21 de Mayo de 2012
Alejandro Luebs – Mateo Harfuch

Ajechess

Abstract

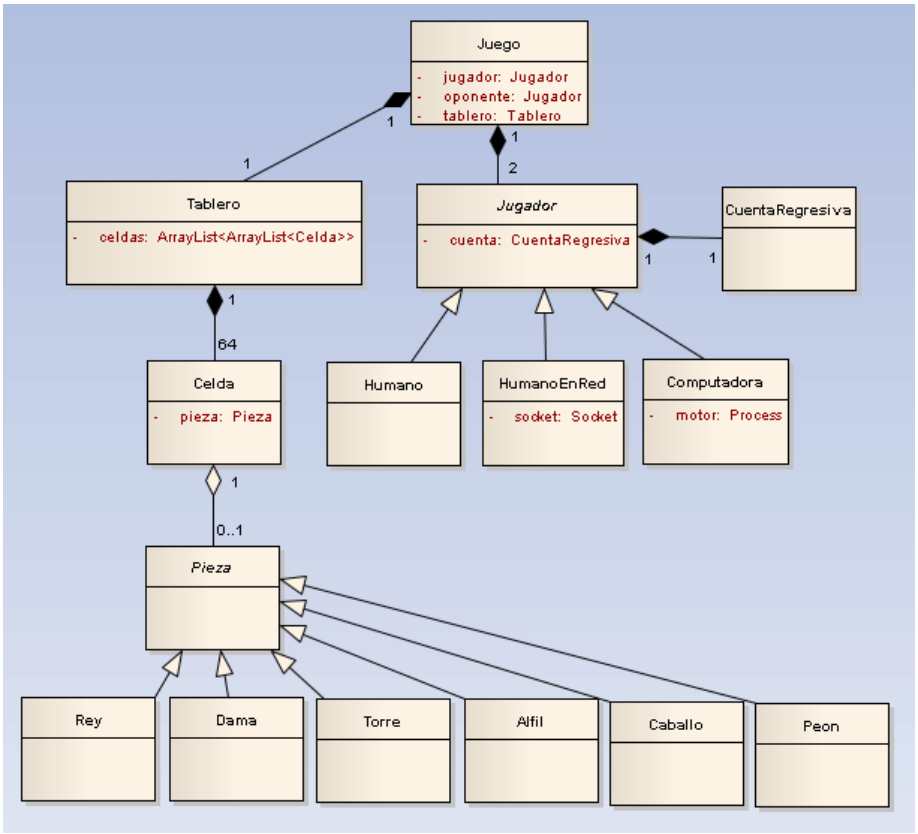
Se programó, utilizando Java, un juego de Ajedrez. Se comenzó por un motor central que representa las reglas y se lo probó en modo multijugador en una sola computadora utilizando una interfaz gráfica hecha en Swing. Luego, se implementó la posibilidad de jugar en red con otra persona o localmente contra un oponente virtual.

Desarrollo

El desarrollo de esta aplicación se encaró de a fases. Desde el comienzo del proyecto se tuvo en mente la posible expansión hacia otras plataformas u otras modalidades de juego no previstas a corto plazo. Esto hizo hincapié en la necesidad de flexibilidad y desacople entre clases. Sin embargo se decidió dedicar todo el esfuerzo en generar un entorno capaz de ser utilizado en distintas situaciones, para luego atacar las aplicaciones particulares.

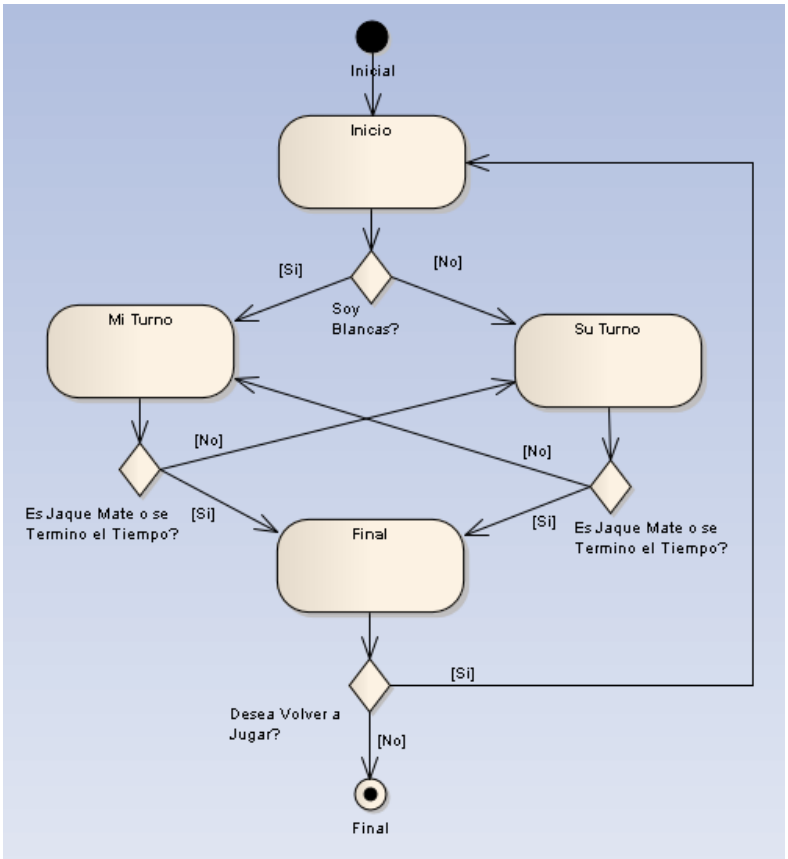
Fase 1: Motor

En la primer fase se comenzó a desarrollar el motor central con las reglas de juego y la lógica básica. Teniendo en cuenta la flexibilidad se implementaron distintas clases y las relaciones en ellas. Luego de tener las clases básicas se trabajó sobre la dinámica del juego, que se pensó como una maquina de estados finitos. Esta ultima es la encargada de organizar el juego propiamente dicho. En la siguiente figura se muestra un diagrama de clases de UML de la aplicación.



