Trabajo Final - Estructuras de Datos y Algoritmos 2017Q1

Enunciado y requerimientos

Se deberá realizar un juego de Skirmish Wars entre dos jugadores. Cada jugador corre el programa en una máquina distinta.

El juego se conecta mediante la librería Boost.Asio en modo TCP/IP utilizada durante el cuatrimestre. El protocolo de comunicación está definido por la cátedra y sus detalles se encuentran más adelante.

La interfaz del usuario deberá ser gráfica y se le deberá permitir al usuario controlar el juego con el mouse empleando la librería Allegro con la que se trabajó durante todo el cuatrimestre.

Será imperativa la correcta modularización del programa y el uso de C++ ya que simplifica la programación.

Se emplearán las reglas de Skirmish Wars que se encuentran en el sitio http://skirmishwars.wikidot.com/start con algunas excepciones que serán aclaradas más adelante. Se incluirán también links con explicación de las reglas y detalles del juego para su correcta comprensión.

No deberá crearse un "jugador virtual" de "Skirmish Wars, sino que el problema se reduce a lograr una correcta comunicación entre dos máquinas, no permitir que ningún jugador viole las reglas que se explicarán en breve, detectar los distintos resultados del juego (ganar o perder), diseñar un entorno gráfico interactivo e intuitivo que se pueda manejar mediante el mouse y detectar silencios suficientemente pronunciados como para entender que se ha perdido la comunicación entre las máquinas.

Detalles importantes (puntos principales de evaluación):

- Correcta modularización.
- Uso de objetos cuando corresponda (su uso desmedido puede tornarse catastrófico).
- Separación clara entre el front-end y el back-end.
- Correcta diferenciación entre el "motor" del juego y el manejo de cada periférico.
- Interfaz gráfica empleando Allegro (front-end).
- Uso del mouse empleando Allegro (back-end).
- Uso del canal de comunicaciones empleando APR (back-end).
- Uso de timers empleando Allegro (back-end).
- Respeto absoluto del protocolo de comunicación (la interoperabilidad entre versiones de 2 grupos será considerada crucial).
- Aplicación de todos los conceptos vistos en clase según corresponda: ***la aplicación de cada concepto supone un ámbito particular en el que aplicarlo; se evaluará el ámbito en que se aplicó el concepto tanto como la correcta aplicación del mismo.***

Dinámica del juego

La dinámica se encuentra en http://skirmishwars.wikidot.com/start. Y tomaremos dichas reglas como las oficiales de Skirmish Wars para nuestra implementación. Sin embargo, con el afán de mejorar la claridad de las reglas y facilitar el juego vamos a repasarlas y cuando corresponda las modificaremos:

La introducción al juego (ítem 1) explica principalmente dos objetivos: tomar el Head Quarters (HQ) enemigo o eliminarle todas sus tropas. Nosotros vamos a adoptar estos dos objetivos como únicos objetivos. Así, el primer jugador en alcanzar cualquiera de dichas condiciones ganará. Para nosotros no existe ninguna otra forma de ganar el juego que no sea alguna de estas dos.

Continuando con las reglas, llegamos a la parte de Set Up (ítem 2). Vamos a modificar esta regla adoptando diez mapas de 12 filas por 16 columnas ya definidos (serán elegidos en clase y se encontrarán comprimidos dentro del archivo "Skirmish Wars Maps.zip" junto con la consigna). En todos los casos, empezamos con $5 y el primer jugador será sorteado según se especifica más adelante. Las condiciones que marcan la victoria ya fueron definidas arriba.

Las reglas ahora contemplan la dinámica del juego propiamente dicha (ítem 3). Notar que los $5 iniciales pueden pensarse como el primer paso del primer turno de cada uno de los jugadores si estos tuvieran nada más que el HQ. Si además tienen casas propias para este primer turno no cuentan. Por lo tanto, para el primer turno, el primer paso resulta en $5 siempre. Para los demás turnos se debe seguir la regla que estipula el site: "recibirá al principio del turno $5 por cada ciudad propia que controla el jugador más $5 por el HQ".

Las reglas explicadas en los ítems 3.2, 3.3, se adoptarán sin modificación alguna. Resultan suficientemente claras como para comentar algo más de ellas aquí.

Por otro lado, vamos a aclarar la regla que explica el ítem 3.4a. En su último punto este ítem establece que la unidad atacada puede contratacar si está en rango. Esto debe leerse como que la unidad atacada va a contratacar siempre que del combate no resulte eliminada y la unidad atacante esté en su rango. Como establece claramente el ítem 3.4a si la unidad atacada fuera reducida a causa del combate esta contraataca como reducida y no como unidad normal aunque antes del ataque así lo fuera. En el ejemplo de batalla que sigue al ítem 3.4b se explica claramente este concepto.

Vamos a agregar una regla denominada "Fog of War". Esta regla establece que inicialmente los casilleros del tablero resultan desconocidos para ambos jugadores, y por lo tanto, no podrán ver el contenido de estos (no podrán saber el tipo de terreno de cada casillero ni tampoco conocer si hay unidades enemigas allí). Quedan exceptuadas de esta regla todas las construcciones y unidades propias. A su vez, las unidades propias podrán "ver" un casillero a la redonda en forma ortogonal. A medida que la unidad avanza esta va descubriendo el perímetro que puede "ver" y se revela para el jugador dueño de dicha unidad el terreno en el mapa. Una vez disipada la niebla de guerra de un casillero esta no reaparece; el jugador podrá ver todo lo que ocurre en el casillero que ya no cuenta con "Fog of War".

Las unidades sólo pueden moverse a casilleros visibles (sin fog of war). Por lo tanto, inicialmente las unidades deberán avanzar paso a paso. A medida que se disipa la niebla de guerra se pueden realizar movimientos que involucren más de un casillero. Esta regla no implica que inicialmente las unidades tienen menos MPs, sino simplemente que por ejemplo para moverse 3 casilleros el jugador deberá indicarlo 3 veces si hubiera fog of

war (para ir descubriendo el camino) en lugar de una sola vez si no la hubiera y sus MPs se lo permitieran.

Si una unidad recibe un ataque y la unidad atacante se encuentra en 1 casillero que posee niebla de guerra para el jugador de la unidad atacada, el jugador que reciba el ataque verá momentáneamente la unidad atacante; sin embargo no podrá ver el terreno donde esta se encuentra (si bien su máquina lo conoce para poder calcular los daños no debe ser revelado al jugador). Este verá la unidad atacante sobre un fondo oscuro que no revela el terreno. Terminado el ataque desaparece de la vista la unidad atacante y nuevamente el casillero vuelve al estado anterior al ataque para el jugador dueño de la unidad que recibiera el ataque (con fog of war). Por favor referirse al Anexo 2 que muestra algunos ejemplos de esta regla.

