Universidad de Alcalá

Paradigmas Avanzados de Programación

Enunciado PECL2

Cundy Crosh Soga

CURSO 2022/2023 – CONVOCATORIA ORDINARIA

Versión: 2 (última revisión, 16 de marzo de 2023).

Índice

1. El Videojuego

El juego "*Cundy Crosh Soga*" que se propone para la PECL1 se basa en el popular juego de móviles "*Candy Crush Saga*", un videojuego multidioma desarrollado por la compañía británica King, originalmente disponible como una aplicación de Facebook y adaptado para los sistemas operativos Android, iOS y Windows Phone, que se lanzó el 12 de abril de 2012 (fuente: Wikipeadia).

1.1. Dinámica

El tablero es una malla formada por cuadrados ("Bloques") que cubren todo el tablero.

Los bloques pueden tomar 6 estados, que son relacionados con 6 colores.

Cada vez que se comienza un nivel se recibe un tablero nuevo.

El objetivo de este juego es ir eliminando bloques para lo cual, si hay dos bloques del mismo color uno al lado del otro, al tocarlos permiten hacer una coincidencia y eliminarlos.

Cuando los bloques se eliminan son cubiertos por los bloques que se encuentran por encima. Posteriormente, se procede a rellenar los huecos con bloques aleatorios.

Si hay cinco bloques del mismo color, uno al lado del otro, cuando lo toca obtiene una bomba. Cuando lo toca, borra todos los bloques en esa fila o columna de forma aleatoria.

Cuando se obtiene seis bloques del mismo color muy juntos, el jugador obtiene un bloque TNT. Cuando lo toca, todos los bloques que lo rodean explotarán (un radio de 4 elementos).

Cuando se obtienen siete o más bloques del mismo color, uno al lado del otro, el jugador obtiene un bloque de rompecabezas. Tendrá un color específico (aleatorio). Cuando se toque, borrará todos los bloques de ese color.

Cada movimiento que no elimina bloques resta una vida. Para el juego cada jugador dispondrá de cinco vidas.

2. Trabajo a realizar

2.1. Implementación básica (obligatoria)

Este apartado es obligatorio, y consiste en la implementación del juego utilizando las técnicas básicas de programación funcional con Scala.

1) Implementar el Juego empleando Scala (con IntelliJ IDEA) para actualizar el estado del juego.
2) La salida será vía consola:
   - Bloque "1" representa el color azul
   - Bloque "2" representa el color rojo
   - Bloque "3" representa el color naranja
   - Bloque "4" representa el color verde
   - Bloque "5" representa el color marrón
   - Bloque "6" representa el color amarillo
   - En función de los bloques eliminados se crearán bloques especiales (su funcionalidad se activa al ser seleccionados y no computan para generar nuevos bloques especiales):
     i) Bloque bomba ("B") que borra línea o borra columna.
     ii) Bloque TNT ("T"), que elimina todos los bloques adyacentes (en un radio de 4 elementos).
     iii) Bloque rompecabezas ("R"), generado con un color aleatorio (mostrados como "R1", "R2" …), que elimina todos los bloques de ese color del tablero.
3) A la hora de ejecutar la práctica, se deberán pedir los siguientes parámetros desde teclado:
   - Ejecución
     i) <m>: Ejecución manual. El programa cada vez que se realicé una iteración esperará a que el usuario pulsé una tecla. Para que indique la [fila,columna] sobre la que quiere interactuar.
     ii) <a>: Ejecución automática. El programa no tiene interacción con el usuario. Interactúa con el juego de forma aleatoria, seleccionando una posición del tablero al azar en cada ronda.
   - El nivel de dificultad,
     i) <1> Fácil. Tomará bloques de los colores 1,2,3 y 4.
     ii) <2> Difícil. Tomará bloques de los colores 1,2,3, 4, 5 y 6
   - Tamaño del tablero
     i) <numColumnas> Indicará el número de columnas del tablero.
     ii) <numFilas> Indicará el número de filas del tablero.
   - Ejemplo: cundycroshsaga –a 2 10 50: Indica que el programa se ejecutará de forma automática, difícil y en una matriz de 10x50.
   - Alternativamente, estos parámetros se pueden pedir por consola al iniciar el juego.
4) El tablero se supondrá finito, con unas dimensiones iniciales que no varían a lo largo del juego.
5) Todo el código entregado deberá estar perfectamente comentado. Aquella práctica que no tenga el código bien documentado puede ser suspendida.
6) Evitar que las inicializaciones de la matriz tablero sean siempre las mismas.

2.2. Implementación optimizada (obligatoria)

De forma avanzada, se piden modificaciones sobre la implementación básica para hacer uso de las técnicas de optimización que se puede obtener de la programación funcional:

7) Consiste en implementar un algoritmo que facilite la opción de juego más óptima al modo de ejecución automático en cada jugada que realice.
   - En vez de elegir una celda aleatoria del tablero, deberá elegir la celda que proporcione la mejor estrategia. Es decir, que borre un mayor número de celdas
   - Se pueden usar, para este punto únicamente, las Colecciones Paralelas. Si se considera oportuno para mejorar la velocidad de ejecución.

