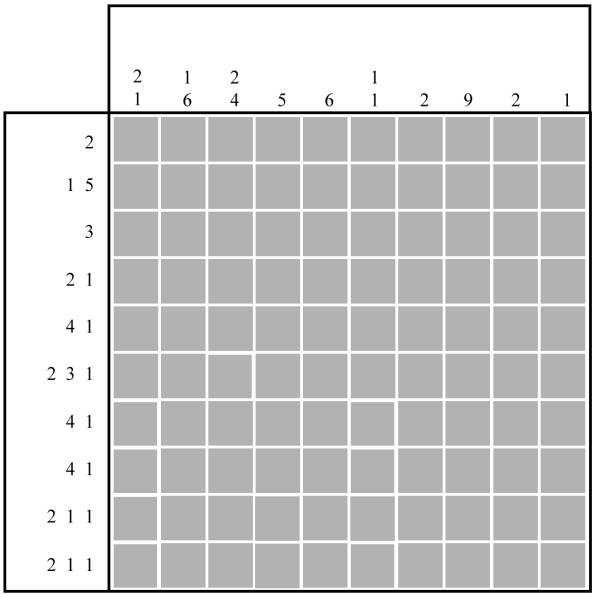
En la pantalla de selección de nivel el juego se deben dar al menos las siguientes opciones de tamaño:

- 5x5
- 8x8
- 10x10

Se valorará implementar otros tamaños de tablero (que tengan sentido en el juego), sobre todo si son tableros con distinto número de filas y columnas.

El tablero

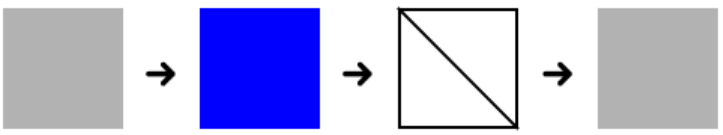
El tablero está formado por un conjunto de celdas (tantas como el tamaño del tablero seleccionado) y los números que hacen de instrucciones para resolver el puzzle. Por ejemplo, en un puzzle 10x10, habrá 100 celdas, 10 columnas de números en la parte superior y 10 filas de números a la izquierda.



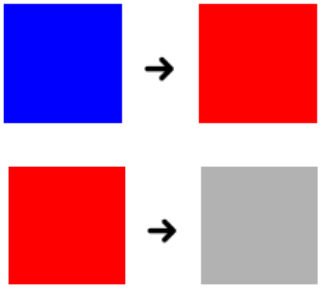
Estas celdas pueden tener 4 estados diferentes:

- Celda gris, es el estado por defecto de las celdas.
- Celda azul, representa que la celda es parte de la solución.
- Celda blanca con borde negro y atravesada por una línea, representa que la celda no forma parte de la solución.
- Celda roja, indica que una celda marcada como parte de la solución (azul) no lo es.

El jugador puede cambiar el estado de las celdas pulsando sobre ellas. El orden en el que cambia es el siguiente:



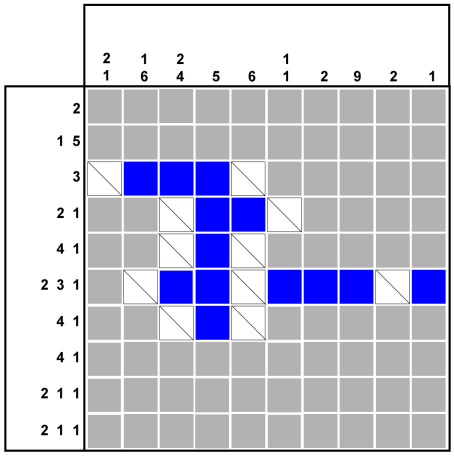
Además una celda solo se volverá roja si estando azul se pulsa sobre el botón “comprobar” y la celda no forma parte de la solución correcta. Una celda roja volverá a ser gris si se pulsa sobre ella o pasa un periodo de tiempo.