Finalmente, llegamos al punto de determinar un campeón. Siguiendo el concepto explicado arriba, las posibles victorias son: conquistar el HQ del oponente o eliminar todas sus unidades. El primer jugador en alcanzar alguno de estos objetivos será el victorioso. Vamos a adoptar esta regla ignorando lo que dicen las reglas del site del ítem 4.0 en adelante.

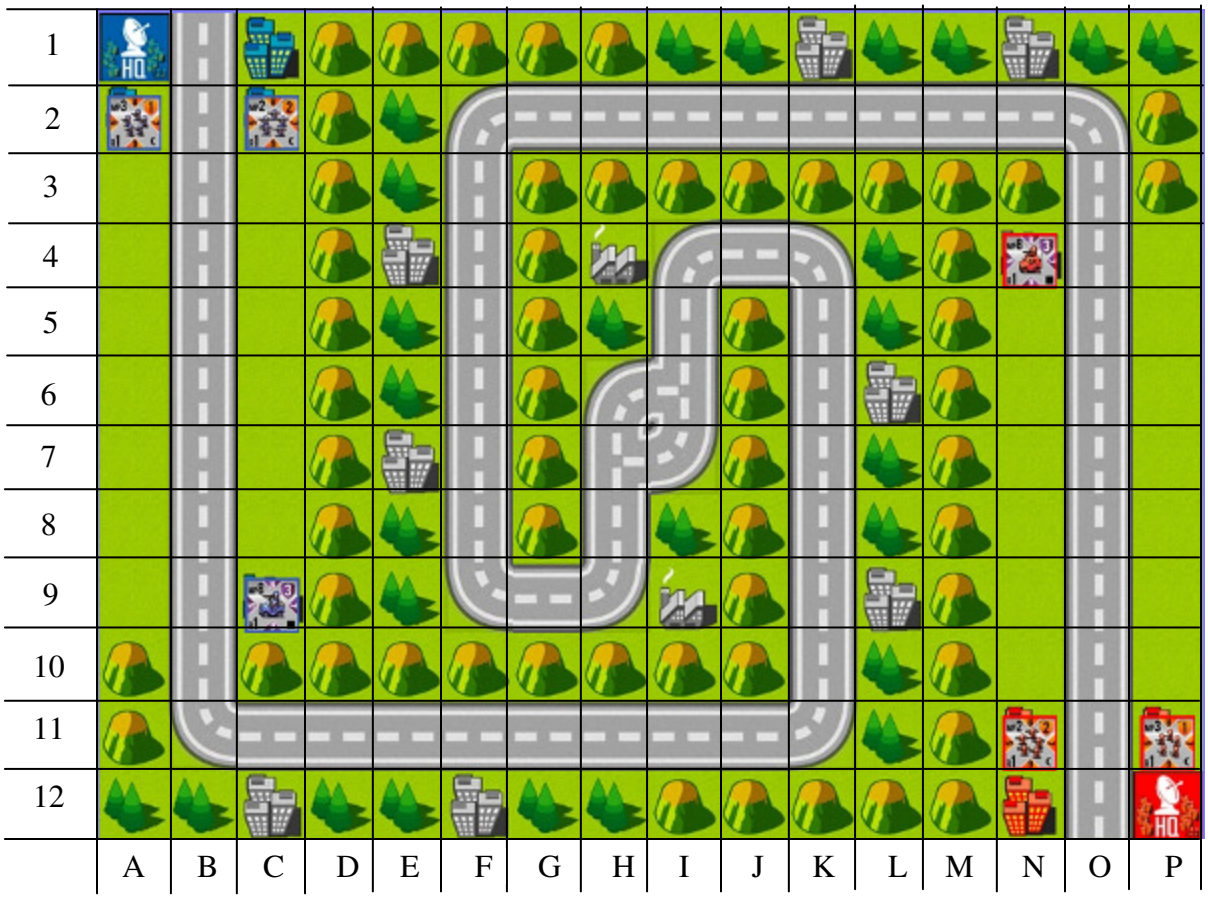
Por último, se incluirá una regla adicional que estipula un tiempo máximo en el que el jugador debe realizar una jugada (ya sea un movimiento, un ataque, una captura o una compra) y será de un (1) minuto. Si un jugador no realizara una jugada dentro del minuto se considerará que su turno terminó pasándoselo al oponente. Por cada jugada realizada se deberá reiniciar la cuenta (cada jugador tiene un minuto para pensar cada una de sus jugadas mientras las ejecuta).

Convenciones y definiciones

Para estandarizar el proceso de transmisión de mensajes vamos a definir algunas convenciones que van a ayudar a que dos programas realizados por grupos distintos se puedan comunicar entre si sin lugar a ambigüedades.

La primera regla que vamos a definir consiste en el método para nombrar las jugadas. Una máquina deberá informarle a la otra la jugada a realizar, y para evitar confusiones se adoptarán los siguientes criterios:

1. En **ambas** máquinas el layout del tablero será el mostrado en la figura 1. (tomamos como ejemplo uno de los diez mapas definidos arriba) recodar según lo establecido en ***Dinámica del juego*** todos los mapas tienen 12 filas por 16 columnas.
2. En **ambas** máquinas el juego lo inicia el jugador 1, representado por el color rojo.
3. Se diferenciarán los casilleros siguiendo el sistema de coordenadas de abajo donde se anota primero la fila con números del 1 al 12 (1, 2, 3, 4, …, 12) y luego la columna con letras mayúsculas de la "A" a la "P" (A, B, C, D, …, P):



4. Para especificar un movimiento, se indica primero la coordenada (fila y columna) en donde se encuentra la tropa y luego la coordenada (fila y columna) en donde se quiere colocar. Como ambas máquinas conocen la totalidad del tablero no es necesario especificar la unidad que se está moviendo.

