Complementos de Programación.

Práctica 8: Comecocos

Manuel Sánchez López DNI: 77137132-P

Índice

1. Introducción.

Esta práctica se basa en el desarrollo del juego del comecocos. Para su realización nos hemos basado en la práctica anterior, orientada a la implementación del juego del tetris. Por lo tanto, tomando como fundamento dicho juego, vamos a desarrollar el "*pacman*".

2. Diagrama de clases UML.

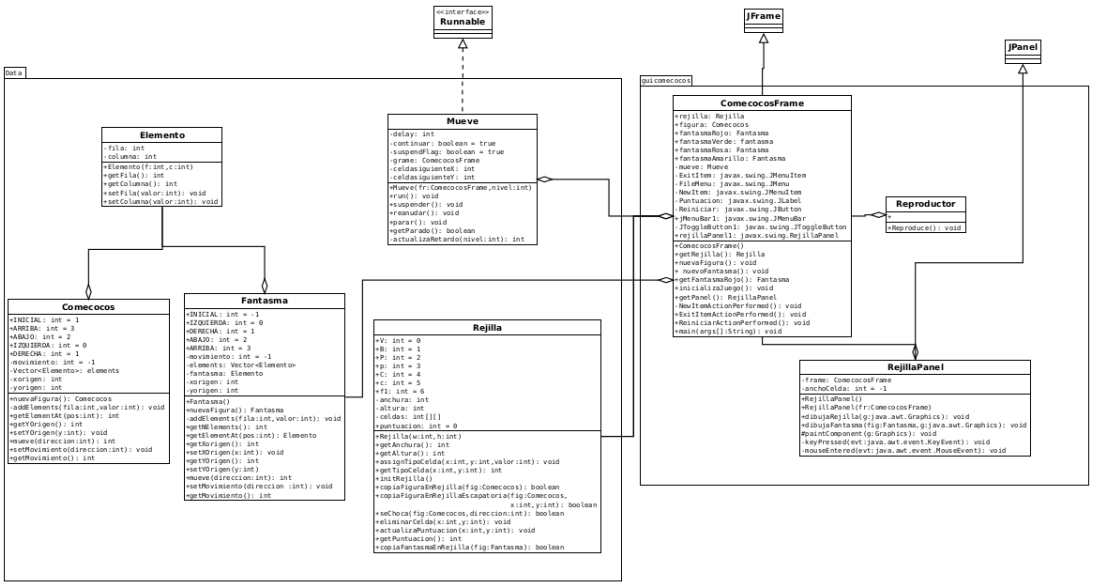

Figura 1: Diagrama de clases UML.

El diagrama viene incluido en PDF en el paquete comprimido.

3. Paquetes.

Nuestro proyecto se fundamentará en dos paquetes diferentes; el primero será el de "data" y contendrá tanto las clases para crear el *comecocos* como los *fantasmas* (mejora propuesta) y nos permitirá tanto sentar las bases tanto de la rejilla como del movimiento de las diferentes figuras. Por otra parte tendremos el paquete "guicomecocos", en el cual se implementará la ventana del juego y contendrá el ejecutable para iniciarlo.

3.1. data.

Como hemos introducido, mediante este paquete definiremos los diferentes elementos que compondrán nuestro juego y el movimiento de las distintas

figuras sobre la rejilla, la cual también vendrá impuesta en una de las clases contenidas en el presente paquete.

3.1.1. Comecocos.

En esta clase nos encargaremos de crear el "comecocos" e indicar tanto su posición como el movimiento que se realizará dependiendo de la acción seleccionada.

En primer lugar definimos los distintos posibles movimientos que podrá realizar nuestra figura y le asignamos distintos valores "int". De esta forma podremos identificarlos a la hora de realizar la acción.

Crearemos también un par de variables que nos indiquen la posición de nuestra figura en la rejilla para así poder controlarla.

El Comecocos será un vector de elementos, en este caso sólo compuesto por una posición de la matriz que representará la rejilla.

Dentro del método "nuevaFigura()" crearemos nuestra figura y la posicionaremos en la posición de origen correspondiente de nuestra matríz. En este caso, en las posiciones x=14 e y=17, que será donde se coloque cada vez que se inicie el juego nuestro "comecocos".

Para saber la posición exacta donde se encuentra nuestra figura crearemos los distintos métodos que la referencien.

