Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

Paradigmas de programación 2024
Curso: K2004
Turno:Jueves Mañana

Paradigma
Objetos

Tema

Wollok Game

Juego
Dungeons & Objects

GRUPO FORKS	
Nombre y Apellido	**Legajo**
Matias Gabriel Hoz	211.500-1
Valentín Fernández Pizzella	176.503-6
Brian Sisco Salazar	206.414-5
Ignacio Rodríguez M.	214.074-3
Matias Goldberg	203.682-4

UTN.BA
INGENIERIA
EN SISTEMAS

Presentación del juego

Eres un valiente caballero adentrándose en una mazmorra. Deberás sortear los obstáculos que se te presenten en búsqueda de la llave que te permita llegar al tesoro. Cuidado, la mazmorra guarda temibles criaturas que harán que tu camino no sea para nada fácil Pero no todo es malo, si la suerte te acompaña, puedes encontrar equipamientos de antiguos aventureros que te fortalezcan para llegar preparado a combatir al dragón Muchos peligros, un objetivo, estás preparado?

Desarrollo del código

El programa se encuentra seccionado en ocho archivos, los cuales son:

- Main: Recibe los métodos presentes en "configuraciones" y se encarga de inicializar el juego
- Configuraciones: Presenta todos los parámetros de inicialización del juego, tales como la configuración de la *pantalla*, las *teclas*, los *movimientos* automáticos de los enemigos y las *colisiones*.
- Pisos: En este archivo se encuentra el objeto mapa, el cual define los *límites* para que el caballero no se salga, las *paredes*, las *zonas prohibidas* (que se componen de las paredes y los límites del mapa) y las *puertas*, que facilitan el cambio de habitación. Además, se inicializan todas las visuales del juego.
- Personajes: Se encuentra el objeto principal del juego, el cual es el *Caballero*, el personaje que maneja el usuario y el objeto *barraDeVida,* el cual es esencial para el desarrollo del juego.
- Enemigos: Define como clase a los dos tipos de enemigos que existen, *Arquero* y *Esqueleto*. Al final se encuentran las declaraciones de cada enemigo individualmente
- Equipamientos: Presenta todos los objetos con visuales que no pueden ser considerados como personaje ni como enemigo. Estos se clasifican con la clase *Objeto*, que abarca el *arco*, la *llave* y la *poción* y con el objeto *flecha* aparte.
- Direcciones: Los objetos encontrados en este archivo definen importantes métodos para el funcionamiento del juego, incluyendo el cambio de visuales y el desplazamiento de los demás objetos.
- Tests: Contiene los tests correspondientes a todos los aspectos del juego.

Lo primero que ocurre cuando presionamos "Run Game" es que el programa envía todos los mensajes a **configuracionesIniciales** y a **mapa** para inicializar todo lo necesario para que el juego funcione correctamente.

configuracionesIniciales llama a game para inicializar la pantalla, a las teclas que se utilizarán para jugar, a todos los ticks de los enemigos y también inicializa las colisiones que pueden ocurrir durante el juego.

Entre los ticks encontramos disparar, el cual utiliza el método disparar(personaje) de la flecha para que cada arquero dispare cada 2 segundos. El otro tick es movimiento, el cual, utilizando el método movimiento(x, y) hace que cada esqueleto siga su patrón de movimiento también cada 2 segundos.

Las flechas son venenosas, y matan automáticamente a quien toca, por eso, su colisión con cualquier objeto le envía el mensaje muerto(flecha) al personaje que toque.

Los esqueletos atacan cuerpo a cuerpo, su colisión con el caballero los lleva a atacarlo, luego verifica que ambos estén vivos para reproducir el sonido del impacto y devolver al caballero un poco hacia atrás

La colisión del caballero con una puerta envía el mensaje pasaElCaballero() a la puerta con la que está colisionando, lo que funciona para que el caballero se teletransporte del otro lado y cambie la visualización de la pantalla.

La colisión del caballero con los botones hace que aparezca la visual indicando que hay un elemento para levantar, esta visual desaparece cuando se aleja o cuando agarra el elemento.

mapa cuando lo llama program se encarga de inicializar todas las visuales correspondientes al comienzo del juego, que son el caballero, todos los esqueletos, todos los arqueros, todos los elementos (arco, poción, llave), las paredes (el objeto que cubre las habitaciones en las que no estamos) y la barra de vida.

Una de las claves del funcionamiento del juego son los objetos **direcciones**, cada vez que un objeto quiere moverse, envía un mensaje a la dirección en la que desea moverse, la cual se encarga de devolverle el cambio de imágen y si está "autorizado" a moverse. La dirección chequea que el objeto no esté tratando de moverse a una *zonaProhibida* (inicializadas en el objeto **mapa**) y si verifica que puede moverse, le envía el movimiento correspondiente al objeto. En caso de que esté tratando de pasar por una zona prohibida, envía tocaBorde(), método importante para la flecha, que se detiene y se remueve su visual, los demás objetos cuentan con el método pero se encuentra vacío.