5. El desembarco de unidades de un APC se indica de la siguiente manera: primero se indica un movimiento del ACP del casillero donde está a sí mismo (por ejemplo si el ACP se encuentra en el casillero 9B, se indica un movimiento 9B a 9B). Esto le indica al contrincante que el ACP no desea realizar más movimientos sino desembarcar unidades. Luego con uno o dos movimientos adicionales (desde donde se encuentra el ACP a los 4 casilleros de la periferia) se indica el desembarco de una o dos unidades según corresponda. Las unidades se desembarcan en el orden inverso en que fueron embarcadas. Para abordar un APC se mueven las unidades al casillero en donde se encuentra el ACP. Se pueden embarcar hasta dos unidades según la ***Dinámica del Juego***.
6. Los ataques, se especifican indicando la coordenada del atacante (fila y columna) seguida de la coordenada del atacado (fila y columna) y finalmente un número de 0 a 6 que indica el valor del resultado de arrojar un dado de seis caras (1 a 6) o 0 si no correspondiera utilizarlo. Dicha información sale del cuadro TERRAIN DEFENSE MODIFIERS que establecen las reglas del juego
7. El inicio de una captura se especifica igual que los ataques. Si en los turnos sucesivos el jugador no retira su tropa de la construcción siendo capturada continúa el proceso de captura hasta completarla o hasta que el jugador retire su unidad de dicha propiedad. No es necesario, en cada turno, volver a especificar que se está capturando la construcción.
8. Los mapas se especifican en un archivo de texto estilo CSV (comma separated values). El formato de este archivo y cómo nombrar las unidades se encuentran en el ***Anexo 1***.
9. Para especificar una compra primero se indica la nueva unidad adquirida (siguiendo el nombramiento de unidades del Anexo 1) y luego la coordenada (fila y columna) en donde se ubica dicha unidad.

La segunda regla que vamos a definir refiere a la selección de la jugada. Como se explicó en la introducción del trabajo práctico, se empleará el mouse como interfase entre el usuario y el tablero. Esta regla contempla los siguientes criterios:

1. El usuario podrá posicionar el cursor sobre cualquier unidad del tablero.
2. Seleccionará la unidad que desea jugar (el programa solo le permitirá seleccionar unidades válidas de acuerdo a las reglas referenciadas en ***Dinámica del juego***) realizando un click con el botón izquierdo del mouse.
3. Luego queda a criterio de los alumnos la operatoria que deberá realizar el usuario para indicarle al programa qué es lo que se quiere realizar.
4. El programa solo realizará la jugada si esta fuera válida. Caso contrario le informará la invalidez de la jugada y esperará un nuevo input.
5. Una vez que se realizó la jugada el jugador no podrá retractarse para realizar otra.

Vamos a definir también, una tercera regla referente a la operación del juego:

1. El sistema le deberá informar al usuario lo que sucede en **todo momento del juego**.
2. Deberá proveerle una forma de abandonar el juego en cualquier momento.
3. Además, una vez comenzado el juego, se deberá llevar un registro de la salud de cada unidad y ofrecerle al usuario una forma visible de conocer la salud de todas las unidades (tanto las propias como las enemigas).

4. Para facilitarle el juego al usuario, se le deberán mostrar las opciones que este puede jugar cuando elije cada unidad en una forma amigable e intuitiva que queda a criterio de los alumnos. Recordar la limitación que impone el fog of war al principio del juego y respetarla.
5. Respetando la secuencia de actividades descripta en ***Dinámica del Juego***, el usuario debe comprar una vez que terminó (o ya no desea) mover y atacar. Una vez que el jugador realiza una compra no puede mover o atacar. De la misma forma, debe atacar una vez que terminó (o ya no desea) mover la unidad atacante. Una vez que atacó no puede seguir moviendo la unidad atacante. Por lo tanto, si el usuario realizara una jugada que genera la inhibición de otra (comprar cuando todavía puede mover o atacar, atacar cuando todavía puede mover con la unidad atacante, etc.) el programa deberá pedirle una confirmación de lo que está realizando advirtiéndole del riesgo de hacerlo. Queda a criterio del alumno incluir la posibilidad de "no volver a mostrar este mensaje" para darle la opción al usuario de "customizar los warnings".

   También es preciso definir un marco de tiempos. A continuación definiremos los criterios a tener en cuenta para el tiempo de cada jugada:

   1. El jugador tiene un tiempo máximo de 1 minuto para realizar cada jugada desde que empieza el turno. Tal como se estableció en ***Dinámica del juego***.
   2. Si el usuario no realiza una jugada dentro del minuto se considerará que su turno terminó, y automáticamente se pasará el turno a su contrincante.
   3. El sistema deberá mostrar en todo momento el tiempo restante. Deberá advertirle al usuario cuando falte medio minuto que ya se ha consumido la mitad de su tiempo y cuando falten 10 segundos deberá resaltarlo de alguna manera para alertar al jugador.

Por último, vamos a adoptar una regla referente a lo que se ha de transmitir entre máquinas. A continuación los criterios que hacen a esta regla:

1. Las máquinas solo transmitirán jugadas válidas, según lo descripto en ***Dinámica del juego***.
2. La máquina que recibe la jugada, antes de representarla en el tablero la deberá analizar. Si la jugada fuera inválida será tomada como un error de comunicación y se dará por terminado el juego (ver ***Protocolo de comunicación*** para más detalles). Caso contrario será representada en el tablero antes de darle la opción al usuario de realizar su jugada.
3. Cada jugada será enviada inmediatamente después de haber sido realizada. No debe esperarse que el usuario pase el turno para enviar todas las jugadas juntas. Recordar que una vez realizada la jugada esta no puede ser deshecha.
4. La transmisión entre máquinas estará regida por el protocolo descripto a continuación bajo el título ***Protocolo de comunicación***. Si se violara alguna de las reglas de dicho protocolo se dará por terminado el juego informándole al usuario que hubo un error de sincronismo entre las máquinas.

Protocolo de comunicación

A continuación se detalla el protocolo de comunicación, tanto en handshake como en el tamaño, forma y contenido de los mensajes.

- ***Esperando conexión***

  Se deberá iniciar la comunicación a una dirección IP indicada por el usuario y al puerto 13225 que utilizaremos como puerto de Skirmish Wars.

  Primero se intentará como cliente. Si no se logra establecer la comunicación (del otro lado no hay nadie) se deberá generar un número aleatorio N entre 2000 y 5000 y la máquina esperará N milisegundos a que se establezca la comunicación como cliente.

  Si ello no sucediera se cancelará el intento como cliente para iniciar en modo servidor. El servidor esperará entonces a que se establezca una comunicación, que el usuario cancele el programa o que surja algún error suficientemente catastrófico como para que el programa entienda que el sistema no se puede recuperar y deba abortar su ejecución.

  Vale recordar los ítems 1 y 2 de la tercera regla definida en ***Convenciones y definiciones***, donde se establece que se deberá informar de todo lo que sucede al usuario y se le permitirá salir del programa en cualquier circunstancia.

  Una vez establecida la comunicación se pasará a ***Máquinas conectadas***.

- ***Maquinas conectadas***

  La máquina que al momento de establecerse la conexión había sido iniciada en modo servidor pasa entonces a gobernar la secuencia del juego y la llamaremos servidor. Su contrincante se llamará cliente.

  La primera acción que debe realizar el servidor es pedir el nombre del cliente enviando un paquete NAME (paquete que no posee campo de datos, solamente consiste en un encabezado de un byte).