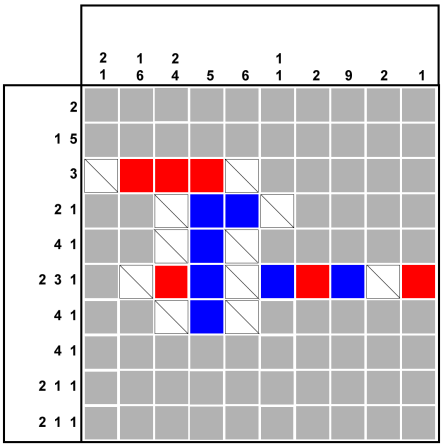
Comprobar el tablero

Durante una partida, el jugador puede pulsar en cualquier momento el botón “Comprobar” para conocer cómo de cerca está de completar el nivel.

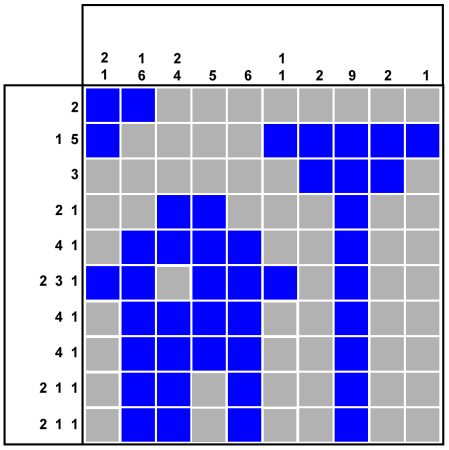
Al pulsar este botón se deben resaltar solo aquellas celdas en azul que no forman parte de la solución correcta. En la siguiente figura se puede ver un ejemplo:



Tablero a medio resolver

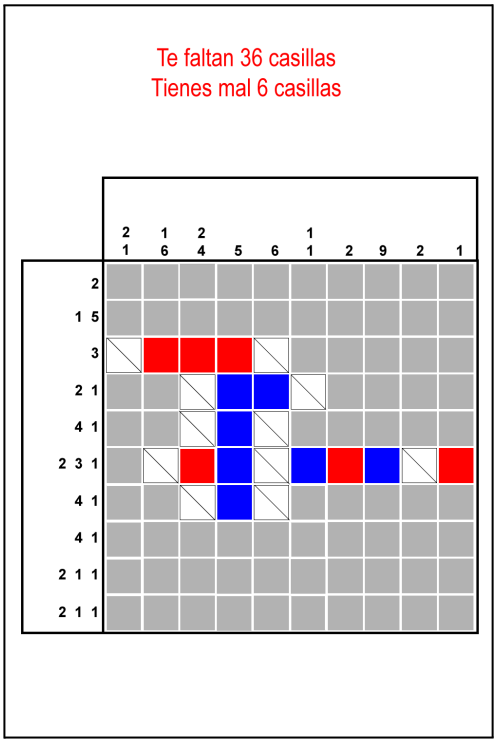


Tablero al pulsar comprobar



Tablero solución

Además de las celdas en rojo, se mostrará al jugador el número de celdas qué están en azul y no son correctas así como el número de celdas que no siendo azules sí deberían serlo. Para el caso anterior la vista sería la siguiente:



El estado de las celdas en rojo así como el mensaje debe mantenerse solo un periodo de tiempo determinado, volviendo el tablero al estado anterior al pulsar comprobar y desapareciendo el mensaje mostrado.

Completar el nivel

El nivel se supera si todas las celdas azules marcadas por el jugador son parte de la solución y no falta ninguna por marcar. No es necesario que las casillas que no son parte de la solución se marquen como casillas blancas para completar el puzzle. Una vez completado el puzzle el juego debe mostrar un mensaje de enhorabuena y el tablero solucionado, sin los números que dan las instrucciones de como rellenarlo y centrado en la pantalla.

3. Requisitos de la implementación

El juego se implementará en Java, y deberán poderse generar dos versiones diferentes, una para Android y otra para sistemas de escritorio (Windows, GNU/Linux o Mac). El código deberá estar correctamente distribuído en paquetes, ser comprensible y estar suficientemente documentado.

Para el desarrollo se hará uso de Android Studio, utilizando un único proyecto con varios módulos. La mayor parte del código deberá ser común y aparecer una única vez, compartido entre ambas versiones. Deberá existir así una capa de abstracción de la plataforma, que se implemente dos veces, una para Android y otra para Escritorio. La implementación del juego deberá hacer uso únicamente de la capa de abstracción y de las funcionalidades del lenguaje que estén disponibles en ambas plataformas.