El objeto más complejo de todo el código es el **caballero** ya que es el que más interacción tiene con el resto de objetos. Como ya explicamos anteriormente, cuenta con numerosas colisiones y además interactúa con las direcciones. Además de esto, cuenta con el método cambiaVida(), que le envía a la barra de vida todo lo necesario para indicar visualmente cuanta vida le queda al caballero. El método agarrar(equipo) es el que permite al caballero hacerse con los diferentes elementos que puede encontrarse en su recorrida por la mazmorra. Cuando el caballero lo agarra, el elemento desaparece y se añade al equipamiento del caballero. Luego, según que tiene el caballero en su inventario, se "desbloquean" diferentes interacciones, como por ejemplo disparar una flecha cuando ya tiene el arco, o poder acceder al salón del tesoro cuando consiguió la llave. En caso de que agarre la poción, automáticamente se añade 3 puntos de vida.

Por último, el caballero cuenta con un método perder(), que luego de mostrar en pantalla que el caballero ha caído en combate, detiene el juego.

El **esqueleto** tiene sus métodos para movimiento, el método para atacar al caballero en la colisión, el método para cambiar su propia vida cuando recibe daño, un método estáEnRango(objeto) que es una versión "mejorada" de la colisión (por motivos de dimensiones) y su propio método muerto(flecha), el cual muestra visualmente que murió impactado por una flecha (única forma que tiene de morir) y luego el cadáver se desintegra.

El **arquero** cuenta únicamente con el método estáEnRango(objeto) y muerto(flecha)

Marco teórico

Durante su desarrollo, tanto el juego como el código pasaron por diferentes versiones. A medida que el juego avanzó, el código se hizo más complejo y tuvimos que aplicar buena parte de la teoría vista durante el año, como el concepto de polimorfismo, o el manejo de clases, característico del paradigma de objetos.

En un principio el código estaba planteado sin el uso de clases, pero a medida que se fueron construyendo más objetos, se volvió completamente necesario. Las clases definidas son:

- Esqueleto: Define como es el enemigo que combate cuerpo a cuerpo, tiene un método para atacar, el cual en primera instancia verifica que siga vivo, un método que define el movimiento y otro que lo envía, un método que cambia la vida, y otros dos para cuando muere, según sea por daño cuerpo a cuerpo o a distancia por una flecha. El método "*estáEnRango*" facilita la colisión con los demás objetos al agrandar su rango, y por último el método vacío "tocaBorde()". Cada esqueleto se inicializa únicamente con su posición, el resto se encuentra definido en la clase.
- Arquero: Esta clase es más sencilla, ya que además de las definiciones de los atributos, cuenta únicamente con dos métodos, uno para cuando muere y otro para chequear que tiene el arco en su equipamiento. Al igual que con los esqueletos, cada arquero se inicializa únicamente con la posición
- Elemento: La función de esta clase es definir todos los objetos que puede encontrar el caballero y que puede agarrar. Para mayor simpleza del código, esta clase cuenta con un único método "*estáEnRango*" y como atributos la posición y la imágen, las cuales se inicializan en cada objeto. Cualquier otro método o interacción que puedan tener estos objetos, se encuentran definidos en otros objetos.
- AgarrarConLaE: Esta es una clase que depende completamente de la clase **Elemento**, su única función es mostrar un objeto con imágen de tecla E cuando el jugador se encuentra en rango de agarrar un elemento. Se definió como clase ya que cada objeto se declara con la posición correspondiente a uno de los elementos.
- Flecha: Esta clase se tuvo que definir al construir la clase de Arquero, ya que como hay varios personajes que pueden lanzar flechas (el jugador, y varios enemigos), cada uno debe contar con una flecha diferente para que pueda haber más de una a la vez en la pantalla. La clase flecha, además de con los atributos, viene con el método *disparar,* que depende del personaje que esté disparando la flecha, y con el método *tocaBorde,* que la hace desaparecer. Al inicializar la flecha, debemos declarar quien es el tirador, junto con su posición y su dirección, ambas dependientes del tirador.
- ZonaProhibida:Esta clase es la que define las zonas por las que ningún objeto puede pasar. Definidas con un X e Y mínimos y máximos, cada objeto declara un espacio de coordenadas las cuales son prohibidas para cualquier objeto. Con esta clase definimos los límites del mapa junto con las paredes de la mazmorra. Además, la clase incluye un método de consulta "*esZonaProhibida*" que devuelve un valor booleano. Este método es utilizado en las direcciones para "confirmar" el movimiento de cada objeto.
- Puerta: La función de esta clase es meramente estética, una de los últimos retoques realizados al juego, fue que el jugador en cada momento ve únicamente la sala en la que se encuentra y los

enemigos o elementos que haya allí. Las puertas funcionan con un “switch” que envían mensajes según la habitación a la que se esté llegando para modificar la visualización de la mazmorra.

En cuanto al polimorfismo, podemos indicar que tuvo que ser utilizado en varias ocasiones, ya que cada objeto envía varias veces el mismo mensaje pero a diferentes objetos, como por ejemplo las direcciones, que envía el método tocaBorde() a la flecha, al caballero y al esqueleto, pero con diferentes comportamientos, la flecha se detiene y desaparece y tanto en el caballero como en el esqueleto, se define como un método vacío.

El método muerto(flecha), enviado por la colisión entre la flecha y un objeto (personaje o enemigo), también presenta polimorfismo, ya que el caballero y los enemigos se comportan diferente al recibir este mensaje.

Diagrama de clases