Trabajo Práctico 2 — AlgoChess

[7507/9502] Algoritmos y Programación III
Curso X
Segundo cuatrimestre de 2019

Alumno:	Vilca, James Joseph
Número de padrón:	97240
Email:	$\text{joseph}_v v@hotmail.com$

Índice

1. Introducción

El presente informe reune la documentación de la solución del segundo trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar un juego de tablero basado en turnos utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

-El juego permite intentar mover una pieza hasta que lo logre. -El juego no permite intentar atacar hasta que lo logre. Si una unidad ataca a una unidad y no pudo atacarla, se cuenta como una accion realizada. -El juego empieza con el turno del jugador1. -La catapulta ataca a enemigos y/o aliados contiguos hasta en un rango de 2 casilleros desde la posicion inicial del objetivo inicial. -Cuando un soldado avanza como batallón, solo las piezas que no tengan ningún obstáculo, aliado o enemigo en la dirección que deberian moverse, pueden moverse. -El juego acaba cuando el enemigo se quedo sin piezas y presionar "terminar turno".

3. Modelo de dominio

Usando el paradigma de programación orientado a objetos realicé el trabajo práctico con el patrón State para la jugabilidadd general del juego junto con el modelo MVC para la interfaz gráfica.

En mi implementación, el juego consta en un sistema de turnos donde se pueden realizar 5 tipos de acciones de las cuales 4 tienen un botón asociado en la interfaz gráfica y una 1 se activa al dar click (seleccionar pieza del tablero).

4. Diagramas de clase

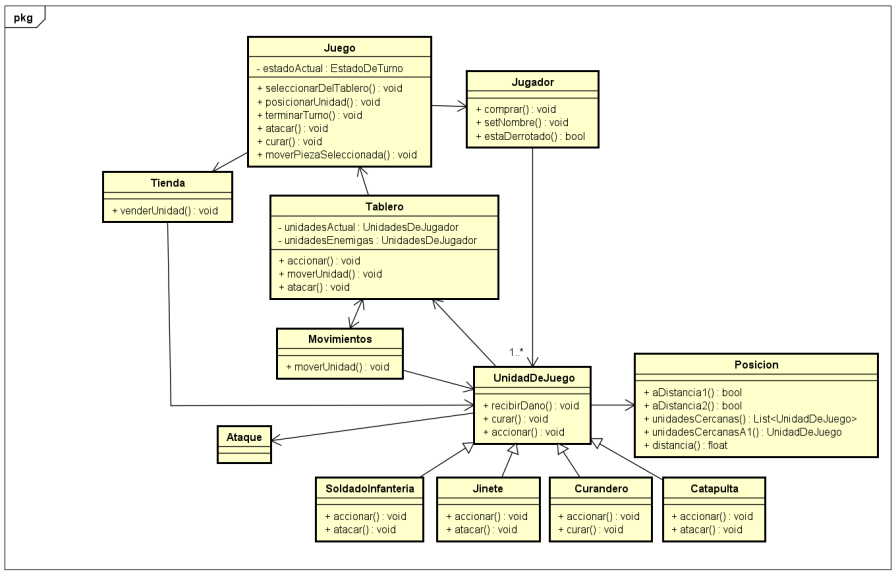

Figura 1: Diagrama del Juego.

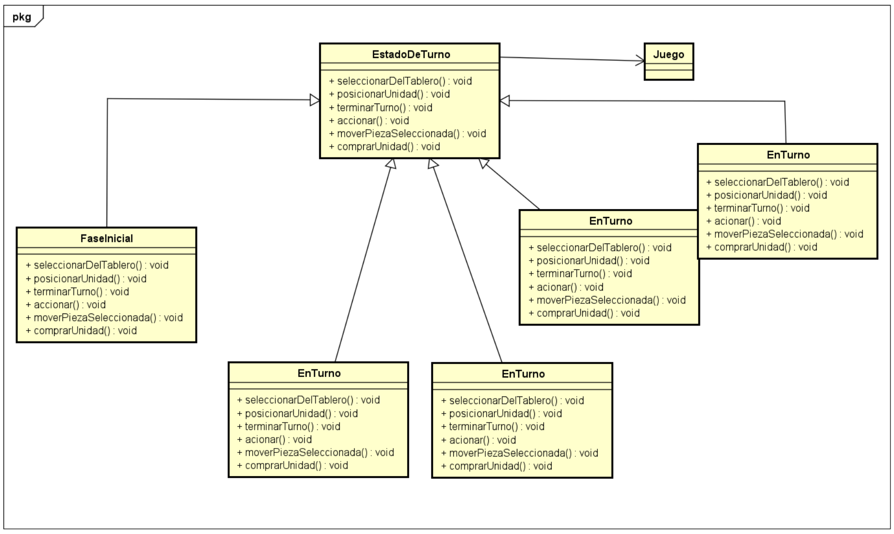

Figura 2: Diagrama del Juego.

5. Detalles de implementación

5.1. Modelo

Ataque cerano, mediano, lejano: Hice una especie de strategy donde la clase Ataque solo se encarga de verificar si el objetivo esta a la distancia requerida según sea el tipo de ataque. (Solo el jinete cambia de comportamiento de ataque según el contexto de su posicionamiento).

