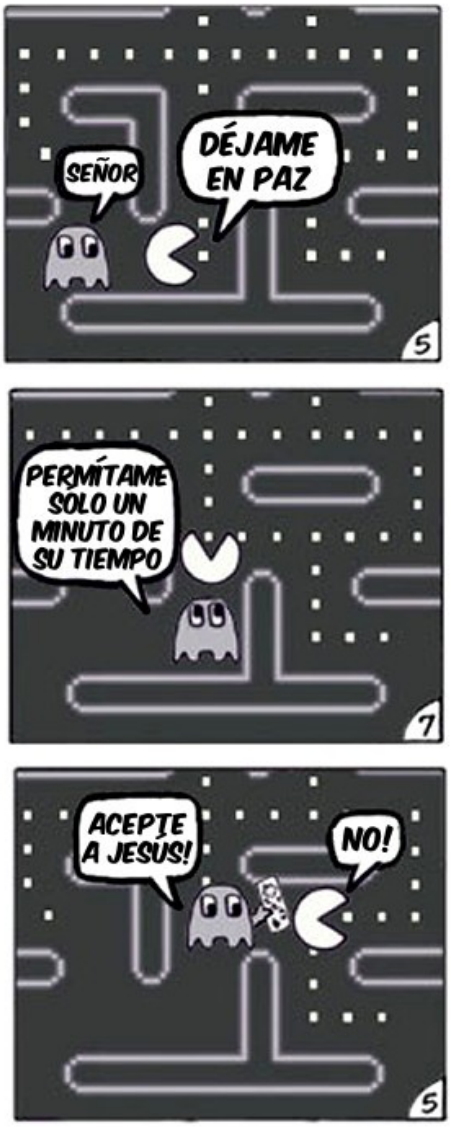
~ Practica 3 NTP ~
Desarrollo de un Comecocos con Java



Por: Juan Miguel Lechuga Pérez
Francisco Jesús Delgado Almirón
2010 - 2011

Pacoman

En ésta práctica se ha realizado un Comecocos con Netbeans abarcando todas las opciones tanto obligatorias como optativas indicadas en el guión de la práctica.

Manual de juego

Para empezar una nueva partida, abra el programa y pulse el menú Archivo->Nueva.
Seleccione un mapa, un nivel inicial de juego y cuantos jugadores van a jugar.
Las teclas de control son:

- Jugador 1: Flechas
- Jugador 2: WASD
- Pausa: Barra espaciadora

El juego consiste en comer todos los puntos que haya en el mapa evitando los fantasmas para así ir subiendo niveles de dificultad. Tocar un fantasma implicará perder una vida. Cuando se pierden todas las vidas, acaba el juego.

Compilación y ejecución del programa

Para compilar el programa, se puede hacer fácilmente desde el programa Netbeans abriendo el proyecto y ejecutando la acción "Build project"

Para ejecutar el programa, ejecutar el .jar de dentro de la carpeta dist del proyecto con:
$> java -jar Comecocos.jar

Explicación de las clases de la aplicación

Para la realización de ésta práctica, hemos optado por dividir las clases en 2 paquetes: guicomecocos y motor_comecocos, de manera que la interfaz y el dibujado de la pantalla está separado del sistema de juego. Así, es posible tener distintos tipos de interfaz para un mismo juego de comecocos.

- **Paquete guicomecocos**: Consiste en toda la programación relacionada con la interfaz: Swing, dibujado del mapa, carga de imágenes, sonidos y récords. Contiene las siguientes clases:
    - *AyudaDialog:* Diálogo simple que contiene la ayuda relacionada con el juego
    - *CargadorSprite:* Carga las imágenes del juego desde sus respectivos archivos en una representación de un HashMap como se explicará más adelante
    - *ComecocosFrame:* Clase principal de la interfaz. Es la que se ejecuta para acceder al juego. En ella se controlan los eventos mas importantes de la interfaz y su conexión con las clases del paquete motor_comecocos.
    - *ConfiguraciónGUI:* Clase simple para organizar en un solo lugar toda la configuración referente a la interfaz: Imágenes, sonidos, tema elegido, teclas y colores.
    - *ConfiguraciónTeclasDialog:* Diálogo para configurar las teclas del juego.
    - *ConfiguracionTemaDialog:* Díálogo para configurar el aspecto del mapa en el juego.
    - *GestorRecords:* Clase serializada encargada de controlar todo el tema de los récords.
    - *GestorSonido:* Clase encargada de cargar y reproducir el sonido asociado al juego.
    - *MostrarRecordDialog:* Diálogo para mostrar los récords.
    - *PanelMapa:* Clase encargada de dibujar el mapa del juego, los comecocos, la barra de estado y la imagen inicial cuando no se está jugando.
    - *Record:* Clase simple usada como estructura para guardar un récord.

