MANUAL mini-Simulador SIMAUTAVA con rejilla Hexagonal (mSS-HEXA) [versión H.1.5]

ÍNDICE

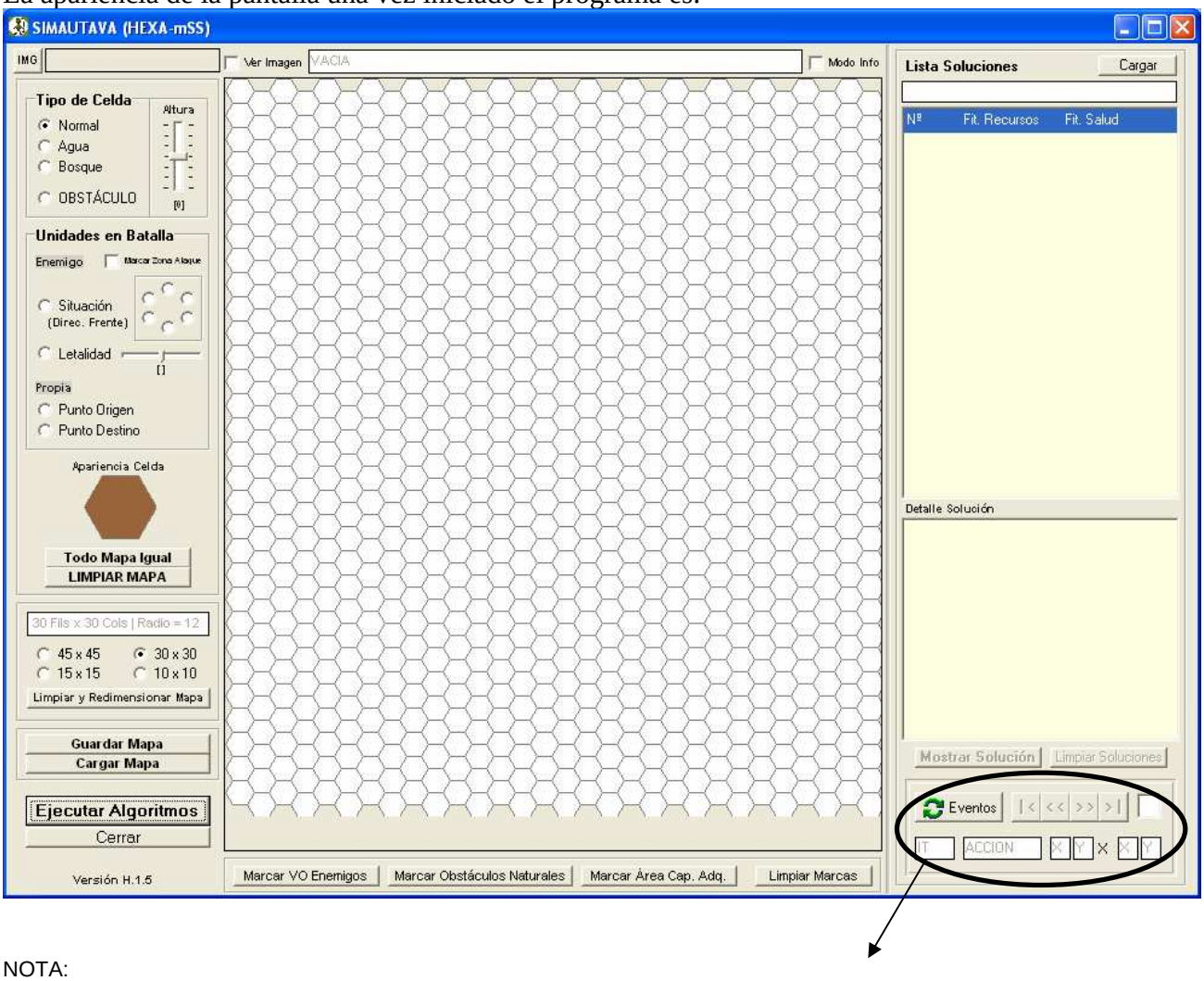
1. CARACTERÍSTICAS GENERALES

1.1. Introducción

En primer lugar habría que aclarar que este programa es considerado un simulador en cuanto a la funcionalidad que ofrece y a las reglas que define para la resolución de los problemas, pero no sirve para hacer simulaciones interactivas en tiempo real. De hecho fue creado como herramienta 'auxiliar' para el desarrollo y estudio de algoritmos.

