**Informe**

**Supuestos (aplicables a esta segunda entrega)**

1. El movimiento de los personajes será de a un casillero (el jugador elige paso a paso a que casillero moverse). La cantidad máxima de movimientos será la indicada por la velocidad de cada personaje.
2. El tablero será rectangular, de 60x20 casilleros.
3. La forma de aparición de los Algoformers en el tablero siempre será en su modo humanoide.
4. La distribución de las superficies en el tablero será aleatoria.

**Modelo de dominio**

Objetos vistos en el dominio del enunciado (hasta la segunda entrega): AlgoFormer, Forma, Tablero, Celda, Superficie, Efecto, Posición, Movimiento, Jugador y Juego.

AlgoFormer en realidad es una clase abstracta, de la cual heredan dos clases hijas concretas: AutoBot y Decepticon.

Cada AlgoFormer tiene dos estados posibles: estado humanoide y estado alterno (el estado alterno puede ser tanto del tipo terrestre como aéreo); es por eso que cada instancia de AlgoFormer está asociada a una instancia de FormaHumanoide, y una de FormaTerrestre o FormaAérea. Si bien el objeto AlgoFormer es uno solo, puede cambiar su comportamiento dinámicamente gracias a estos estados que guarda.

Jugador también es una clase abstracta, de la cual heredan dos clases hijas concretas: JugadorAutoBots y JugadorDecepticons.

Jugador representa a cada uno de los jugadores del juego. Contiene a todos los AlgoFormers de su equipo.

La clase Tablero representa el escenario en el cual se va a desarrollar el juego; es una cuadrícula rectangular de instancias de Celda.

Celda contiene a un AlgoFormer (si hubiese uno en la celda) y a la chispa suprema (si estuviese en la celda). Las celdas tienen asociada una instancia de Posición (que representa a las coordenadas de cada celda dentro del tablero).

Además, cada Celda tiene dos Superficies (una SuperficieAerea y otra SuperficieTerrestre), que a su vez contienen un Efecto (el efecto que contienen depende de qué tipo de superficie son).

Movimiento es un enum que representa a los movimientos que un AlgoFormer puede realizar, esto es las posibles direcciones en el tablero.

Juego es la clase encargada de utilizar a todas las clases descriptas recién (directa o indirectamente) para llevar cabo la ejecución del juego. Es decir, inicializa a todas las clases del juego y luego las utiliza para llevar adelante la ejecución.