- **Paquete motor_comecocos**: Se encarga de gestionar la partida, desde el movimiento de los comecocos y fantasmas hasta las puntuaciones asociadas a los jugadores.
    - *Bicho:* Clase abstracta para representar un comecocos o fantasma en el juego y sus datos en común, como el estado, la velocidad y su posición.
    - *Comecocos:* Clase que hereda de Bicho. Añade un control de las direcciones del comecocos como se explicará más adelante.

    - *DatosMapa:* Clase simple que contiene la información relativa a un mapa de juego: sus dimensiones, nombre, los datos de cada coordenada y las posiciones iniciales de los comecocos y fantasmas.
    - *EstadoJuego:* Clase serializada que mantiene toda la información relativa y volátil referente al juego en un sólo lugar. De ésta manera, se pueden guardar y cargar partidas simplemente cargando una instancia de esta clase.
    - *Fantasma:* Clase heredada de Bicho que contiene información de la ruta que tiene que tomar hasta el comecocos, la probabilidad de ir a por él, y los datos de tiempo asociados a su estado comestible y salida de la caja de los fantasmas.
    - *GestorMovimiento:* Clase usada para lanzarse como hebra y gestionar las iteraciones del juego actualizando en cada momento la información de EstadoJuego.
    - *Mapa:* Clase para gestionar un mapa de juego y la búsqueda de rutas en él mediante el algoritmo A*.
    - *MotorComecocos:* Clase principal del paquete. Controla las demás clases dentro del paquete y sirve como interfaz para otros paquetes ajenos a motor_comecocos.

Gestión de las partidas
Cada partida del comecocos se puede controlar mediante los diferentes menús del juego. Así se pueden lanzar nuevas partidas, cargar/guardar y pausarlas.

- **Cargar/Guardar partidas:** Mediante la serialización de la clase EstadoJuego, se pueden cargar y guardar partidas fácilmente simplemente controlando las instancias de la clase.
- **Pausar:** El juego permite pausar de 2 maneras:
    - *Pausa automática:* El juego se pausará/reanudará cada vez que se pierda el foco del PanelMapa.
    - *Pausa manual:* Mediante la pulsación de la barra espaciadora (tecla por defecto para la pausa) o de la acción del menú editar, se puede pausar manualmente el juego hasta que se reanude manualmente otra vez. Estando pausado manualmente, los eventos de perder/ganar el foco en PanelMapa no despausan ni reanudan el juego.

Dibujado del mapa
El mapa se dibuja en cada iteración gracias a la clase PanelMapa, que se encarga de coger la información de la clase Mapa y la representa haciendo uso de un objeto Graphics2D.

- **Barra de estado:** Se dibuja en la parte de arriba del mapa una barra de estado conteniendo información referente a los jugadores (vidas y puntos de cada uno) y también del tiempo restante para completar el nivel, el tiempo que se lleva jugando y el nivel actual.
- **Casillas:** Cada casilla se dibuja acorde al carácter que lo representa en el sistema. Los puntos se comprueban antes si han sido ya comidos por el comecocos y en ese caso se descarta su dibujado.
- **Comecocos y Fantasmas:** Se dibujarán dependiendo de la dirección hacia la que miran.

Gestión de los puntos
Los puntos que se puede comer el comecocos se guardan en una variable ArrayList<Point> de manera que cada vez que se coma un punto, se agrega su coordenada al ArrayList y a la hora de dibujar se comprueba si está en el ArrayList o no. De ésta manera, se evita tener que mantener 2

mapas (uno para el mapa original en caso de iniciar partida nueva y otro para ir borrando los puntos de él). Por cada punto pequeño se suman 10 puntos y por cada punto grande 50, y se ponen los fantasmas en estado "huyendo". Una vez que todos los puntos se hayan comido del mapa, se pasa directamente al siguiente nivel.

Movimiento de los comecocos

Los comecocos se mueven de la siguiente manera:

1. Avanzan continuamente en una dirección hasta que se queden bloqueados
2. Si se pulsa una dirección alternativa, se guarda en una variable
3. Cada vez que avance una casilla, se comprueba la siguiente dirección que se le ha ordenado y si puede cambiar, cambia de dirección.
4. La siguiente dirección se guarda durante 3 casillas, de manera que se puede indicar antes de llegar a una esquina la siguiente dirección sin tener que ser justo en la ultima casilla. Así, es muchísimo mas fácil controlar el comecocos.

