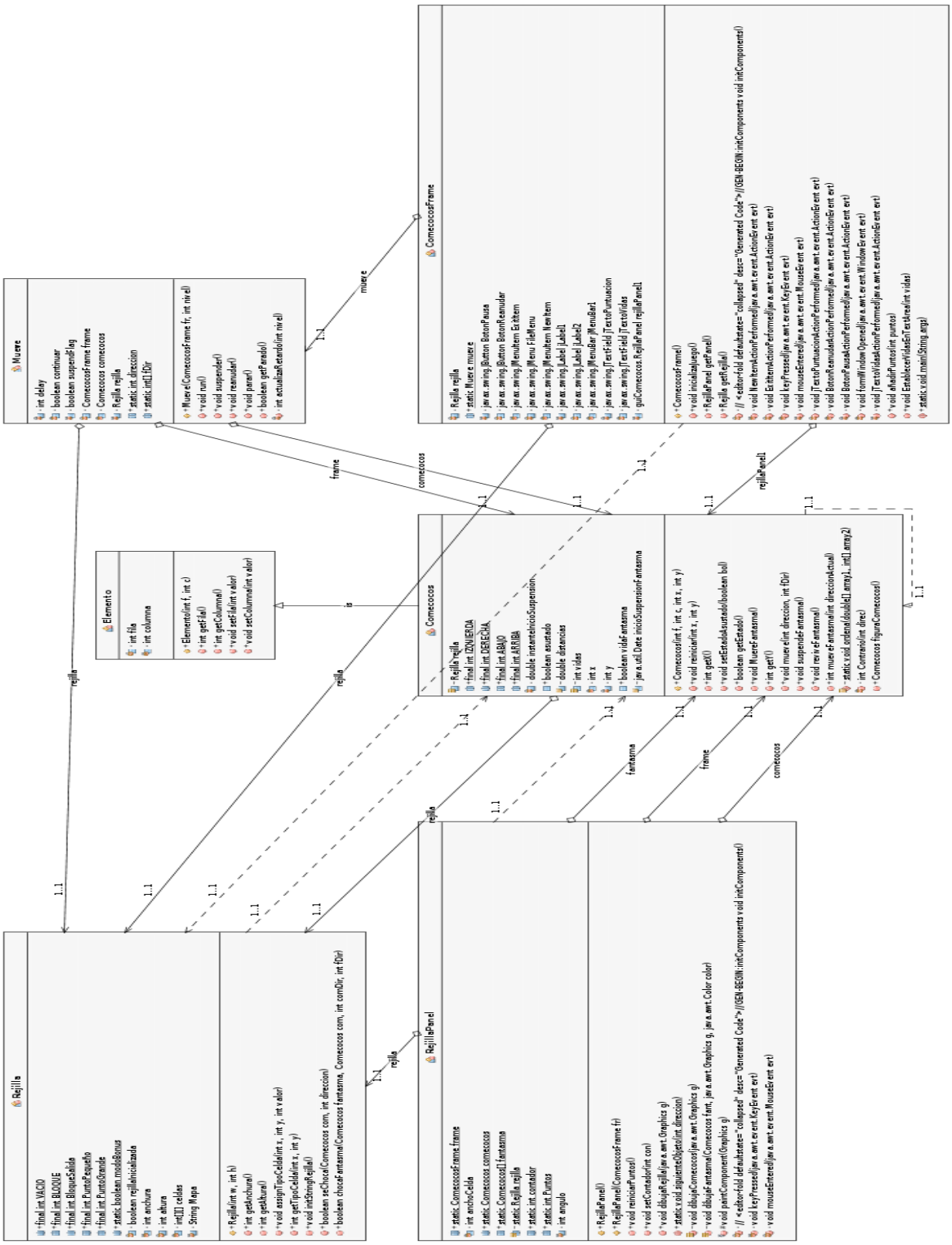
Práctica 8
Informe Juego Comecocos

Manuel Moya Ferrer, José Manuel García Giménez.
manuelmoya@correo.ugr.es , jgarciagimenez@correo.ugr.es
3º GITT Especialidad en Telemática Universidad de Granada.

UML


Rejilla
RejillaPanel
Elemento
Comecocos
Muere
ComecocosFrame
rejilla
frame
comecocos
fantasma
rejillaPanel1
is
1..1
1..4

Justificación de la solución

Para la resolución de este problema separamos las clases en dos grandes bloques. El de la interfaz gráfica, que se encuentra en el paquete guiComemcocos. Y el segundo bloque, que es en el que se encuentran todas las clases necesarias para el funcionamiento del juego.

