Práctica 2: Pacman 2.0

Curso 2017-2018. Tecnología de la Programación de Videojuegos. UCM

Fecha de entrega: 19 de diciembre de 2017

El objetivo fundamental de esta práctica es introducir el uso de la herencia y el polimorfismo en la programación de videojuegos mediante SDL/C++. Para ello, partiremos del Pacman 1.0 de la práctica anterior y desarrollaremos una serie de extensiones que se explican a continuación. En cuanto a funcionalidad, el juego presenta las siguientes tres modificaciones/extensiones:

1. El objetivo del juego es ahora llegar a una determinada puntuación. Para ello, se debe establecer qué acciones suman puntuación y cuánta (comer comida, comer vitaminas, comer un fantasma, ...). Cuando se acaba con toda la comida del mapa se pasa al siguiente nivel (otro mapa diferente que se carga del fichero correspondiente). Debes mostrar de alguna manera la puntuación actual del juego en todo momento (se darán detalles en clase).

2. Ahora hay dos clases de fantasmas: los fantasmas simples (igual que los de la práctica 1), de colores rojo, amarillo, rosa o azúl, y los fantasmas *inteligentes* o *evolucionados*, de color morado. Éstos últimos nacerán, crecerán, se reproducirán (si se dan las condiciones) y morirán. Tendrán por tanto un atributo edad que irá aumentándose en cada paso del juego, con el consiguiente aumento de tamaño. A partir de cierto punto se considerará que son *adultos*, momento a partir del cual, el tamaño dejará de crecer, se moverán de manera agresiva tratando de acercarse al pacman, y se reproducirán si se cruzan con otro fantasma evolucionado adulto, y hay un espacio libre al lado de alguno de los padres, dando lugar a un nuevo fantasma evolucionado que aparecerá pegado a sus padres. Al llegar a cierta edad los fantasmas evolucionados morirán y su cuerpo quedará inerte en el sitio hasta ser comido por el pacman.

3. Las partidas pueden guardarse y cargarse: Al iniciar el juego aparecerá un menú con dos opciones, jugar y cargar partida. La opción debe seleccionarse con el ratón. En caso de seleccionarse la opción de cargar partida, se introducirá a continuación el código numérico de la partida y se cargará la partida a partir del fichero correspondiente (en caso de encontrarse). De manera análoga, mientras se está jugando, si se pulsa la tecla *s* seguida de un código numérico y seguida de *intro*, la partida se guardará con el código numérico introducido. Se darán detalles en clase sobre esta funcionalidad.

Detalles de implementación

Formato de ficheros

Los ficheros de mapas y configuraciones iniciales cambian respecto a los de la práctica 1 para adecuarse a las nuevas funcionalidades. Hay que tener en cuenta que el formato será común al de los ficheros de partidas guardadas, a excepción de un par de aspectos que se indican a continuación. En concreto:

- En el caso de partidas guardadas el fichero comienza con dos números enteros separados con espacio, indicando el nivel de juego y la puntuación actual.

- Después (o como primer dato en el caso de ficheros de nivel) aparecerá el mapa, que se representará como los mapas de los ficheros de la práctica 1, excepto que no se incluirán entre sus casillas las posiciones iniciales de los fantasmas y pacman. Los posibles valores numéricos serán por tanto los siguientes: 0 = vacío, 1 = muro, 2 = comida y 3 = vitamina.
- A continuación vendrá un número entero indicando cuántos fantasmas hay, apareciendo después una línea por cada fantasma con números enteros separados por espacio incluyendo los siguientes datos: tipo de fantasma (0 = fantasma simple y 1 = fantasma evolucionado), posición actual, posición inicial, dirección de movimiento, y en el caso de fantasmas evolucionados, la edad.
- Finalmente, aparecerá una línea con números enteros separados por espacios con los datos del pacman. En concreto: posición actual, posición inicial, dirección de movimiento, y en el caso de ser un fichero de una partida guardada, energía y vidas restantes.

Cambio de nivel

Como se ha indicado, cuando el pacman come toda la comida y vitaminas de un mapa, el juego pasa al siguiente nivel. El juego incluye por tanto dos nuevos atributos, la puntuación y el nivel. Este último se utiliza para formar el nombre del fichero del siguiente nivel, el cual tiene la forma `levelX.pac` siendo X el nivel correspondiente. Para ello puedes hacer uso de un flujo de tipo `stringstream`, que permite manejar *strings* como flujos de texto y por tanto aprovechar las funcionalidades para insertar y extraer datos en flujos de texto. Al pasar de nivel los fantasmas se destruyen y se crean de nuevo a partir de los datos del fichero. El pacman sin embargo se mantiene y simplemente se recoloca en la posición inicial indicada en el nuevo nivel.

