TPE Final
Programación Orientada a Objetos

Eugenio Damm - 57094
Martín Craviotto - 57756

-Año 2018-

Índice

1. Implementación

Para la implementación tanto del programa original como de los niveles agregados por nosotros optamos por mantener la estructura original para agilizar el agregado de nuevas clases y métodos. De esta forma, los primeros días de haber recibido el enunciado y el código de la cátedra nos dedicamos a leerlo, entenderlo y, ejecutandolo múltiples veces, ir comprendiendo cómo se comunicaban frontend y backend.

Luego de tener un primer pantallazo del funcionamiento del programa, nos dedicamos a pensar, dentro de las variantes de niveles para elegir, cuáles parecían ser las más "sencillas". Ésta elección se basó principalmente en nuestras opiniones iniciales viendo cómo se desarrollaba el código al ser ejecutado y con las clases ya existentes, viendo cuáles nos daban la posibilidad de generar la menor cantidad de variaciones. En nuestro caso, optamos por el nivel de los *Gaps* o espacios vacíos en desuso, como también, el nivel con *fruits* (Cherry y Hazelnut).

2. Elección de Niveles

a) Gaps

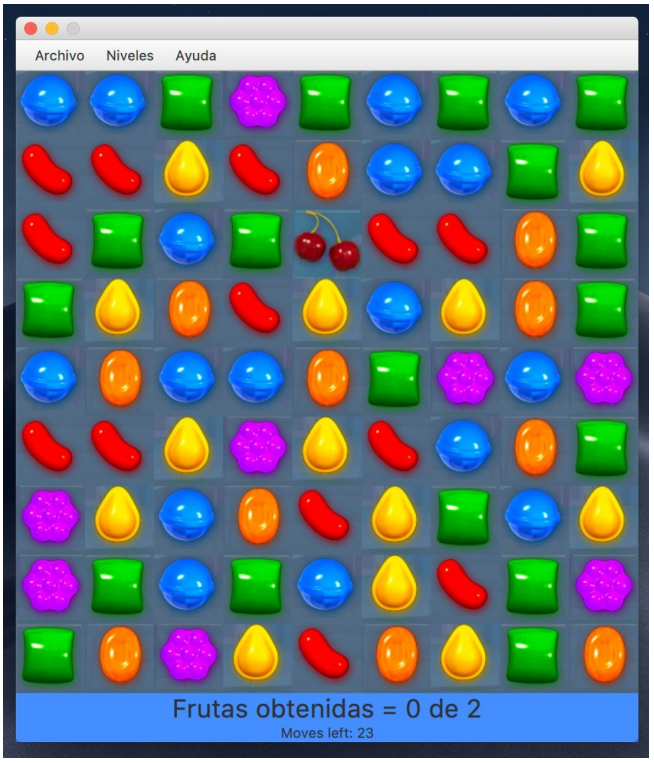
Decidimos elegir el nivel de los gaps debido a que desde el momento que vimos la captura de ejemplo, se nos ocurrió una primer manera de implementarlo (si bien no fue la que utilizamos). La implementación consiste en modificar la inicialización de la grilla de juego, en vez de tener una grilla de 9x9 Cells, tenemos dos rectángulos de 3x9 Cells y un rectángulo central de 3x9 pero de una *celda* nueva llamada Gap. Ésta nueva versión de la celda nos facilitó la generación visual de la grilla como también el reconocimiento de las mismas para distribuir los caramelos.

Este nivel nos pareció conveniente ya que desde un principio creíamos que iba a solicitar simplemente una reestructuración de la grilla, ya que luego los caramelos se ordenarían solos dependiendo de si se encuentran en una Cell o un Gap.

b) Fruits

A diferencia del nivel de gaps, decidir sobre si implementar el nivel con frutas no fué tan simple. Originalmente dudamos sobre optar por un nivel como jaula donde nos parecía que podría llegar a ser parecida a la de gaps. De todas formas, luego de releer el código pensando en cuál sería la mejor alternativa, nos dimos cuenta que agregando un *Element* nuevo en forma de fruta, y evitando su combinación con otros caramelos iba a resultar de gran facilidad y así lo fue.