Vamos a ver primero las clases para el funcionamiento del juego, empezando por el paquete Data:

Paquete Data:

La clase **rejilla**. Esta clase sirve para formar el mapa del juego. En esta clase tenemos como atributos a los distintos parámetros necesarios para la formación del mapa como la altura o la anchura. En lo que respecta a los métodos de esta clase, tenemos algunos para acceder a los distintas celdas del mapa así como para modificarlas. Luego tenemos tres métodos importantes para el funcionamiento del juego:

- initStringRejilla(), este método sirve para asignar los valores enteros a cada celda a partir del String que se tiene como base para la construcción del mapa.

- SeChoca(), sirve para ver si alguno de los personajes de nuestro juego se va a chocar con algún elemento del mapa. Para ello le pasamos como parámetro un de los personajes y su dirección. Devuelve un booleano en función del resulatdo.

- SeChocaFantasma(), este método sirve para comprobar si el comecocos se va a chocar con alguno de los fantasmas, para ello le pasamos el objeto comemecocos , un vector con los objetos fantasmas y las direcciones de todos estos elementos.

La clase **Mueve**. Esta clase identifica la tarea que se implementará en una hebra para que los diferentes personajes del juego se muevan constantemente. Implementa la interfaz Runnable. Los atributos de esta clase identificar el estado de la hebra así como para identificar los distintos elementos que se encarga de mover la misma.
La parte importante de esta clase es el método run(), que se ejecutará al lanzar la hebra. En este método simplemente tenemos un bucle infinito en el cual se va actualizando la posición de los distintos elementos y se espera un tiempo hasta efectuar el siguiente cambio.
Se implementan también otros métodos que sirven para modificar el estado de la hebra: renudar(), parar(), suspender() y otros para modificar sus datos miembro como setRetardo().

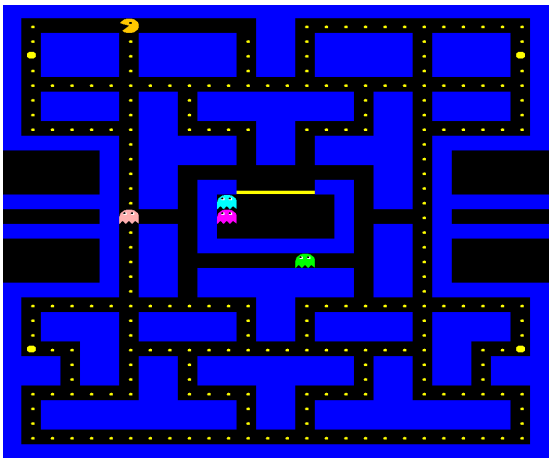
La clase **Elemento.** Esta clase sirve como base para crear la clase comecocos. En ella se identifica un elemento del mapa y se localiza en el mismo. Los datos miembro son fila y columna donde se encuentra el elemento. Los métodos sirven simplemente para acceder a la escritura y lectura de los datos miembro.
La clase **Comecocos.** Esta clase sirve para crear tanto los objetos comecocos como los objetos fantasma. Extiende de la clase elemento que hemos visto anteriormente y se le añaden algunos otros datos miembro que sirven para especificar algunos atributos específicos como las vidas del comecocos. Dentro de los métodos de esta clase podemos destacar los siguientes:

- mueve(); sirve para mover el comecocos. Se le pasan como parámetros la dirección a la se va a mover y las direcciones de los fantasmas. El método comprueba primero que el movimiento solicitado es válido, es decir , si no se choca con ningún elemento estático del mapa. A continuación comprueba si se va a chocar con algún fantasma. Si esto ocurre se reinician los fantasmas y el comecocos y se le resta una vida al jugador mientras tenga vidas suficientes. Si se choca con un fantasma y era su última vida el juego acaba.

- MueveFantasma(); sirve para mover a los fantasmas. Podemos dividir el movimiento de los fantasmas en dos partes, el movimiento en estado normal o el movimiento cuando están asustados(El comemcocos ha conseguido un punto grande del mapa). El movimiento normal es un movimiento semialeatorio que elige aleatoriamente entre los movimientos más óptimos para llegar al comecocos de esta forma se consigue que el movimiento sea más o menos impredecible pero no se distancie demasiado del objetivo. Sin embargo cuando los fantasmas están asustados , estos siempre eligen la mejor ruta para huir del comecocos.