Diseño de clases

A continuación se indican las nuevas clases y métodos que debes implementar obligatoriamente y las principales modificaciones respecto a las clases de la práctica 1. Deberás implementar además los métodos (y posiblemente las clases) adicionales que consideres necesario para mejorar la claridad y reusabilidad del código. Cada clase se corresponderá con la definición de un módulo C++ con sus correspondientes ficheros .h y .cpp.

Clase GameObject y jerarquía de objetos del juego: La clase raíz de la jerarquía es la clase abstracta `GameObject`, que declara la funcionalidad común a todo objeto del juego mediante los métodos abstractos `render`, `update`, `loadFromFile` y `saveToFile`. Declara también un atributo con el puntero al juego. De ella heredan las clases `GameCharacter`, clase abstracta base de las clases `Pacman` y `Ghost`, y la clase `GameMap`.

Clase GameCharacter: Define todo el estado y comportamiento común de pacman y fantasmas. En concreto debe definir atributos para las posiciones actual e inicial, dirección, y puntero a la textura y su celda a usar para el renderizado. Debe implementar además los métodos `render`, `loadFromFile` y `saveToFile`.

Clase Pacman: Hereda de la clase `GameCharacter`, añade los atributos y métodos extra necesarios, implementa el método `update` (más los métodos auxiliares necesarios) y re-define los métodos `loadFromFile` y `saveToFile` para incorporar la lectura/escritura de los atributos añadidos según corresponda.

Clases Ghost y SmartGhost: La clase `Ghost` hereda de `GameCharacter` e implementa el método `update` con el movimiento aleatorio básico de los fantasmas simples. La clase `SmartGhost` hereda de `Ghost`, añade un atributo para la edad y redefine los métodos `update`, `loadFromFile` y `saveToFile` correspondientemente (haciendo uso de los métodos correspondientes de la clase base).

Clase GameMap: Hereda de la clase `GameObject` y añade por tanto respecto a la versión de la práctica 1 la implementación de los métodos `loadFromFile`, `saveToFile` y `update`.

Clase Game: Añade el soporte necesario para llevar a cabo las nuevas funcionalidades indicadas. En concreto, ahora las personajes del juego (pacman y fantasmas) deben guardarse en una lista polimórfica (tipo `list<GameCharacter*>`) y por tanto los recorridos y búsquedas sobre ella deben hacer uso de iteradores.

Pautas generales

A continuación se indican algunas pautas generales que vuestro código debe seguir:

- Se debe hacer un buen uso de la herencia y del polimorfismo de manera que el código sea claro y se eviten las repeticiones de código.
- Asegúrate de que el programa no deje basura (usa para ello el mecanismo de detección de basura explicado en el CV).
- Todos los atributos deben ser privados o protegidos excepto quizás algunas constantes del juego en caso de que se definan como atributos estáticos.
- Define las constantes que sean necesarias (no deberían aparecer literales numéricos arbitrarios en el código).
- No debe haber métodos que superen las 30-40 líneas de código.
- Escribe comentarios en el código, al menos uno por cada método que explique de forma clara qué hace el método. Sé cuidadoso también con los nombres que eliges para variables, parámetros, atributos y métodos. Es importante que denoten realmente lo que son o hacen. Preferiblemente usa nombres en inglés.

Funcionalidades opcionales

Se darán detalles en clase y/o en el Campus Virtual.

Entrega

En la tarea del campus virtual *Entrega de la práctica 2* y dentro de la fecha límite (19 de diciembre), cada uno de los miembros del grupo, debe subir un fichero comprimido (.zip) que contenga los archivos de código .h y .cpp del proyecto, las imágenes que se utilicen, y, un archivo `info.txt` con los nombres de los componentes del grupo y unas líneas explicando las funcionalidades opcionales incluidas y/o las cosas que no estén funcionando correctamente. Además, para que la práctica se considere entregada, deberá pasarse una *entrevista* en la que el profesor comprobará, con los dos autores de la práctica, su funcionamiento en ejecución, y si es correcto realizará preguntas sobre la implementación. Las entrevistas se realizarán en las dos sesiones de laboratorio siguientes a la fecha de entrega, o si fuese necesario en horario de tutorías.