Cada paso de una casilla a otra se divide en un número de pasos indicados por las variables posOffset y maxPosOffset. La primera indica el desplazamiento que lleva desde esa casilla y la segunda el máximo desplazamiento que puede tener antes de pasar a la siguiente casilla.

Control de las teclas

Las teclas para jugar se gestionan por eventos keyPressed en la clase PanelMapa y se van comparando con las teclas configuradas en ConfiguracionGUI para realizar una accion u otra.

Gestión de los fantasmas

En el juego aparecerán los 4 fantasmas que habían en el comecocos original. Estos perseguirán al comecocos y huirán de él cuando se coma un punto gordo.

- **Explicación de la inteligencia artificial:** Cada fantasma funciona de la siguiente manera:
    1. Aparece al comienzo de la partida esperando en la caja central del mapa durante un tiempo determinado (Cada fantasma tardará mas o menos para salir de ella)
    2. Cuando pasa ese tiempo, el fantasma cambia de estado ENCERRADO a SALIENDO. Durante este estado, el fantasma se dirige desde la posición inicial hasta su posicion_salida que es fuera de la caja.
    3. Una vez llega a la posición salida, comienza a perseguir al comecocos al pasar al estado NORMAL.
    4. Cuando el comecocos come un punto gordo, el fantasma pasa al estado HUYENDO. Durante este estado, irá en cualquier dirección que no sea la mas cercana para llegar al comecocos.
    5. Si el comecocos come al fantasma, éste último pasa al estado VOLVIENDO, donde sólo se le dibujarán los ojos en escena e irá hasta la casilla posicion_salida.
    6. Una vez que llega a la casilla posición salida, cambia su estado a ENTRANDO y se dirige a posicion_inicial, dentro de la caja.
    7. Al llegar a posicion_inicial, vuelve al estado ENCERRADO y comienza de nuevo el proceso.
- **A*:** Los fantasmas siempre tienen constancia de cómo llegar al comecocos por el camino mas corto. Esto se ha conseguido mediante una versión ligeramente modificada del algoritmo A* muy usado en los juegos de hoy en día para encontrar una ruta entre 2 puntos situados en una matriz 2D (para 3D el algoritmo se complica un poco mas) evitando obstáculos. El algoritmo se asemeja bastante a la mecánica de ramificación y poda (Branch & Bound) y se usa en la clase Mapa para buscar una ruta de direcciones a tomar entre 2 casillas del mapa. Así, en cada "esquina" del mapa (casilla donde haya más de 1 alternativa a la dirección actual del comecocos o fantasma) se calcula una ruta de direcciones a tomar en cada esquina hasta el comecocos.

- **Probabilidades:** Para diferenciar unos fantasmas de otros, hemos decidido incluir una probabilidad diferente (25%, 50%, 75% y 100%) en cada fantasma de usar la ruta proporcionada por el algoritmo A*. Así, se produce una tirada de probabilidad y si es menor que la probabilidad, el fantasma irá hacia el comecocos. Si no, tomará una dirección aleatoria (evitando ir al contrario de la dirección actual)
- **Huyendo:** A la hora de huir del comecocos, los fantasmas toman una dirección aleatoria entre el conjunto de direcciones posibles eliminando la que le lleve al comecocos en menos movimientos. Tambien, en algunas ocasiones entrarán en estado de pánico y empezarán a temblar sin saber hacia donde moverse.
- **Puntos al comer:** Cada vez que se come a un fantasma, a la puntuación se le suman 200, 400, 800 o 1600 puntos dependiendo del número de fantasmas comidos con un solo punto gordo. Si se come otro punto gordo, el contador de fantasmas se reseteará para evitar puntuaciones abusivas.
- **Colisiones:** Para detectar cuando hay una colisión entre un comecocos y un fantasma, se usa un sistema de colisiones con cajas (Bounding boxes) usado en muchos juegos 2D. Consiste en asociar un rectángulo a cada comecocos y fantasma de manera que cuando un rectángulo interseca con otro existe colisión. Cabe destacar que las cajas son ligeramente más pequeñas que la imagen de los comecocos y fantasmas, así se realizan colisiones mas realistas donde se ve que una imagen toca a otra.

Gestión del juego
Cada iteración del juego está controlada por la hebra lanzada mediante la clase GestorMovimiento. En ella se actualizan los tiempos de ejecución y esperas de los objetos, se comprueban los estados de los comecocos y fantasmas y se indica cuando ha de redibujarse la imagen del tablero.