  El cliente al recibir el paquete NAME deberá contestar con un paquete NAME_IS conformado de la siguiente manera: un byte de encabezado con el valor definido por NAME_IS seguido de un campo de datos de longitud variable. Éste consiste en: un byte que indica la longitud del nombre, en formato unsigned char (de 0 a 255 caracteres de largo) seguido de una cadena variable de bytes que contienen el nombre del jugador en formato ASCII.

  No deberá enviarse ningún carácter no imprimible en el nombre del jugador. Si bien la longitud máxima del nombre está limitada a 255 caracteres no es necesario imprimir todos en la pantalla (está permitido truncar el nombre).

  El paquete de NAME_IS toma entonces la siguiente forma (1 byte por celda):

NAME_IS	**COUNT**	***Nombre***				
0x11	0x05	A(0x41)	R(0x52)	I(0x49)	E(0x45)	L(0x4C)

Una vez que el servidor recibe el nombre del oponente (paquete NAME_IS) contesta con un ACK (paquete que tampoco contiene campo de datos, se constituye simplemente de un encabezado de 1 byte).

Es ahora el cliente quien debe averiguar el nombre del servidor. Por lo tanto, al recibir el ACK le enviará al servidor un paquete NAME. El servidor contestará con el paquete NAME_IS de la misma forma que antes lo había hecho el cliente y éste último contestará con un ACK al recibir el paquete NAME_IS del servidor.

Cuando que el servidor reciba el ACK a su paquete NAME_IS sorteará aleatoriamente uno de los diez mapas que acompañan a esta consigna según lo definido en ***Dinámica del juego*** enviará entonces un paquete MAP_IS conformado de la siguiente manera: un byte de encabezado con el valor definido por MAP_IS seguido de un campo de datos de longitud variable. Éste consiste en: un byte que indica la longitud del nombre, en formato unsigned char (de 0 a 255 caracteres de largo) seguido de una cadena variable de bytes que contienen el nombre del mapa en formato ASCII (el nombre de los mapas se especifica en el Anexo 1) y finalmente un bytes de checksum (que puede tomar valores de 0 a 255) que se debe calcular según lo definido en el Anexo 1.

El paquete de MAP_IS toma entonces la siguiente forma (1 byte por celda):

MAP_IS	**COUNT**	***Nombre del Mapa (ver Anexo 1) Ej: gkuracz_mapa1***					**CHECKSUM**
0x12	0x0D	g(0x67)	k(0x6B)	u(0x75)	(...)	1(0x31)	N/N ∈ [0x00; 0xFF]

El cliente verificará la existencia y el checksum del mapa al que hace referencia el servidor y de ser válido responderá ACK. Caso contrario responderá con un error siguiendo la política de errores que se especifica más adelante.

Por último, una vez que el servidor recibe el ACK a su paquete MAP_IS sorteará aleatoriamente quien comenzará el juego.

De ser el cliente el elegido, el servidor transmitirá el paquete YOU_START (que carece de campo de datos) mientras que de lo contrario transmitirá el paquete I_START (que también carece de campo de datos). El cliente solo responde con un paquete ACK si recibiera I_START. Caso contrario envía directamente su primera jugada.

Desde este momento deja de existir la distinción entre cliente y servidor.

- ***Comienzo del juego***

Cuando el jugador que empieza (según se explicó arriba el cliente si recibe YOU_START o el servidor en caso contrario) realice su primera jugada, la misma se transmitirá con uno de cuatro paquetes: MOVE, PURCHASE, ATTACK o PASS.

A los dos primeros el usuario responderá ACK excepto que detecte un error en alguna jugada en cuyo caso responderá con un error siguiendo la política de errores que se especifica más adelante.

El paquete MOVE está conformado de la siguiente manera: un byte de encabezado con el valor definido por MOVE seguido de dos bytes que indican la fila y la columna de origen de la pieza y por último dos bytes que indican la fila y columna del destino de la unidad.

El paquete de MOVE toma entonces la siguiente forma (1 byte por celda):

MOVE	**FILA_OR**	***COL_OR***	***FILA_DE***	***COL_DE***
0x31	N/N ∈ [0x00; 0x0C]	N/N ∈ [0x41; 0x50]	N/N ∈ [0x00; 0x0C]	N/N ∈ [0x41; 0x50]

El paquete PURCHASE está conformado de la siguiente manera: un byte de encabezado con el valor definido por PURCHASE seguido de dos bytes que indican la unidad comprada (En el ítem 9 del apartado **Especificación de la notación de los mapas del Anexo1** se explica cómo nombrar cada unidad; nótese que hace falta indicar el equipo al que pertenece pues se sabe qué jugador la está comprando) y por último dos bytes que indican la fila y columna en donde se coloca dicha unidad.

El paquete de PURCHASE toma entonces la siguiente forma (1 byte por celda):

PURCHASE	**UNIDAD (1ª letra)**	**UNIDAD (2ª letra)**	***FILA***	***COLUMNA***
0x32	N/N ∈ ver Anexo 1	N/N ∈ ver Anexo 1	N/N ∈ [0x00; 0x0C]	N/N ∈ [0x41; 0x50]

Al paquete ATTACK el contrincante responderá con otro paquete ATTACK con el contraataque excepto que detecte un error en la jugada en cuyo caso responderá con un error siguiendo la política de errores que se especifica más adelante.

El paquete ATTACK está conformado de la siguiente manera: un byte de encabezado con el valor definido por ATTACK seguido de dos bytes que indican las coordenadas (fila y columna) en donde se encuentra la unidad atacante, dos bytes que indican la fila y columna en donde se encuentra la unidad atacada y por último un byte con el valor del dado si correspondiera. En caso de que la unidad atacada resulte eliminada por el ataque se deberá igualmente responder con un paquete ATTACK de contraataque que será ignorado por el atacante.

El paquete de ATTACK toma entonces la siguiente forma (1 byte por celda):

ATTACK	**FILA_OR**	***COL_OR***	***FILA_DE***	***COL_DE***	***DADO***
0x33	N/N ∈ [0x00; 0x0C]	N/N ∈ [0x41; 0x50]	N/N ∈ [0x00; 0x0C]	N/N ∈ [0x41; 0x50]	N/N ∈ [0x00; 0x06]

Al recibir el paquete ACK de su contrincante, la máquina que empezó el juego volverá a mandar cualquiera de los tres paquetes de arriba reflejando las acciones que realiza el jugador hasta que este decida pasar el turno o se haya cumplido un minuto de inactividad entre sus jugadas. En ese caso se manda el paquete PASS que carece de campo de datos.