Ahora implementamos un método de gran importancia; el que nos devolverá la dirección que sigue nuestra figura. Como sabemos, nuestro "comecocos" se podrá mover en todas las direcciones siempre y cuando no haya un bloque ó un fantasma que lo impida. Por lo tanto, deberemos indicarlo en dicha función. Además, sabemos que existe un "pasadizo" en nuestra rejilla que permite pasar al "comecocos" desde el lateral izquierdo al derecho y viceversa, por lo que también deberemos tenerlo en cuenta para la implementación. Cabe destacar que, al iniciar el juego, la figura permanecerá quieta hasta que el usuario seleccione un movimiento. Para ello crearemos una posición "INICIAL".

Por último, creamos dos métodos que sean capaces de captar el movimiento actual y devolverlo cuando se solicite el mismo (getMovimiento() y setMovimiento(int direccion)).

3.1.2. Elemento.

Esta clase representará una celda en la que se situará el comecocos. El elemento vendrá definido por una fila y una columna para determinar su

posición. Con esta clase podremos determinar la posición en cuanto a filas y columnas de la matriz generada.

3.1.3 Fantasma.

En esta clase nos encargaremos de crear a nuestros “fantasmas” y sus posibles movimientos tal y como hicimos con el “comecocos”.

3.1.4 Mueve.

Esta clase implementará una hebra que nos permitirá mover la figura de forma automática hacia la dirección seleccionada por el usuario sin tener que mantener pulsada la tecla. Dicho movimiento se realizará siempre y cuando no se produzcan choques contra bloques u otras figuras.

Para su implementación tendremos que poder tanto suspender como reanudar el movimiento, saber cuál será la siguiente celda a visitar para saber si se puede realizar el movimiento y conocer como está compuesto nuestro recorrido.

El constructor de esta clase inicializará la referencia utilizada por la hebra al ComecocosMidlet, establece el retardo en milisegundos entre los distintos movimientos y comenzará a ejecutar la hebra actual.

En esta clase nos encargaremos de que el comecocos esté en continuo movimiento salvo cuando choque contra un muro. También procederá a refrescar la pantalla donde se va dibujando todo y los puntos que se van consiguiendo serán también actualizados aquí. Además se controlarán los posibles choques contra los fantasmas para retornar al punto inicial en caso del mismo.

3.1.5 Rejilla.

Esta clase representa la rejilla con una determinada anchura y altura, en la que cada celda puede estar VACÍA(V), contener un BLOQUE(B), contener un “Punto Pequeño”(p), contener un “Punto Grande”(P), al COMECOCOS(C) ó a un FANTASMA (f).

El constructor de esta clase se encargará de crear el espacio para crear la rejilla, que vendrá definida por un conjunto de celdas de determinada anchura y altura.

3.2 guicomecocos.

En este paquete incluiremos tres diferentes clases; una para ejecutar el programa, otra para crear la ventana con la rejilla y demás aspectos incluidos

y una última que nos permitirá incluir la música del comecocos a nuestro juego.

3.2.1 ComecocosFrame.

Esta clase será la encargada de ejecutar el programa. En ella inicializaremos la rejilla y colocaremos en ella los distintos elementos que se necesiten. Además daremos comienzo a la hebra que permite realizar el movimiento.

Necesitaremos un método que nos devuelva la figura en el lugar donde se encuentra en el preciso instante, además de un método que coloque el comecocos en su posición inicial (nuevaFigura).

Por otro lado necesitaremos asignar a cada botón incluido su función, ya sea iniciar, reiniciar, parar/reanudar ó salir del juego.

Cuando se reinicie la puntuación también volverá a cero.

3.2.2 RejillaPanel.

En esta clase fijamos cual es el ancho de la celda y dibujamos en ella los distintos componentes.

Con la clase "dibujaRejilla" rellenaremos la misma con los diferentes elementos definidos en la clase "Rejillla" definida en el paquete "data". Por lo tanto, según corresponda, colocaremos un bloque, un punto pequeño, un punto grande, una celda, el comecocos ó un fantasma. Para que se ajuste a nuestras medidas deberemos aplicar un offset a las distintas componentes de posición.

En esta clase se nos permite dar color y forma a cada elemento, por lo que es de gran utilidad a la hora de crear las diferentes figuras.

Además, en esta clase llamamos al método "setMovimiento" para ir analizando hacia donde quiere el usuario que vaya el comecocos y poder dibujar el "comecocos" donde corresponda.

3.2.3 Reproductor

En esta clase hemos creado un reproductor de sonido, incluyendo en el paquete la canción del comecocos para poder ser reproducida. Por lo tanto, el método "Reproduce" utiliza dicho archivo ".waw" para poner música mientras jugamos.