Niveles
Cada vez que se completan todos los puntos de un mapa, el juego avanza de nivel. Por cada nivel que se avanza, los fantasmas serán mas veloces y saldrán más rápido de la caja central.

Condiciones de perder una vida
Los comecocos perderán una vida si entran en contacto con un fantasma si no está huyendo o si se acaba el tiempo del mapa. Si entra un comecocos en contacto, pierde vida él sólo. Pero si es por fin de tiempo todos los comecocos perderán una vida. Cada comecocos tiene 3 vidas al comenzar una partida. En cuanto pierde las tres, ya no puede jugar. Si no queda ningún comecocos, se acaba la partida.

Menús
El juego dispone de distintos menús para organizar y realizar acciones útiles para el juego:
- **Archivo:** Contiene las acciones de nueva partida y salir del programa.
- **Edición:** Dispone de las acciones que guardan/altera el curso del juego: Cargar, Guardar partida y la pausa manual.
- **Configuración:** Dispone de las distintas opciones para personalizar el juego. Desde el cambio de la temática de las imágenes hasta la configuración de las teclas pasando por activar/desactivar el sonido.
- **Récords:** Incluye un lanzador del diálogo de mostrar los récords.
- **Ayuda:** Permite ver un diálogo de ayuda y el diálogo “sobre...”.

Gestión de los récords
El juego dispone de un sistema de gestión de récords de manera que en cuanto acaba la partida se comprueba si se ha batido un récord y se añade a la lista. La lista de los récords se guarda y carga automáticamente desde un archivo “records.dat”. En caso de no existir tal archivo, se crean unos récords fáciles de batir.

Los récords se guardan bajo la clase Record guardando puntuación, nombre, nivel y tiempo que duró la partida, aunque luego solo se ordenan de mayor a menor puntuación. Para cada mapa que haya en el juego, habrá una lista de 10 récords.
Sólo se podrá añadir un récord cuando se juegue en modo un jugador, y no se cargue desde una partida. Así se evitan posibles "trucos" para alterar la tabla de récords ya que jugando 2 jugadores sería muy fácil sacar mas puntuación que la normal y cargando una partida se podrían poner 20 récords de la misma partida.

Añadidos propios
Aparte de los objetivos obligatorios y optativos, hemos decidido añadir una serie de extras bastantes interesantes al juego:

- **Multijugador:** A la hora de comenzar una nueva partida, se puede elegir si jugar con un amigo en el mismo ordenador. Cada jugador llevará sus vidas y puntuaciones propias de cada uno y los fantasmas irán a por ambos comecocos, aunque preferirán el que menos giros necesite para llegar a él. La partida acabará cuando ambos comecocos hayan perdido todas sus vidas y entonces se mostrará quien es el ganador (aquel que tenga la mayor puntuación).
- **Sprites:** Los comecocos, fantasmas y frutas en vez de ser dibujados mediante funciones geométricas del objeto Graphics2D, se ha optado por que sean imágenes cargadas desde un sólo archivo por cada tema. Así, el juego cargará todas las imágenes como subimágenes de ese solo archivo y aplicando un filtro para eliminar el color de fondo (en este caso RGB(255, 0, 255)) se obtienen las imágenes deseadas. En el juego se pueden elegir varios temas que son simplemente los archivos de imágenes de donde sacar los sprites.
- **Sonidos:** Se ha añadido reproducción de sonido en el juego reproduciendo los archivos .wav que hay incluidos en el .jar gracias al objeto AudioSystem de Java. Al igual que los Sprites, cada tema puede tener sus propios sonidos, aunque para no inflar demasiado el tamaño del programa se ha optado porque la mayoría sean compartidos. Hay 3 sonidos que se reproducen: Al iniciar una partida, al moverse el comecocos y al perder una vida.

  Existe un problema en GNU/Linux con el sonido, que parece ser debido al driver PulseAudio, ya que ni con el openjdk ni con el oficial de java se evita una excepción que ocurre al leer datos del stream del archivo de sonido, con lo cual solo se reproducen un par de sonidos hasta que se produce la excepción y se cierra el sonido para el resto de la ejecución para evitar bloqueo del juego. En cambio, en versiones de Windows no ocurre tal problema.
- **Frutas:** Cada cierto tiempo constante, se genera una fruta en una posición del mapa. Para evitar que aparezca en posiciones imposibles de acceder, se genera en una posición donde había antes un punto (ya sea gordo o grande). Comer esta fruta añadirá 500 puntos al marcador del comecocos que se la coma. Despues de otro tiempo constante, la fruta desaparece del mapa y se vuelve a iniciar el contador para que aparezca en otra posición.