Cuando una máquina recibe un paquete PASS le plantea a su jugador que es su turno y comienzo de nuevo el ciclo. No debe responder al paquete PASS con un ACK sino que directamente se debe enviar algunos de los cuatro paquetes mencionados arriba. Se van alternando así los roles de ambos jugadores.

- ***Final del juego***

El juego termina ante alguna de las siguientes dos situaciones: cuando una máquina al recibir un ataque detecta que este concluye el juego ya que elimina a su última unidad o cuando una máquina al recibir el paquete PASS detecta que una unidad contrincante tomó su HQ. Ante ambas situaciones la máquina derrotada debe responder con un paquete YOU_WON. En el caso en el que la maquina atacante fuera la victoriosa, la atacada en lugar de contraatacar (enviando un paquete ATTACK) enviará el paquete YOU_WON. Si fuera el contraataque el que produce la victoria, el usuario que inició el ataque responderá YOU_WON al recibir el contraataque en lugar de enviar algún otro movimiento.

Al igual que los paquetes NAME, ACK, YOU_START y I_START, el paquete YOU_WON está constituido únicamente por un encabezado de un byte.

La máquina que recibe el paquete YOU_WON interrogará a su jugador para saber si desea volver a jugar.

En caso afirmativo enviará un paquete PLAY_AGAIN; de lo contrario mandará GAME_OVER. Los paquetes PLAY_AGAIN y GAME_OVER tampoco poseen campo de datos, únicamente un encabezado de un byte.

La máquina que recibe el paquete PLAY_AGAIN, procederá luego a interrogar a su usuario si desea volver a jugar.

En caso afirmativo, sorteará aleatoriamente uno de los diez mapas que acompañan a esta consigna según lo definido en ***Dinámica del juego*** y enviará un paquete MAP_IS.

Su contrincante verificará la existencia y el checksum del mapa al que hace referencia el paquete MAP_IS recibido y de ser válido responderá ACK. Caso contrario responderá con un error siguiendo la política de errores que se especifica más adelante.

Al recibir el paquete ACK quien envió previamente el paquete MAP_IS enviará ahora I_START o YOU_START según corresponda. Para definir quien empieza la próxima partida se empleará el criterio "si se juega más de una partida se alternarán los turnos". Al recibir cualquiera de los dos paquetes, se vuelve a cumplir la condición planteada en el último párrafo de ***Máquina conectadas***. (La máquina que los recibe contestará con ACK o algunos de los cuatro paquetes que representan las jugadas).

Si el usuario de la máquina que recibió el paquete PLAY_AGAIN no desea seguir jugando esta contestará con GAME_OVER.

La máquina que recibe el paquete GAME_OVER le explicará a su usuario que del otro lado no desean seguir jugando y contestará con un paquete ACK. En ese momento ambas máquinas se desconectarán cerrando la comunicación entre ellas.

- ***En cualquier momento de la comunicación***

  Cuando la comunicación se encuentre en la etapa descripta en ***Comienzo del juego*** si una maquina no recibe un paquete de respuesta valido después de 2 minutos y medio de enviado el propio, el juego se considera colgado y se finaliza advirtiéndole al usuario que hubo un error de comunicación (se toma un margen de medio minuto para ecualizar cualquier diferencia de relojes que pudiera existir entre máquinas así como el tiempo de tráfico).

  Por otro lado, no se impondrá límite de tiempo fuera de la etapa ***Comiendo del juego*** cuando se espere input del usuario (por ejemplo que envíe su nombre o decida si desea volver a jugar). Para los paquetes de respuesta al input de usuario (ACK, NAME, etc) se estipula un tiempo de 2 minutos y medio también.

  Pueden existir diversos errores de comunicación: por ejemplo, no se ha respondido dentro del tiempo estipulado, se recibe un paquete erróneo, una jugada inválida, un checksum de mapa erróneo, un paquete inexistente, etc.

  La máquina que detecta un error de comunicación enviará un paquete de ERROR y luego cerrará el canal de comunicación abierto. No esperará un ACK del otro lado ya que no puede establecer con certeza que la comunicación no se haya corrompido.

  En todos los casos si se recibe un paquete de ERROR se debe cerrar el canal de comunicación indicándole al que hubo un error en la comunicación.

  Se le debe dar al usuario la posibilidad de cerrar el programa en cualquier momento. En dicho caso deberemos informárselo apropiadamente al usuario remoto enviándole el paquete QUIT.

  Al recibir un paquete de QUIT, la máquina deberá responder con un paquete ACK, informarle al usuario lo sucedido y cerrar el canal de comunicación. Vale aclarar que la respuesta al paquete QUIT con un paquete ACK también debe circunscribirse en el marco de tiempo de 2 minutos y medio, siendo así, si después de 2 minutos y medio de enviado el paquete QUIT, no se recibiera el paquete ACK del contrincante, se deberá concluir que hubo un error y se procederá a informarlo de la manera descripta arriba.

  Nótese que los paquetes QUIT y ERROR no poseen campo de datos. Están constituidos únicamente por un encabezado de un byte.

- ***Definición de los encabezados de cada paquete:***

Comando	*Valor (hexadecimal)*
ACK	0x01
NAME	0x10
NAME_IS	0x11
MAP_IS	0x12
YOU_START	0x20
I_START	0x21
PASS	0x30
MOVE	0x31
PURCHASE	0x32
ATTACK	0x33
YOU_WON	0x40
PLAY_AGAIN	0x50
GAME_OVER	0x51
ERROR	0xFE
QUIT	0xFF