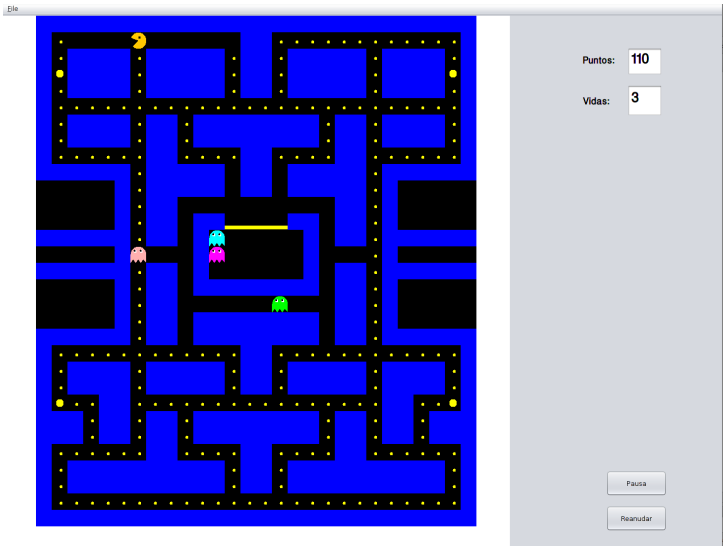
Paquete GuiComecocos:

En **clase RejillaPanel** definimos una ventana en la cual tendrá transcurso el juego:



En esta clase sobreescribiremos métodos como paintComponent, un capturador de teclas para el movimiento, así como sobreescribiremos el método mouseEntered para capturar el foco del ratón en dicha ventana.
Principalmente, en esta clase, para los diferentes componentes, usaremos las clases dibujaFantasma, dibujaComecocos y dibujaRejilla, los cuales se llamarán desde paintComponent para realizar repaint posteriormente.

La clase **ComecocosFrame** es la ventana principal del juego:



En ésta tenemos un menú File, que nos permite iniciar una nueva partida, así como salir, y ademas tenemos indicadores sobre el numero de vidas que tenemos y la puntuación. También tenemos 2 botones para parar y reanudad el juego. Dentro de este jFrame principal, incluiremos la ventana RejillaPanel, por lo que el juego quedará incluido dentro de ComecocosFrame. Para incluir esta, creamos un “custom code” en un elemento añadido a ComecocosFrame, e insertamos “new RejillaPanel(this);” creándose una ventana RejillaPanel1.

Mejoras implementadas

En nuestro proyecto implementamos las siguientes mejoras:

1. Incluimos los puntos grandes y pequeños en el tablero como hemos comentado en el apartado anterior de forma que se van sumando puntos cuando se comen y se acaba el juego cuando se acaban los puntos. En la Clase RejillaPanel se lleva el recuento de los puntos y se va haciendo que aumente el marcador cuando el comecocos se los come.

2. Dibujamos el laberinto con otros colores respetando el diseño original del juego, para ello simplemente modificamos los colores de los elementos en la clase RejillaPanel.

3. Incluimos un botón de pausa y uno de reanudar, estos botones simplemente llaman a los métodos correspondientes de la clase mueve para que se pare o reanude el juego.

4. Modificamos la visualización del comecocos de forma que cambie según la dirección en la que se mueva. También añadimos el efecto de abrir y cerrar la boca. Para ello modificamos el ángulo de apertura en cada iteración de la hebra modificando el método dibujaComecocos de la clase RejillaPanel.

5. Modificamos la visualización de los fantasmas para que tuvieran forma de fantasma así como para que cambien de color en función del estado con el método dibujaFantasma() de RejillaPanel.

6. Añadimos los fantasmas, implementamos el movimiento de los mismo y la gestión de los puntos y las interacciones entre los fantasmas y el comecocos como hemos visto en el apartado anterior añadiendo una serie de métodos a las clases comecocos, mueve y rejilla.

7. Añadimos la mejora de tener 3 vidas, de forma que cuando mueras se reste una vida y se reinicie el juego pero no los puntos que ya han sido comidos, es decir, se reinicia la posición de los fantasmas y el comecocos.