Creamos tanto una celda nueva que nos genere de manera aleatoria caramelos y frutas para lanzar al tablero (limitándose a una fruta máximo en el tablero), como también un nuevo enumerador con nuestros dos tipos de fruta *Cherry* y *Hazelnut.*



Para el funcionamiento en sí, el nivel es muy similar al original del código fuente en cuanto a que genera las celdas y suelta caramelos/frutas, y luego mantiene activo un contador de cuántas frutas hay en juego. Si una fruta exitosamente alcanza la última fila, la misma desaparece del tablero y una nueva se genera. Éste nivel finaliza si el usuario logra que 2 frutas alcancen el final del tablero dentro del límite de movimientos.

3. Funcionalidades agregadas

Para nuestra propia implementación del juego agregamos en principio más funcionalidad visual en forma de un menú para el jugador, donde se le da la posibilidad de elegir sobre los 3 niveles disponibles con su respectiva información y descripción en la barra de menú en la parte superior.

Una vez seleccionado el nivel adecuado, se genera la grilla correcta y comienza la nueva partida. En la parte inferior del tablero ahora podemos apreciar tanto el puntaje como los movimientos disponibles y/o las frutas que han alcanzado el final del tablero (dependiendo qué nivel se seleccione).

Sumando al entorno gráfico agregamos las imágenes adecuadas para nuestras *gaps*, *cherries* y *hazelnuts*.

Adentrándonos en el backend, como se mencionó previamente agregamos un "tipo" de celda nueva llamada ***Gap*** que está internamente definida por default con un nuevo *Element* llamado ***GapElement*** que le da su funcionamiento. Éste implica la inamovilidad de la celda como su incapacidad de ser intercambiada con otro caramelo o fruta. Por otro lado, creamos un nuevo *Enum* llamado ***FruitType*** que nos da ambas frutas, y el cual utilizamos para la generación aleatoria de las mismas. Para ésto, tenemos un nuevo generador rotulado ***CandyFruitGeneratorCell*** que como su nombre indica nos generará tanto caramelos como frutas que luego el tablero ubicará en las celdas.

Como se puede apreciar, todas nuestras implementaciones de clases se adecúan al código de la cátedra para, como se mencionó previamente, agilizar la programación del juego, y mantener el uso del paradigma OO.

4. Dificultades encontradas

Desde un principio tuvimos problemas con la configuración del IDE para poder ejecutar el código, teniendo que recurrir a consultar via mail para poder solucionarlo.

Una vez solucionado eso, debido a la cantidad de archivos y movimientos que hay en el juego en si, tardamos un tiempo en lograr comprender el funcionamiento íntegramente y saber donde realizar modificaciones y donde agregar nuevas clases. De todas formas, gracias al correcto diagrama de las clases existentes y el uso del paradigma, logramos hacer un seguimiento y entender el código con mayor facilidad.

En cuanto a dificultades nuestras a la hora de programar, nuestro mayor inconveniente fue en el relleno de las celdas para arrancar el juego debido a que las llamadas recursivas realizadas en el método *fallUpperContent*, por ejemplo, ya que al principio no las comprendíamos. Entonces, nos ocurría que teníamos las nuevas clases diagramadas y con sus métodos definidos, pero cuando queríamos crear un nuevo nivel teníamos excepciones de tipo *NullPointerException* o bien se nos generaban grillas incorrectas. Un ejemplo claro de esto surgió con el nivel de los gaps ya que nuestra idea original era crear solamente el elemento *Gap* que fuera inmutable dentro del tablero y en el método *fillCells* "saltearamos" las filas con gaps, de modo que cuando cayera el contenido de la fila superior los gaps funcionaran como un "puente". Luego de múltiples intentos fallidos y debuggeo interno del código, nos dimos cuenta de que iba a ser más conveniente crear celdas de tipo Gap que fueran incluídas al tablero mediante el método *initialize*.

Por otro lado, para el nivel de frutas, nuestra principal complicación fue relacionada con la aleatoriedad de frutas y caramelos. Posteriormente nos encontramos con otras trabas como que no nos genere una fruta en la última celda desde un principio, que también lo solucionamos mediante condiciones nuevas en la creación del tablero sobre cuáles son las posiciones "mínimas" en las que puede existir una fruta.

5. Conclusión

En resumen, logramos implementar dos de los 4 niveles, agregando un GUI para interactuar con el jugador/usuario en la pantalla principal que le diera un estilo único a nuestro programa. Para lograrlo tuvimos que dedicarle gran parte del tiempo a la lectura y comprensión del código de la cátedra para posteriormente facilitar la implementación. Si bien tuvimos complicaciones en el desarrollo del mismo, se puede decir que en términos generales logramos llevarlo adelante sin grandes problemas ni problemas de tiempo significativos.