Anexo 1 - Mapas

Especificación de la notación de los mapas

1. El mapa se debe armar en un archivo de texto tipo CSV y su extensión debe ser “,csv”.
2. Se constituye de 12 filas por 16 columnas
3. El separador puede ser tanto un carácter ‘,’ como ‘;’.
4. El terminador de línea puede ser tanto un CR (13d) como un LF (10d) o combinaciones de ellos.
5. La tabla que hace al mapa empieza en la fila 1 columna A termina en la fila 12 columna P.
6. Cada accidente del mapa se representa con una letra.
7. El mapa es case incensitive (no distingue mayúsculas de minúsculas).
8. Terrenos:
    a. La tierra vacía (plain o tile) se especifica con la letra "t".
    b. Los ríos (rivers) se especifican con la letra "r".
    c. Las calles (road) se especifican con la letra "a".
    d. La selva (forest) se especifica con la letra "f".
    e. Las Montañas (hills) se especifican con la letra "h".
    f. Las fábricas neutrales (Manufacturing Facilities) se especifican con la letra "m" seguida del número "0": "m0"
    g. Las fábricas del Equipo 1 se especifican con la letra "m" seguida del número "1": "m1".
    h. Las fábricas del Equipo 2 se especifican con la letra "m" seguida del número "2": "m2".
    i. Las ciudades neutrales (Cities) se especifican con la letra "c" seguida del número "0": "c0"
    j. Las ciudades del Equipo 1 se especifican con la letra "c" seguida del número "1": "c1".
    k. Las ciudades del Equipo 2 se especifican con la letra "c" seguida del número "2": "c2".
    l. El Head Quarters del Equipo 1 se especifica con la letra "q" seguida del número "1": "q1".
    m. El Head Quarters del Equipo 2 se especifica con la letra "q" seguida del número "2": "q2".
    n. En ninguna celda de la matriz puede haber más de un tipo de terreno/ facility de los definidos en los puntos 6 a 18. Ej no está permitido que una celda contenga hm0. o tiene hills o tiene un neutral manufacturing facility.
9. Ejércitos:
    a. Además del tipo de terreno una celda puede contener 1 ejército.
    b. Se debe especificar el ejercito contenido adicionando el caracter "+" a continuación de la denominación del terreno seguido del tipo de ejército. SIN ESPACIOS- Ej: a+in1 (esta celda contiene una autopista y arriba de la autopista hay un Infantry que pertenece al Equipo 1).

c. No existen ejércitos neutrales por lo tanto todos deben ser especificados con el número del equipo al que pertenecen siguiendo su tipo. Ej: me1. Mech que pertenece al Equipo 1.
d. Los Infrantries se especifican con las letras "i" y "n" seguidas: "in".
e. Los Mechs se especifican con las letras "m" y "e" seguidas: "me".
f. Los Rockets se especifican con las letras "r" y "o" seguidas: "ro".
g. Los Recons se especifican con las letras "r" y "e" seguidas: "re".
h. Los APCs se especifican con las letras "a" y "p" seguidas: "ap".
i. Los Anti-Airs se especifican con las letras "a" y "a" seguidas: "aa".
j. Los Artillery se especifican con las letras “a” y “r” seguidas: “ar”
k. Los tanks se especifican con las letras "t" y "a" seguidas: "ta".
l. Los med-tanks se especifican con las letras "m" y "t" seguidas: "mt".

10. Se debe respetar siempre el orden. Primero se especifica el tipo de terreno y después, si hubiera, el ejército que contiene.

Nombramiento de los mapas

Los nombres de los mapas mencionados en ***Dinámica del Juego*** que acompañan a esta consigna serán determinados en el futuro en función de lo que se entregue en clase.

Los archivos .csv que contienen los mapas y que se encontrarán dentro del archivo comprimido “Skirmish Wars Maps.zip” serán nombrados nombrados de igual manera adicionándoles la extensión “.csv”. Vale aclarar que **la extensión no forma parte del nombre del mapa** (hecho que debe ser tenido en cuenta a la hora de intercambiar el nombre del mapa con nuestro oponente).

Verificación de los mapas (Checksum)

Para asegurarnos de que ninguno de los jugadores alteró el mapa y que por lo tanto ambas máquinas cuentan con la misma información, el servidor deberá enviar un checksum del archivo del mapa que luego será validado por el cliente en su propio archivo de mapa.

El checksum se calcula teniendo en cuenta todos los bytes que constituyen al archivo del mapa aplicando el siguiente algoritmo:

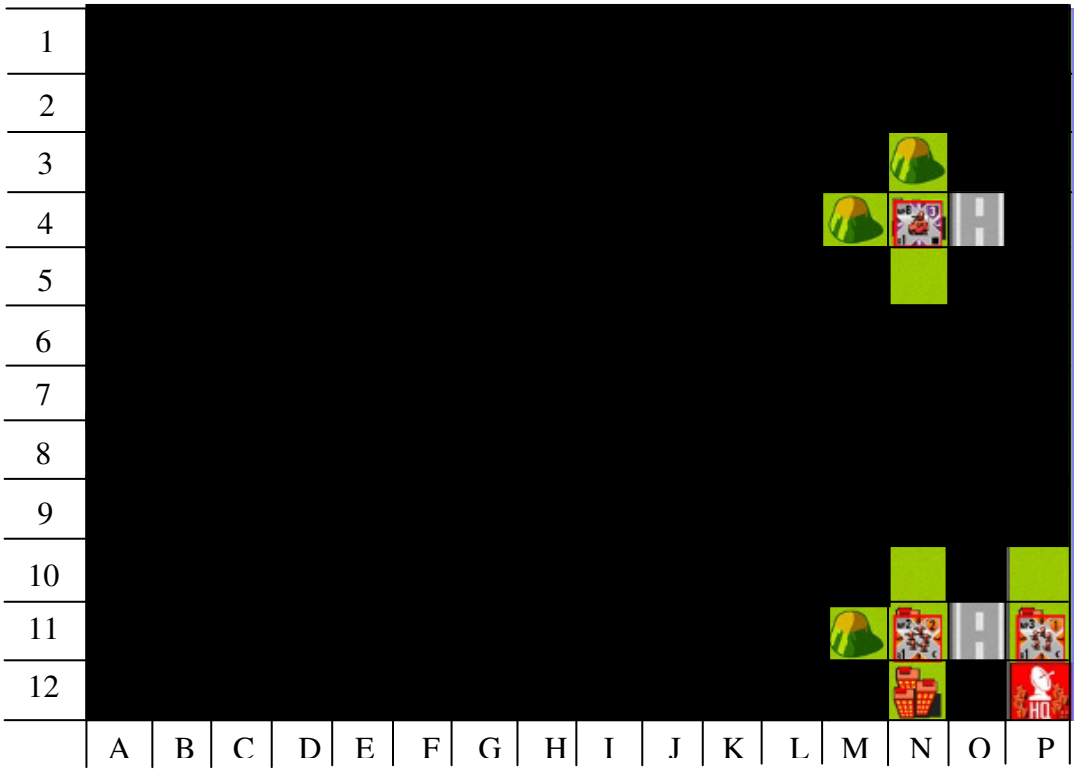
index(n) = Tabla[index(n-1) XOR Archivo(n)]
donde:
n toma valores de 1 hasta la longitud del archivo.
Archivo(n) refiere al byte n-ésimo del archivo.
index toma valores entre 0 y 255.
index(0) = 0.
Tabla constituye un arreglo de 256 bytes definidos que se especificarán en breve.

Resulta entonces checksum = index(longitud del archivo).