2.3. Implementación Gráfica (extra)

Como apartado extra se pide realizar la implementación con interfaces gráficas. Es un apartado completamente opcional, y sirve para obtener puntos extra.

8) A aquellos alumnos que implementen el juego utilizando para ello las librerías gráficas disponibles en Scala, podrán obtener puntos extra, aunque será necesario haber realizado cada una de las partes requeridas en la parte básica y de optimización.
   - Scala proporciona una librería propia, scala-swing (https://index.scala-lang.org/scala/scala-swing), para realizar GUIs. Es un warp de Java Swing. Aquí podeis encontrar un tutorial interesante: https://otfried.org/scala/gui.html.

3. Operaciones limitadas y prohibidas con Scala

Se debe tener en cuenta que, para la realización de los programas en Scala, se deben usar vals para la definición de variables, objetos inmutables y métodos sin efectos colaterales. Con limitadas excepciones, y previa autorización de los profesores, se podrán usar vars, objetos mutables y efectos colaterales.

En la medida de lo posible, se utilizarán listas. En el caso de utilización de otros elementos (Sets o Maps), deberá justificarse por qué no ha podido utilizarse listas.

Operaciones **prohibidas** (puede usarse funciones que hagan esa funcionalidad, pero deberán implementarse con funciones propias del alumno):

- Utilización del operador de concatenación de listas (::: o ++).
- Operaciones de las listas, como reverse(), length(), last() e isEmpty().
- Otras operaciones, como filter(), flatten(), flatMap() y operaciones derivadas de reverse (como, por ejemplo, reverseMap(), reverse_:::).
- No se pueden usar bucles (for, while y similares) para recorrer elementos.
- Colecciones mutables.

4. Prerrequisitos de entrega

Es requisito indispensable haber entregado las actividades semanales en la plataforma gamificada (http://plataformagamificacion.cc.uah.es/pap), en fecha y hora anunciada cada semana en las sesiones de Laboratorio.

En concreto, las actividades a entregar son:

- Tarea 4.1 – Ejercicios I
- Tarea 4.2 – Ejercicios II
- Tarea 4.3 – Ejercicios III
- Tarea 4.4 – Ejercicios IV

Según la Guía Docente de la Asignatura, se otorgará la calificación de "*No presentado*'' al alumno que, habiendo optado por el procedimiento de evaluación continua, no se presenten a la evaluación de todos los instrumentos de Evaluación. Por tanto, la no presentación de todas las entregas semanales supondrá un "*No Presentado*" en la convocatoria actual.

Nota: No se admiten entregas semanales manifiestamente erróneas para cumplir con los prerrequisitos de la PECL.

5. Documentación a entregar

La entrega de la PECL1 deberá constar de:

- Proyectos de IntelliJ IDEA con la solución codificada.
  - 1x Proyecto de implementación básica.
  - 1x Proyecto de implementación optimizada (ejercicio 7)
  - (extra) (opcional) 1x Proyecto de implementación gráfica
- Breve video comentando la ejecución
  - 1x Video por cada proyecto aportado
  - Los videos no superarán los 3 minutos de duración. Deben participar ambos miembros de la pareja de trabajo. Se comentarán los detalles clave de la implementación y demostración de ejecución.
- Breve memoria PDF comentando los detalles clave de la implementación.

6. Forma de entrega

El trabajo se realizará en parejas. Se entregará mediante el correspondiente Buzón de Entrega disponible en BlackBoard. La fecha tope de entrega y fecha de Defensa se encuentra disponible en BlackBoard. Al ser un trabajo en pareja, solamente es necesario que uno de los miembros haga la entrega.

Nota: No se admitirán entregas enviadas al correo electrónico (o mensajería interna de BlackBoard) de los profesores. Únicamente se permite la interacción con el Buzón de entrega.

7. Defensa

La defensa de las prácticas será en la fecha y la forma indicada por el/la profesor/a del laboratorio, pudiendo ser esta oral o escrita. En el caso de no contestarse correctamente a las cuestiones presentadas, la práctica presentada podrá considerarse como suspensa.

8. Evaluación

La nota máxima a la que puede acceder se pondera en función de:

a) Calidad del material entregado por el alumno.
b) Documentación de seguimiento presentada durante su realización
c) Realización e implementación de la práctica.
d) Defensa e implementación de las modificaciones solicitadas.
e) Manejo del software utilizado para el desarrollo de la práctica.
f) Código desarrollado.

La puntuación de cada apartado es el siguiente:

Sección	Nota
Implementación básica (obligatoria)	5 pts. Máx.
Implementación optimizada (obligatoria)	3 pts. Máx.
Implementación Gráfica (extra)	(máximo 3 puntos extra)
Documentación entregada	2 pts. Máx.
Defensa	1x, 1/2x o 0x a la nota de la PCL1