**Diagramas de clases**

Diagrama general de clases del programa

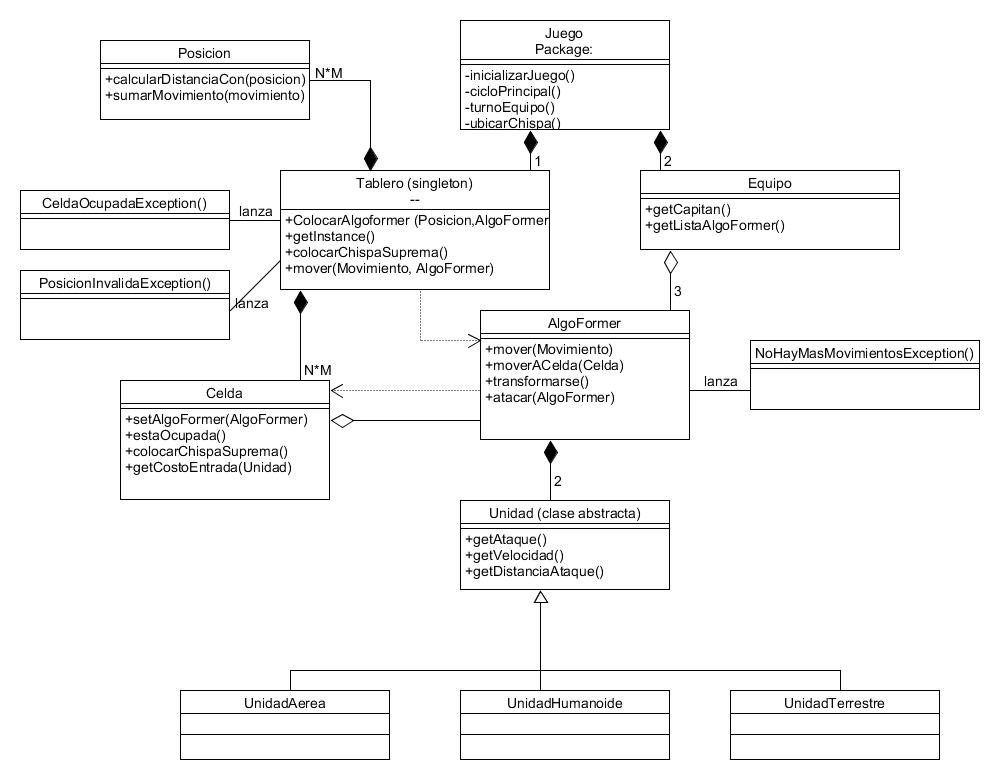


Diagrama de clases Tablero-Posicion-Celda-Movimiento

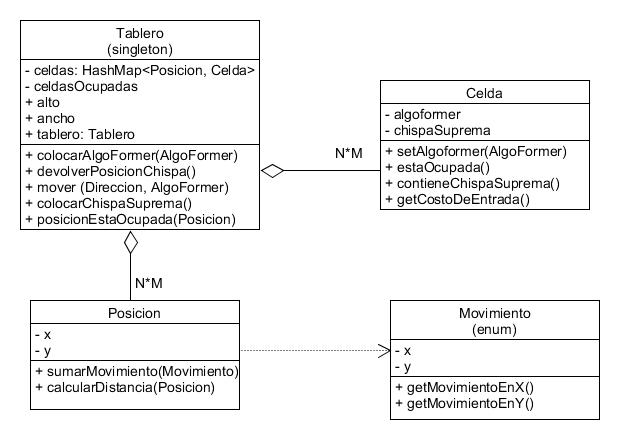
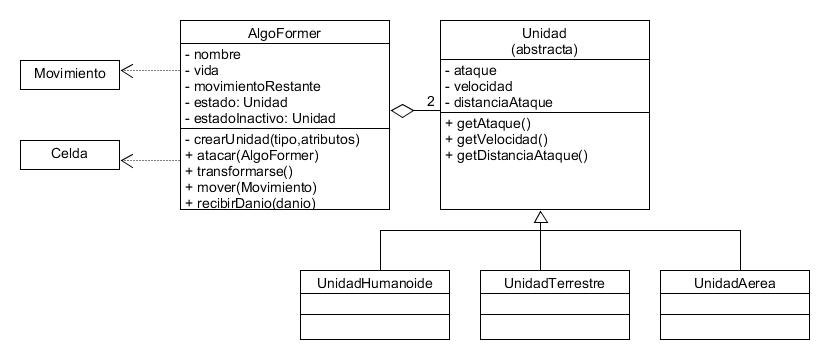
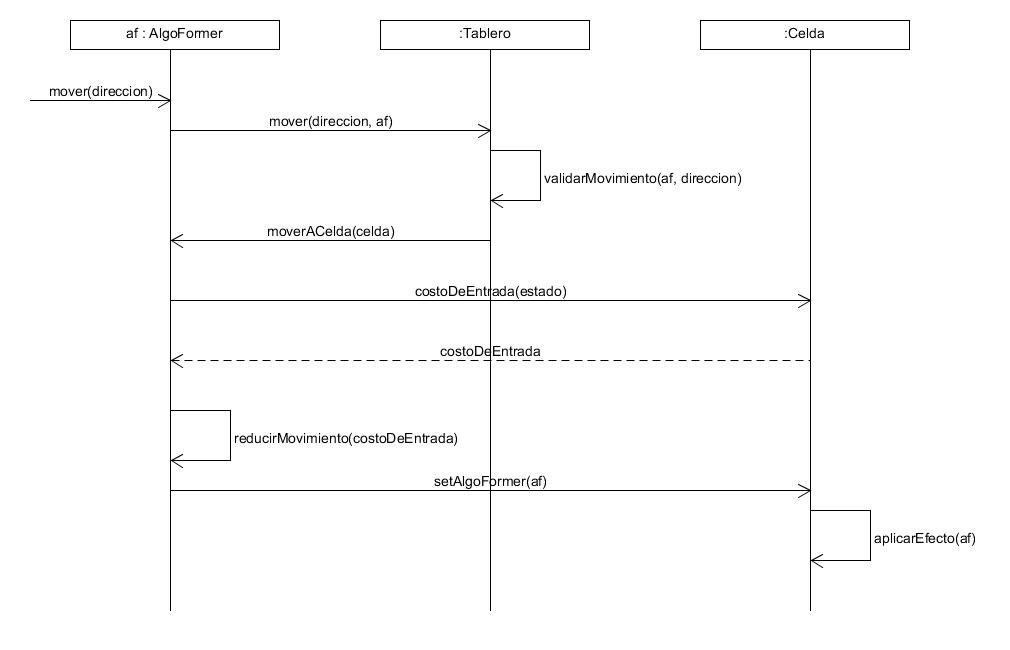
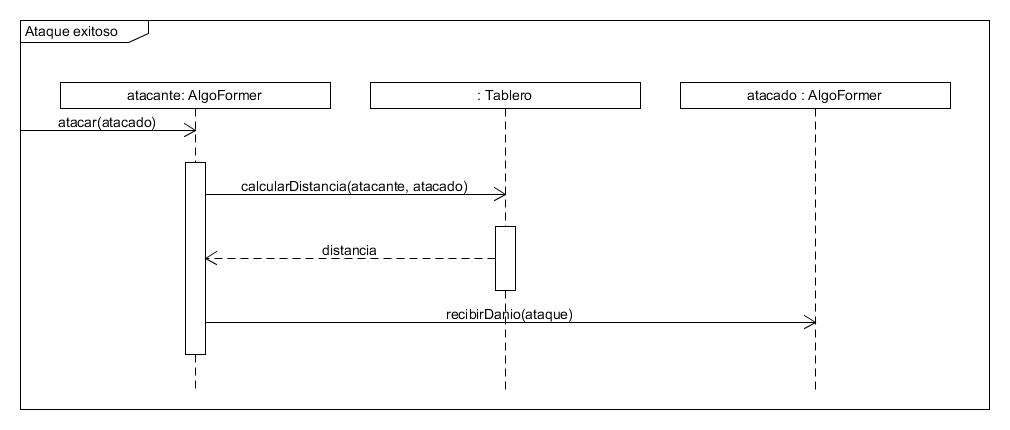