Dado que el punto de entrada de la aplicación es diferente en cada plataforma (en Android se necesita una Activity y en escritorio un main()) se permitirá también la existencia de dos módulos distintos (minimalistas) de “arranque del juego”.

La aplicación se debe adaptar a cualquier resolución de pantalla. Independientemente de su relación de aspecto, el juego no deberá deformarse, sino aparecer centrado en la pantalla con “bandas” arriba y abajo o a izquierda y derecha. A modo de ejemplo, la siguientes figuras muestran cómo debe verse si se ejecutara en un móvil en vertical con una resolución de 2220 × 1080 (relación 18.5:9) y en una pantalla de ordenador. La lógica espera una relación de 2:3 (por ejemplo 400 × 600) por lo que se añaden bandas para que la zona de juego quede centrada.

Las bandas se muestran en gris, pero a la hora de realizar la práctica estas deben ser del color del canvas.

El juego además debe reproducir una música ambiente desde que entramos al menú principal y ejecutar sonidos cortos cuando se cambia una celda de estado y se completa un nivel.

Los niveles de juego serán generados de manera totalmente aleatoria (con restricciones para que los niveles tengan sentido, como puede ser evitar niveles con todas o ninguna de las casillas como solución) o generado a través de un conjunto de niveles predefinidos. Cómo se generan los niveles es decisión de cada grupo pero deben tener sentido como nivel de juego y ser entretenidos.

4. Consejos y pistas a la hora de implementar

Para abstraer la plataforma, podéis definir los siguientes interfaces:

- `Image`: envuelve una imagen de mapa de bits para ser utilizada a modo de sprite:
  - `int getWidth()`: devuelve el ancho de la imagen.
  - `int getHeight()`: devuelve el alto de la imagen.
- `Font`: envuelve un tipo de letra para ser utilizado al escribir texto.
- `Graphics`: proporciona las funcionalidades gráficas mínimas sobre la ventana de la aplicación:
  - `Image newImage(String name)`: carga una imagen almacenada en el contenedor de recursos de la aplicación a partir de su nombre.
  - `Font newFont(filename, size, isBold)`: crea una nueva fuente del tamaño especificado a partir de un fichero `.ttf`. Se indica si se desea o no fuente en negrita.
  - `void clear(int color)`: borra el contenido completo de la ventana, rellenándolo con un color recibido como parámetro.
  - Métodos de control de la transformación sobre el canvas (`translate(x,y)`, `scale(x,y)`; `save()`, `restore()`). Las operaciones de dibujado se verán afectadas por la transformación establecida.
  - `void drawImage(Image image, ...)`: recibe una imagen y la muestra en la pantalla. Se pueden necesitar diferentes versiones de este método dependiendo de si se permite o no escalar la imagen, si se permite elegir qué porción de la imagen original se muestra, etc.
  - `void setColor(color)`: establece el color a utilizar en las operaciones de dibujado posteriores.
  - `void fillSquare(cx, cy, side)`: dibuja un cuadrado relleno del color activo.
  - `void drawSquare(cx, cy, side)`: dibuja un cuadrado sin relleno y con el borde del color activo.
  - `void drawLine(initX, initY, endX, endY)`: dibuja una línea recta del color activo.
  - `void drawText(text, x, y)`: escribe el texto con la fuente y color activos.
  - `int getWidth(), int getHeight()`: devuelven el tamaño de la ventana.
- `Input`: proporciona las funcionalidades de entrada básicas. El juego no requiere una interfaz compleja, por lo que se utiliza únicamente la pulsación sobre la pantalla (o click con el ratón).

    - `class TouchEvent`: clase que representa la información de un toque sobre la pantalla (o evento de ratón). Indicará el tipo (pulsación, liberación, desplazamiento), la posición y el identificador del "dedo" (o botón).
    - `List<TouchEvent> getTouchEvents()`: devuelve la lista de eventos recibidos desde la última invocación.
- `Sound`: Envuelve los sonidos
- `Audio`: proporciona la funcionalidad para gestionar y reproducir sonidos.
    - `Sound newSound(string file)`
    - `Sound playsound(string id)`
- `Engine`: interfaz que aglutina todo lo demás. Puede ser el encargado de mantener las instancias:
    - `Graphics getGraphics()`: devuelve la instancia del "motor" gráfico.
    - `Input getInput()`: devuelve la instancia del gestor de entrada.
    - `Audio getAudio()`: devuelve la instancia del gestor de sonidos

Para independizar la lógica de la resolución del dispositivo (o de la ventana) podéis ampliar la clase `Graphics` para que reciba un tamaño lógico (de "canvas") y que todas las posiciones se den en ese sistema de coordenadas. También podéis plantear el desarrollo de clases adicionales que proporcionen un mayor nivel de abstracción. En particular, para facilitar la puesta en marcha de la aplicación en cada plataforma, es posible que queráis ampliar `Engine` para incorporar la idea de estado de la aplicación.