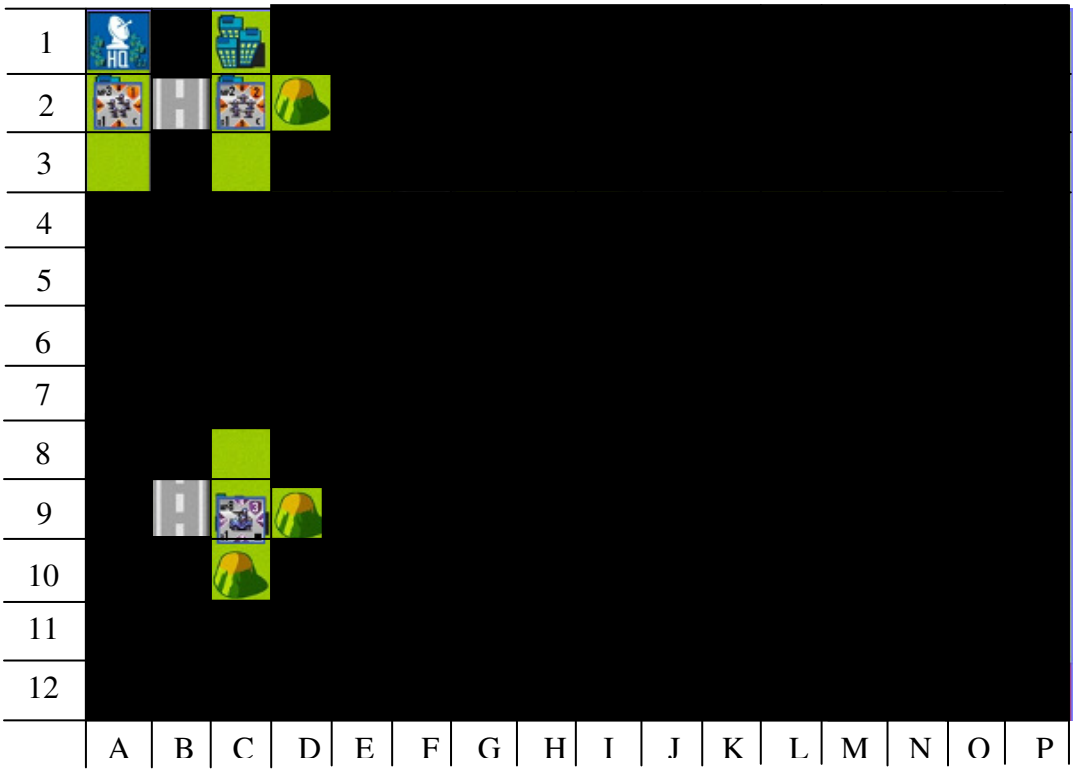
```
Tabla = {
     98,  6, 85,150, 36, 23,112,164,135,207,169,  5, 26, 64,165,219, //  1
     61, 20, 68, 89,130, 63, 52,102, 24,229,132,245, 80,216,195,115, //  2
     90,168,156,203,177,120,  2,190,188,  7,100,185,174,243,162, 10, //  3
    237, 18,253,225,  8,208,172,244,255,126,101, 79,145,235,228,121, //  4
    123,251, 67,250,161,  0,107, 97,241,111,181, 82,249, 33, 69, 55, //  5
     59,153, 29,  9,213,167, 84, 93, 30, 46, 94, 75,151,114, 73,222, //  6
    197, 96,210, 45, 16,227,248,202, 51,152,252,125, 81,206,215,186, //  7
     39,158,178,187,131,136,  1, 49, 50, 17,141, 91, 47,129, 60, 99, //  8
    154, 35, 86,171,105, 34, 38,200,147, 58, 77,118,173,246, 76,254, //  9
    133,232,196,144,198,124, 53,  4,108, 74,223,234,134,230,157,139, // 10
    189,205,199,128,176, 19,211,236,127,192,231, 70,233, 88,146, 44, // 11
    183,201, 22, 83, 13,214,116,109,159, 32, 95,226,140,220, 57, 12, // 12
    221, 31,209,182,143, 92,149,184,148, 62,113, 65, 37, 27,106,166, // 13
      3, 14,204, 72, 21, 41, 56, 66, 28,193, 40,217, 25, 54,179,117, // 14
    238, 87,240,155,180,170,242,212,191,163, 78,218,137,194,175,110, // 15
     43,119,224, 71,122,142, 42,160,104, 48,247,103, 15, 11,138,239  // 16
};
```

Anexo 2 – Ejemplos de Fog of War

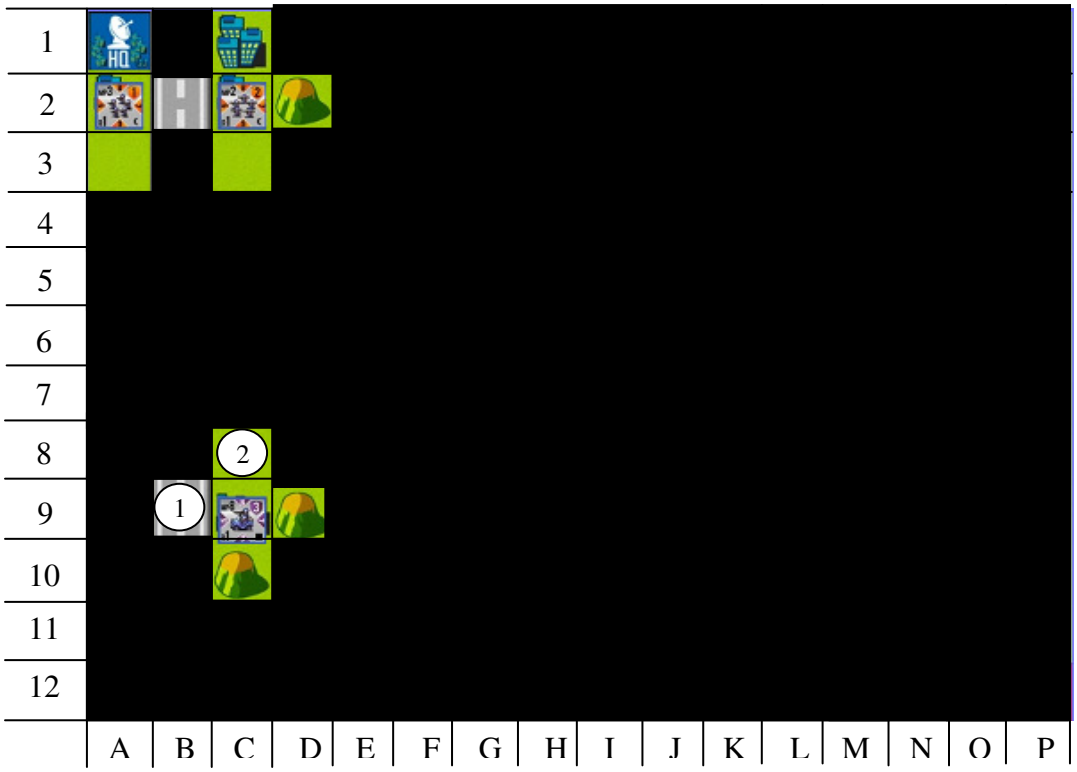
Vista inicial del mapa para el jugador Rojo