4. Mejoras implementadas.

4.1 Representación de puntos.

Esta mejora se basa en representar puntos pequeños y puntos grandes en las celdas correspondientes.

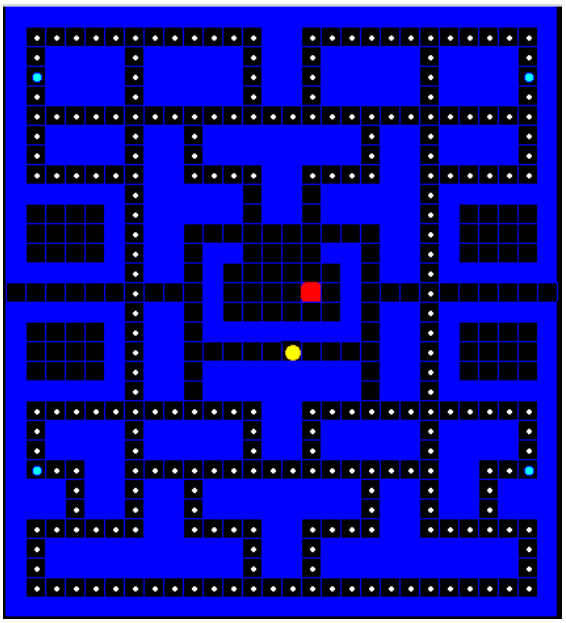

Figura 2: Representación del panel de juego.

Como vemos en la *figura 1*, los puntos blancos hacen referencia a los puntos pequeños y hay cuatro puntos azules, representando los puntos grandes.

4.2 Puntuación.

He añadido un método de puntuación, de tal forma que si el comecocos se come un punto blanco sumará 10 puntos y si come un punto azul sumará 50. Además, se ha creado en la ventana un apartado que muestra cual es la Puntuación actual del jugador y que va aumentando conforme comemos los puntos.

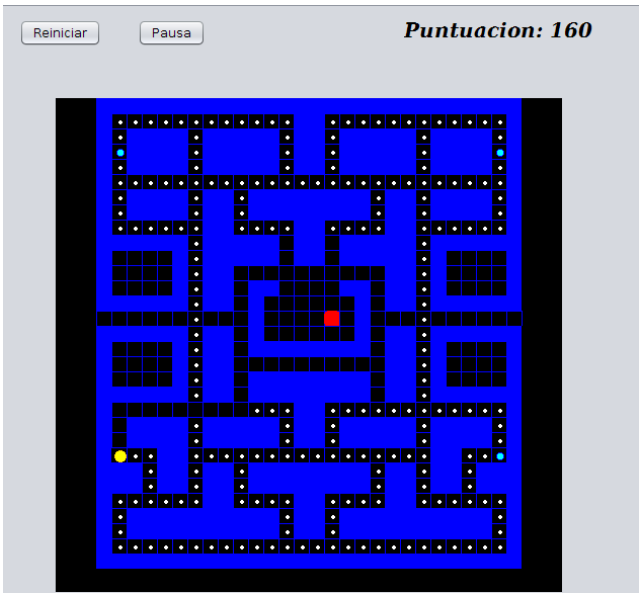

Figura 3: representación del panel con puntuación.

En la *figura 2* podemos ver como la puntuación es de 160, ya que el comecocos se ha comido 11 puntos blancos y uno azul.

4.3 Música.

Se ha incluido una clase para que, mientras el usuario juega, se escuche la canción típica del comecocos. Una vez se inicializa el juego comienza a escucharse la música.

4.4 Fantasmas.

Se han incluido también fantasmas al juego. No obstante, por falta de tiempo y problemas, se ha deshabilitado la opción de moverlos, por lo que aparece comentada en el paquete.

Los fantasmas tienen forma rectangular con bordes redondos y tienen ojos, como se muestra en la imagen:

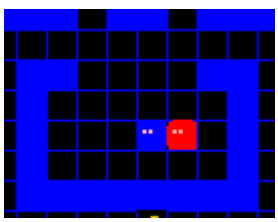

Figura 4: Fantasmas.

5. Conclusiones.

Como conclusión, podemos decir que esta práctica nos ha ayudado a ver como se puede crear un juego que, a priori, se puede ver como algo de gran complejidad pero, enfocándolo desde un punto de vista adecuado y analizando lo que se requiere en cada momento de forma aislada, podemos conseguir la realización del mismo de forma eficiente y asequible.

Esta práctica ha contribuido a afianzar el manejo de hebras y el desarrollo de las ventanas.