TP 3: NetHack

noviembre 9

2015

Integrantes: Santiago Bozzo (98772), Tomás Pinto (98757)

Análisis del problema:

Debemos leer el archivo .map y cargar cada entidad con sus respectivas funciones, las clases que debemos implementar son:

1. Enemigo (Goblin y Orco) que el héroe puede eliminar intentando ocupar su celda y al morir se informa en pantalla, y puede arrojar una moneda con 50% de probabilidad.
2. Moneda, que aparecerán en el mapa cada vez que se elimine a un enemigo con un 50% de probabilidad. Si el héroe intenta ocupar su celda, esta desaparece y la cantidad de monedas del héroe aumenta, mostrando en pantalla la cantidad de monedas que tiene en total.
3. Pared, que actúa de manera tal que si el héroe quiere ocupar su celda, no suceda nada, el héroe y la pared deben quedar en su lugar.
4. Salida, simplemente finaliza el juego una vez que el héroe intenta ocupar su celda.

Resolución:

Cargado del mapa:

El mapa se carga a partir de una lista que contiene cada línea del archivo .map que hay que cargar. Primero se crea un mapa en blanco con el alto igual a la cantidad de líneas que hay y se referencia el ancho a la cantidad de caracteres que contiene la primer línea.

Una vez creado el mapa en blanco se referencian las variables ‘columna_fila’ e ‘elemento_filas’ a cero y se recorre carácter por carácter cada línea. El valor de ‘columna_fila’ crece cada vez que se pasa al siguiente carácter y vuelve al valor cero una vez que se produce un salto de línea, incrementando el valor de ‘elemento_filas(que representa cada linea)’, ‘columna_fila’ e ‘elemento_filas’ representan la posición en el archivo .map, que representaran lo mismo en la instancia de Mapa.

Cuando el carácter actual representa a un Actor se crea una instancia del mismo y se lo agrega en su posición ‘columna_fila’ e ‘elemento_filas’ en el mapa. Cuando no representa ningún actor simplemente se pasa al siguiente carácter dejando ese espacio en blanco (‘.’).

Al finalizar el cargado del mapa se verifica si se agregó un Héroe (si no, no tendría sentido el juego) en el mapa levantando una excepción si no se encuentra u otra excepción si se añadieron héroes de más.

Implementación de clases:

La clase Enemigo:

Su comportamiento no difiere del comportamiento de Actor, por lo tanto ‘hereda’ del mismo, sus funciones. Los enemigos en esta implementación del juego no tienen movilidad ni pueden atacar, lo único que puede suceder con ellos es que si el héroe intenta moverse a su posición, estos mueran y en su lugar aparezca una moneda con cierto grado de probabilidad.

Goblin y Orco:

Son los enemigos del juego, están representados en el juego como “g” (Goblin) y “o” (Orco), en lo único que difieren es en como son representadas en el mapa, así que ambas entidades son representadas por la clase Enemigo mencionada anteriormente.

La clase Moneda:

Su comportamiento no difiere del comportamiento de Actor, por lo tanto ‘hereda’ del mismo, sus funciones. Se delega la instancia del juego, para que una vez que el héroe intente ocupar su celda, la moneda se sume al inventario de monedas total del héroe.

La clase Salida:

- Esta clase hereda de actor, y solo se redefinirán los métodos dibujar (para que devuelva “<”) e interactuar_con_heroe.
- En cuanto a cómo funciona una vez que el héroe intenta interactuar con ella, una vez más se delega la instancia del juego, se finaliza el mismo, y se muestran carteles informativos por pantalla.

La clase Pared:

- Esta clase hereda de actor, y solo redefinimos el método “dibujar”, para representarla con el carácter “#”.
- No tiene ninguna funcionalidad especial en el juego, solamente bloquear el paso al héroe.

Bozzo, Santiago (98772)
Pinto, Tomás (98757)