Al ser interfaces, observad que no se indica cómo se crearán las instancias. Así, por ejemplo, la clase que implemente `Graphics` para la versión de escritorio podría necesitar recibir en su constructor la ventana de la aplicación, y la versión de Android el `SurfaceView` y `AssetManager`. La puesta en marcha de la aplicación tendrá que ser diferente en cada plataforma y estar encapsulada, en la medida de lo posible, en los módulos correspondientes. No obstante, la carga de los recursos no debe programarse dos veces, y debe formar parte del módulo de lógica.

Para implementar la lógica, cread un "modelo" del juego con clases para las celdas, el tablero o las pistas que sean independientes de la representación en pantalla. Esas clases deberían poder usarse para implementar el mismo juego sin interfaz gráfica (para un terminal). Luego implementad clases que se encarguen de la representación visual de esos elementos (la "vista").

Si en la versión para móvil tenéis problemas de rendimiento, minimizad la creación de objetos (analizad la posibilidad de cachearlos con pools) y usad, si es posible, superficies hardware (`SurfaceHolder::lockHardwareCanvas()`).

5. Opcional

Siempre que los requisitos básicos de la práctica funcionen correctamente y estén bien implementados, se valorará positivamente la incorporación de características adicionales como por ejemplo:

- Fade In - Fade Out de las pantallas al transicionar de una a otra. También otro breve al cambiar el estado de las celdas de un tablero.
- En la versión de escritorio:
    - Uso de pantalla completa.

    - Pausa del juego (sin consumir recursos) cuando la aplicación pierda el foco.
- Opción de niveles en los que el tablero no sea cuadrado.

6. Instrucciones de entrega

La práctica debe entregarse utilizando el mecanismo de entregas del campus virtual, no más tarde de la fecha y hora indicadas en el propio campus.

Sólo un miembro del grupo deberá realizar la entrega, que consistirá en un archivo .zip con el proyecto completo de Android Studio eliminando los ficheros temporales. Se añadirá también un fichero **alumnos.txt** con el nombre completo de los alumnos y un pequeño **.pdf indicando la arquitectura de clases y módulos de la práctica y una descripción de las partes opcionales y extras desarrolladas**, si ha habido alguna.

Para aprobar la práctica es obligatorio que la representación se **adapte al tamaño de la ventana/pantalla**. También es necesario que no haya errores de compilación para aprobar.

7. Evaluación de la práctica

Es obligatorio que la aplicación se ejecute en Android y en Escritorio adaptándose a diferentes resoluciones y manteniendo la relación de aspecto. A la hora de corregir la práctica se tomará como referencia la configuración y software de los laboratorios. Los pesos en la evaluación de la práctica serán los siguientes:

- Funcionalidad y código - 85%. En este apartado entran por ejemplo la arquitectura multiplataforma planteada, las funciones creadas, limpieza de código, nombres usados, código comentado, estructuras de datos utilizadas, consumo de recursos….
- Apartado gráfico y experiencia de usuario - 15%. Fluidez del juego, coherencia de la interfaz, de los textos y del sonido…

Podrá pedirse a los miembros de un grupo que defiendan la práctica frente a los profesores de la asignatura de manera individual para comprobar el trabajo realizado. Si durante la defensa no se demuestran conocimientos sobre el trabajo presentado el miembro correspondiente del grupo puede obtener menos nota que sus compañeros, llegando a suspender..

Copiar código de la práctica conlleva el suspenso directo de la asignatura.

Bibliografía

- Beginning Android Games, Third Edition, Mario Zechner and J. F. DiMarzio, Apress, 2016.

- Developing games in Java, David Brackeen, Bret Barker, Lawrence Vanhelsuwe, New Riders.