|  |
| --- |
| Programación I - 2019 |
| Torre Mágica |
| Informe de implementación. |

|  |
| --- |
| Brian Almada – (legajo) – (mail)  Nicolas Lohuandus – 37986655/2016 – nlohuandus@gmail.com |

# Introducción

Se nos pide implementar un simulador de batalla en la torre mágica. Para eso se debe mostrar por pantalla una serie de vigas, un mago que lucha contra los monstruos y por supuesto los monstruos. El mago, aparecerá siempre centrado en la viga superior, tendrá que moverse horizontalmente por las vigas y al llegar al final de una viga, debe caer a la viga inferior hasta llegar a la parte inferior de la pantalla. Una vez llegado al final el mago reaparecerá por la parte superior de la pantalla cayendo en la primera viga.

Los monstruos, tienen un comportamiento predeterminado moviéndose siempre en la misma dirección, hasta chocar con un el borde de la pantalla donde cambiaran su dirección. De la misma manera que el mago, cada monstruo al llegar al extremo de una viga, caerá a la viga inferior y al llegar al final de la pantalla reaparecerán.

Los hechizos, cada vez que el mago cae de una viga a otra, se generara un hechizo que congelara al monstruos por unos segundos, si el monstruo es tocado por el mago mientras este congelado el monstruo rodara hasta el final de la pantalla y desaparecerá, si el monstruo no es tocado por el mago, volverá a moverse.

El objetivo del simulador es eliminar todos los monstruos del nivel.

Como extras se implementó, la posibilidad del saltar para el mago, además de moverse en línea recta puede ir a una viga superior y cuando la toca también se produce un hechizo. Además, se implementó distintos comportamientos para los monstruos, en concreto tres tipos de movimientos distintos. Se implementó también, un contador del tiempo transcurrido de juego en segundos, un contador de puntaje y nivel actual, un sistema donde el jugador tiene 3 vidas en total, lo que quiere decir que puede ser tocado por los monstruos 3 veces antes de morir. También, un sistema de niveles y dificultad, una vez eliminados todos los monstruos el jugador puede elegir pasar al siguiente nivel presionando la tecla enter, en los siguientes niveles los monstruos se volverán cada vez más veloces a medida avancen los niveles. Por ultimo una implementación de un botón de pausa que detiene el juego.

# Clases implementadas

* Clase Viga: Su constructor contiene las mismas variables necesarias para dibujar un rectángulo, y agregadas dos variables bordeIz y bordeDer para identificar los límites de la figura dibujada.
  + - Contiene todos sus getters y setters para poder utilizar sus variables y la función Dibujar(entorno) utiliza entorno.dibujarRectangulo para mostrar en pantalla.
* Clase Mago: Su constructor contiene variables necesarias para dibujar un rectángulo y además, 3 boolean estado, mover, saltar que controlan su estado durante el juego y necesarios para sus funciones.
  + - Como variables globales de la clase además contienen un Arraylist de Disparo que almacena los disparos producidos durante el juego. También se instancia un objeto Disparo para poder trabajar con el dentro de la clase otras variables adicionales útiles para controlar el estado del mago durante el juego.
    - El método disparar(): Es quien crea los disparos y devuelve objetos del tipo Disparo.
    - El método avanzar(): controla el movimiento hacia la derecha y actualiza el estado de estar mirando a la derecha mediante un boolean.
    - El método retroceder():Controla el movimiento hacia la izquierda y actualiza el estado del boolean derecha a false.
    - El método caer(): controla el movimiento en la pantalla hacia abajo, ósea, las caídas del mago desde el borde de las vigas. Cuando este termina de caer se genera un disparo y se lo almacena en el ArrayList lDisparo.
    - El método saltar(): controla el movimiento hacia arriba en la pantalla, genera saltos.
    - El método contacto(): controla si los personajes tocan al mago o viceversa y actualiza sus estados de acuerdo a eso. En caso de que el mago ser tocado por un enemigo descuenta una vida (Corazón) en pantalla. Y en caso de ser un enemigo lo congela. Si el enemigo ya se encontraba congelado hace que ruede en pantalla, y en caso de estar congelado y no volver a ser tocado el enemigo se descongela y sigue su curso. Este es el método más importante de esta clase ya que controla todas las interacciones entre objetos.
    - Además contiene los getters y setter propios de la clase.
* Clase Disparo: Dentro de su constructor se almacenan las variables necesarias para dibujar de la misma forma que un rectángulo. Además xdir e ydir útiles para dar la trayectoria del disparo y un boolean que será quien indicara si el disparo va hacia izquierda o derecha.
  + - Método dibujar():dibuja el rectángulo en pantalla y le da su trayectoria para moverse.
    - Método trayectoria():Dependiendo del estado del boolean le otorga movimiento al disparo para izquierda o derecha.
    - Getters y setters propios de la clase.

# Problemas y decisiones durante la implementación

El principal problema para este trabajo práctico, fue la coordinación de los códigos implementados por ambos integrantes. La solución que encontramos aunque con algo de dificultad, fue el uso de GitHub integrado a eclipse, de esa forma todo lo implementado se sincroniza con el repositorio[[1]](#footnote-1) git en internet y ambos integrantes podemos tener el mismo código.

Para la implementación de los hechizos, se decidió usar ArrayList, ya que permite almacenar más de un hechizo producido por el mago, y cuando ya no se utilicen eliminarlo con facilidad, el ahorro en líneas de código con respecto a utilizar un arreglo fue muy grande.

El hecho de integrar saltos, fue una decisión tomada por la dificultad que traía que el mago solo dispare al tocar la viga, ya que si congelaba a un monstruo y tenía que evadir a otro debía recorrer toda la pantalla para aparecer arriba y recién ahí tocar al monstruo congelado, nos pareció más apropiado poder subir o bajar de las vigas con libertad.

Por cuestiones de diseño se decidió que, si bien el disparo tiene su propia clase, el ArrayList de disparos y la generación de disparos sean controlas dentro de la clase de mago. Así se podría deducir que quien dispara efectivamente él es mago (mago.disparar).

Para objetos relativamente no dinámicos, se decidió darle valores estáticos definidos dentro de su constructor, por ejemplo, el ancho y el color de las vigas, el ancho y alto del mago etc. Estos datos a lo largo del juego no cambian, por lo que no hay ningún método que los modifique.

# Códigos



Contenido

[Introducción 1](#_Toc9872796)

[Clases implementadas 2](#_Toc9872797)

[Problemas y decisiones durante la implementación 3](#_Toc9872798)

[Contenido 4](#_Toc9872799)

1. <https://github.com/Nlohuandus/TorreMagica> [↑](#footnote-ref-1)