Unidad de Juego: Lo planteé como una clase abstracta donde todos tienen pueden accionar, atacar, curar y recibir daño. Para diferenciar el tipo de ataque o movimiento, la unidad al llamar su método accionar se envia asi mismo al tablero y el tablero recibe la clase en particular para decidir como atacar o mover.

```
public void atacar(UnidadDeJuego atacante, Posicion posVictima){

UnidadDeJuego victima = this.obtenerUnidad(posVictima);
if(victima == null || this.unidadAliada(victima)) return;

atacante.atacar(victima);
if(victima.estaDestruido()){
this.unidadesEnemigas.perderPieza(victima);
this.casillas.remove(posVictima);
}
}
```

5.2. Controlador y Vista

En la vista al terminar de realizar las operaciones, accionar y mover (en el modelo), se actualiza por completo el mapa (es un GridPane) y se vuelven a poner las imagenes a partir del modelo atra ves del Controlador. De esta manera la vista se abstrae de como funciona el modelo y el controlador conoce lo basico y necesario del modelo para poder actualizar la vista.

6. Excepciones

Trabajé sin excepciones, todo es controlador por el modelo.

7. Diagramas de secuencia

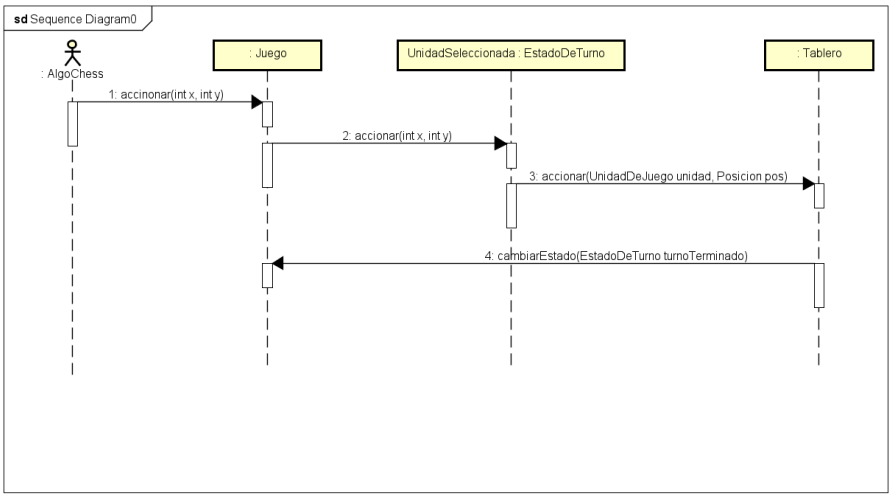

Figura 3: Jinete ataca a una unidad en la fase UnidadSeleccionada

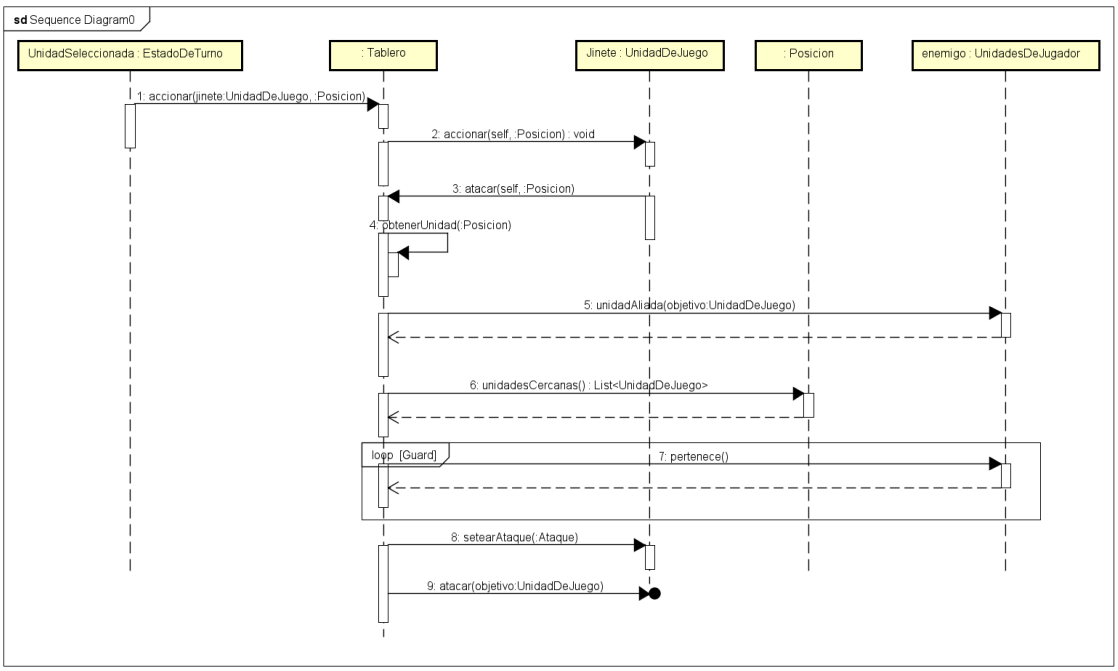

Figura 4: Parte donde actua el tablero cuando el Jinete ataca

Continuación de la situación anterior.