Vista inicial del mapa para el jugador Azul

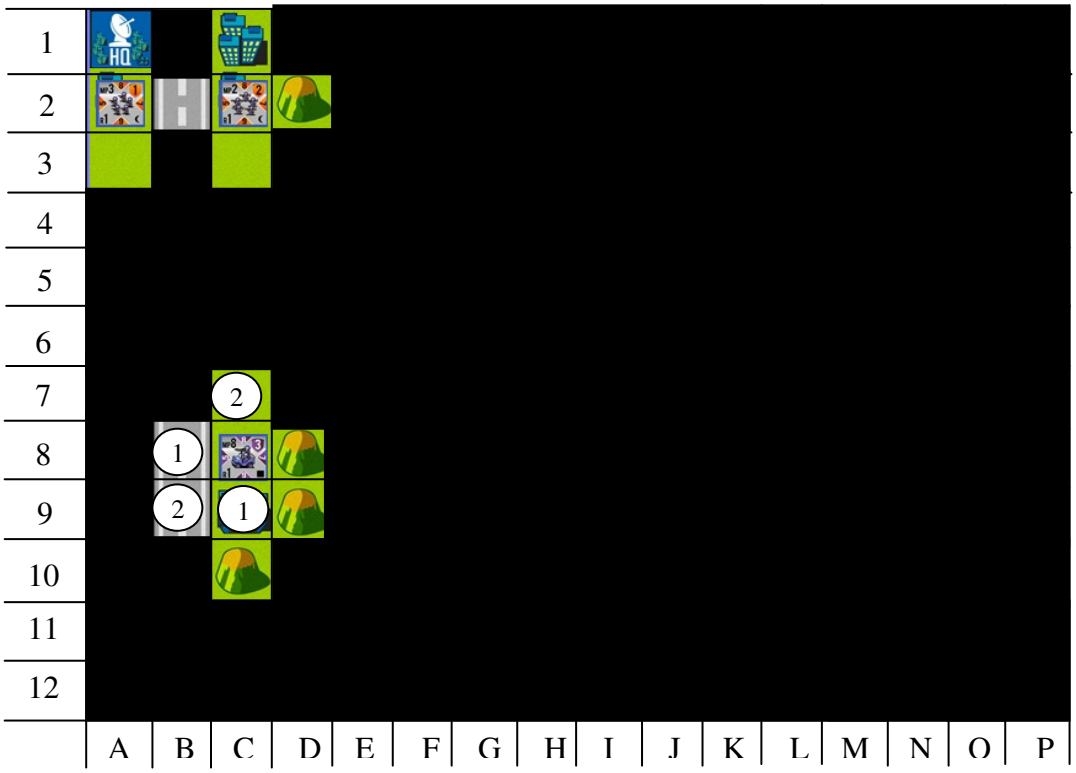


El Recon del jugador 2 (jugador azul) ubicado en 9C puede realizar los movimientos mostrados en el gráfico. El Recon tiene 8 MPs (ver carta de movimientos en http://www.mediafire.com/?sharekey=a23dba878ab0d711d6baebe61b361f7c330b0de6746487f1a9a26c4ed87536eb), sin embago, como el mapa tiene "fog of war", solamente puede realizar los movimientos mostrados en el gráfico. A saber, el Recon azul puede moverse hacia arriba (8C) con un costo es de 2 MP o hacia la izquierda (9B) por 1 MP. Hacia abajo (10C) o hacia la derecha (9D) no se puede mover.



Como el Recon tiene 8MPs, si el jugador 2 moviera a la izquierda (9B), se le restaría 1 MPs y se revelarían tres casilleros de su periferia (9A, 10B y 8B ya que 9C, el casillero donde estaba, ya es visible).

Si por otro lado, el jugador 2 moviera hacia arriba (8C), se le restarían 2MP al Recon azul y se revelarían 3 casilleros de su periferia (8B, 7C y 8D). Veamos este último ejemplo en un nuevo gráfico:



Al mover hacia arriba ahora se revelaron 3 nuevos casilleros (8B, 7C y 8D). El Recon azul posee 6 MPs luego de este movimiento (ya que consumió 2 para moverse hacia 8C).

Como no agotó sus MPs, el jugador azul puede elegir mover el Recon nuevamente en este mismo turno.

Sus opciones son:

- Hacia arriba (7C) pagando 2MPs,
- Hacia abajo (9C, volviendo a la posición donde estaba) pagando 1MP (pues las construcciones cuentan como rutas para movimientos y ataques),
- Hacia la izquierda (8B) pagando 1 MP.
- Al casillero 9B. En este caso la PC calcula el camino de menor costo dentro de los caminos posibles. Los caminos posibles son aquellos visibles (sin "Fog of War") que la unidad pueda atravesar según su carta de movimiento. En el ejemplo, ya sea pasando por 8B o por 9C el movimiento a 9B cuesta 2 MPs, pues ambos terrenos se consideran ruta.

Bibliografía

1. http://www.boardgamegeek.com/boardgame/40455/skirmish-wars-advance-tactics (Página "oficial" de Skirmish Wars).
2. http://skirmishwars.wikidot.com/start (Reglas del Skirmish Wars).
3. http://en.wikipedia.org/wiki/Pearson_hashing (Pearson Hashing. Checksum algo.)
4. http://www.mediafire.com/?sharekey=a23dba878ab0d711d6baebe61b361f7c330b0de6746487f1a9a26c4ed87536eb (Link con dibujos de los distintos elementos del tablero e información de la unidades).

Aquellos que quiera mejorar su programación en C++ pueden consultar:

5. Effective C++ Third Edition 55 Specific Ways to Improve Your Programs & Designs. Scott Meyers. Addison Wesley Professional. 12/05/2005. ISBN: 0-321-33487-6.
6. Design Patterns: Elements of Reusable Object-Oriented Software. Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides. Addison-Wesley Professional. 10/11/1994. ISBN: 0-201-63361-2.
7. The C++ Standard Library: A Tutorial and Reference. Nicolai M. Josiuttis. Addison Wesley. 06/08/1999. ISBN: 0-201-37926-0.
8. The Art of C++, Herbert Schildt. McGraw-Hill. 12/04/2004. ISBN: 0-072-25512-9.
9. C++: The Complete Reference, 4th Edition, Herbert Schildt. McGraw-Hil, 19/11/2002. ISBN: 0-072-22680-3.