Diagrama de clases AlgoFormer-Unidad

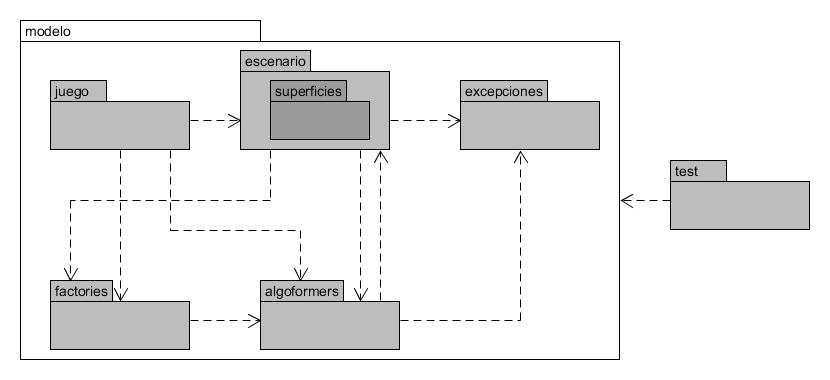


**Diagramas de secuencia**

**

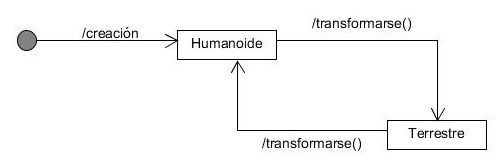
**

**Diagrama de paquetes**



**Diagramas de estado**

Diagrama de estado que representa la transformación de un AlgoFormer a su estado alterno y viceversa (el estado alterno puede ser tanto Terrestre como Aéreo).



**Detalles de implementación**

Los puntos más conflictivos que encontramos durante esta primera parte del desarrollo son la creación de los AlgoFormers y el movimiento de los mismos.

Con la creación nos referimos a que existen seis personajes específicos que tienen que existir en el juego, y si bien la solución que primero se nos ocurrió fue hacer una clase que herede de AlgoFormer para cada personaje, nos pareció que era una forma de implementación muy difícil de mantener (si se quisiera agregar más personajes, por ejemplo). Por lo que decidimos que los AlgoFormers se creen a partir de una clase AlgoFormerFactory, que contenga toda la información de los AlgoFormers existentes y que, cuando se lo requiera, devuelva un AlgoFormer ya creado.

Por otro lado, el movimiento fue un poco conflictivo también ya que un AlgoFormer tiene una cantidad de movimientos máximos por turno que no puede sobrepasarse, además de que hay muchos caminos para llegar desde la celda A hasta la celda B, con distintas consecuencia cada uno. Para tratar estos problemas, decidimos que el usuario se irá moviendo de a un casillero, eligiendo paso a paso por donde moverse, por lo que será su responsabilidad elegir con que camino llegar hasta la celda B; al mismo tiempo, se irá contando la cantidad de movimientos que se hicieron en el turno y verificando que no supere la velocidad del personaje, y si esto ocurriera, el AlgoFormer ya no se moverá por más que el usuario lo requiera.

**Excepciones**

**AlgoformerFueraDeAlcanceException:** es lanzada cuando un AlgoFormer intenta a atacar a otro que está más lejos que lo que su distancia de ataque le permite alcanzar.

**NoHayMasMovimientosException:** el movimiento de los AlgoFormer es de a un casillero por vez, por eso, cuando se intenta hacerlo avanzar más casilleros que lo que su velocidad le permite en un solo turno, esta excepción es lanzada.

**CeldaOcupadaException:** es lanzada cuando se intenta colocar o mover a un AlgoFormer en una celda del tablero que ya está ocupada por un AlgoFormer.

**PosicionInvalidaException:** es lanzada cuando se intenta colocar o mover a un AlgoFormer a una posición inválida, es decir, que no existe en el tablero.

**FriendlyFireException:** es lanzada cuando se intenta atacar a un AlgoFormer del mismo equipo.