Como se puede ver en el diagrama anterior, el Juego cumple un rol central dentro del sistema y es el encargada de instanciar los demás objetos que forman parte de un partido. Para que pueda existir un juego, debe haber un tablero, que consiste de una cuadricula de 8 por 8 celdas. Dentro de cada celda puede o no haber una pieza. Hay diferentes tipos de piezas con sus movimientos y particularidades. Por otro lado, una partida cuenta con dos jugadores. Estos últimos pueden ser personas presentes en la misma computadora, jugadores a través de la red u oponentes virtuales. Finalmente, cabe destacar que cada jugador tiene una cuenta regresiva, que lleva el tiempo acumulado de sus turnos. Si a algún jugador se le termina el tiempo antes de que haya un jaque mate, pierde el partido.

La dinámica del juego es implementada a través de una maquina de estados, la cual es representada en la siguiente figura en un diagrama de estados de UML.



Como se puede apreciar del diagrama anterior, siempre se comienza en el estado de inicio, donde se inicializan todos los parámetros necesarios para comenzar el juego, como por ejemplo las piezas en el tablero. Luego, comienza el jugador que es blancas. Hasta que no haya un jaque mate o se termine el tiempo de algún jugador, hay un continuo salto entre los estados de mi turno y su turno. Finalmente, al terminar la partida se cae en el estado final, el cual da la posibilidad de jugar una nueva partida y volver a comenzar desde inicio.

Fase 2: Primer implementación

En paralelo con el motor se comenzó a desarrollar una interfaz gráfica utilizando Swing para poder realizar pruebas del motor. De esta forma, la primer aplicación particular del sistema es un programa de escritorio utilizando una ventana de Swing que permite partidos multijugadores en una misma computadora. Nuevamente, al plantear el sistema como un motor con distintas aplicaciones, se busco desacoplar por completo la ventana del resto de la aplicación. La interfaz se dividió en dos ventanas: una de configuración,

donde se ajustan los parámetros del partido, y la principal donde se desarrolla todo el juego. En la siguiente figura se muestran ambas ventanas.

Como se ve en la captura anterior en la ventana de configuración se puede ingresar la modalidad del juego, los nombres de los jugadores y el tiempo en minutos con el que cada uno comienza (o la dirección IP del oponente en caso de ser cliente en una partida por red). En la ventana principal a la izquierda se ve el tablero con las fichas donde se resalta la ultima jugada realizada. A la derecha están los nombres de los jugadores con sus respectivos tiempos restantes (el color rojo representa que es el turno de ese jugador), sus fichas comidas y los puntos que estas representan y las jugadas realizadas desde el comienzo del partido utilizando notación algebraica.

Fase 3: Modalidades de juego

Mientras se estaba añadiendo los últimos detalles funcionales a la primera implementación se comenzó a agregar nuevas modalidades de juego a la aplicación: jugar contra un compañero a través de la red (Internet) y contra un oponente virtual. Como el motor del programa fue pensado para poder agregarse estas modalidades los cambios que se tuvieron que introducir a este fueron mínimos. Prácticamente solo se tuvo que programar la clase del jugador correspondiente y su instanciación cuando corresponda.

La primera modalidad que se agregó fue la a través de la red, permitiendo a dos jugadores jugar desde distintas ubicaciones siempre y cuando tengan una conexión a Internet. Para ello prácticamente solo se tuvo que agregar la clase de HumadoEnRed, que encapsula toda la conexión, los sockets y los mensajes necesarios para poder jugar. El protocolo de comunicación fue diseñado específicamente para esta aplicación. Evidentemente, para poder jugar en red es necesario que uno de los jugadores sea servidor y el otro cliente. Esto lo decide el usuario desde la ventana de configuración. La única diferencia para los usuarios es que el servidor decide el tiempo que se va a jugar y el cliente debe conocer la dirección IP de este para poder conectarse a el. Internamente el orden de los mensajes del protocolo de comunicación cambia según uno sea cliente o servidor.

Luego se implementó la posibilidad de jugar contra un oponente virtual, dándole la posibilidad al usuario de jugar solo. Se investigó la posibilidad de implementar la inteligencia de este jugador, pero se llegó a la conclusión que representa una gran dificultad y que no es el fin de este proyecto. Por lo tanto se decidió utilizar un *chess engine* open source que implemente UCI (Universal Chess Interface) y comunicarse con el. Para la elección de un *chess engine* en particular se tuvo en cuenta que la aplicación debe ser multiplataforma, por lo que se buscó uno que este implementado en Java. Finalmente se trabajó con el motor de Carballo. Para comunicarse con el se estudió el protocolo de UCI y se implementó la clase Computadora que encapsula esta comunicación. Para ello se tuvo que ejecutar el *engine* desde la aplicación y enviar mensajes al standard input, recibiendo las respuestas en el standard output.

Trabajo a futuro

Partiendo de una aplicación de escritorio con varias modalidades de juego, se piensa migrarla a distintas plataformas. Al haber tenido en cuenta esta posibilidad desde el comienzo del desarrollo, la interfaz gráfica se encuentra completamente desacoplada del resto del juego y esto es lo único que se debe cambiar para poder realizar la migración. Es decir que se puede reutilizar prácticamente todo el código gracias a la cualidad multiplataforma que presenta Java. Ya se trabajó en la aplicación para Android y se investigó la posibilidad de realizar una aplicación web que corra en un servidor.