Por esa razón, sus posibilidades pasan por la *creación y edición de mapas*, que serían definiciones de escenarios que modelan un problema (búsqueda de camino óptimo en un campo de batalla, por parte de una unidad). Así como la *ejecución de varios algoritmos* para resolver dicho escenario, los cuales han sido desarrollados dentro del simulador. Por último permite la *visualización y estudio* de las soluciones obtenidas de forma intuitiva.

La apariencia de la pantalla una vez iniciado el programa es:



NOTA:
La funcionalidad de la zona de Eventos, está en desarrollo, para poder incorporar cierto dinamismo a la simulación, pero todavía no es operativa.

En la que se puede ver en la parte izquierda las posibilidades de edición de mapas, en el centro el mapa con el que se está trabajando y en la parte derecha, las opciones en relación a la visualización y estudio de las soluciones obtenidas (por el algoritmo elegido sobre el mapa en cuestión).

Los escenarios, como se puede observar, consisten en una retícula de celdas hexagonales de distintos tipos que definen un terreno (el campo de batalla). Sobre él se situarán los enemigos (si hay), se marcarán las zonas batidas por dichos enemigos (o simplemente dañinas para nuestra unidad) y, por último, se marcarán el punto origen del camino (situación actual de nuestra unidad) y el punto destino (punto que se desea alcanzar). Una vez definido un escenario, que será un problema del tipo que nos ocupa (búsqueda de camino óptimo), lo podremos resolver con los algoritmos implementados y visualizar las soluciones de forma gráfica e intuitiva.

Como apuntes decir que se va a considerar, como ya se ha comentado, una Compañía (CIA) como unidad del problema y como enemigos se considerarán Secciones (3 veces más pequeñas) y que el tamaño de las celdas equivaldría a unos 500x500 metros en la realidad.
La unidad dispondrá de una *salud (energía)*, correspondiente a los efectivos de que dispone y estado de los vehículos de la misma, y de unos *recursos*, correspondientes a combustible y otros consumibles. Ambas cantidades irán mermando a lo largo del camino en base a las penalizaciones que tengan asociadas las celdas según su tipo, diferencia de alturas o letalidad.

El programa ha sido desarrollado en Borland Delphi 7 y en principio será solo compatible con Windows.

1.2. Restricciones

Como ***restricciones*** simplemente hay que considerar:
- tanto la CIA, como cada unidad enemiga ocuparán exactamente una celda, ya que el tamaño de dicha celda equivale al tamaño del despliegue de una CIA en formación de ataque (unidad del problema) y de una Sección en formación defensiva (enemigos).
- una celda solo podrá ser de un tipo (y de un subtipo opcionalmente).
- podrá haber ninguno, uno o varios enemigos en el escenario.
- para definir completamente el problema es necesario indicar un punto origen y un punto destino, (solo uno de cada tipo).
- el camino solución no podrá atravesar celdas ocupadas por un enemigo ni obstáculos (naturales o artificiales).

1.3. Terreno (Celdas)

El terreno ha sido modelado, como ya se ha visto, como una retícula de celdas hexagonales, dónde cada **Celda:**
- Tiene unas *coordenadas X,Y*, estando el origen en la esquina superior izquierda y creciendo la X hacia la derecha y la Y hacia abajo.
- Es de un *tipo*:
  - Normal (Tierra) → terreno firme por el que circular sin problemas.

  - Agua → terreno pantanoso o con agua como ríos, lagos, etc.
  - Bosque → terreno con abundante y alta vegetación que dificulten el avance por él e incluso oculten a la unidad.
  - Obstáculo → terreno infranqueable para una CIA creado artificialmente (puede ser un agujero profundo, un muro, un campo de minas, etc).
- Tiene una *altura* entre -3 y 3, siendo 0 el nivel normal, positivo para elevaciones y negativo para depresiones.
- Tiene asociado un *daño por defecto* a la unidad (consumo de energía), justificado como bajas de no combate y averías de vehículos u otros aparatos.
- Puede tener una *letalidad* asociada (daño por impacto de armas enemigas o por otras causas).
- Tiene asociado un *consumo de recursos por defecto* a la unidad, que indica el combustible o cansancio que produce atravesar esa celda. Es un indicativo de la dificultad que conlleva cruzarla.

1.4. Subtipos de Celdas

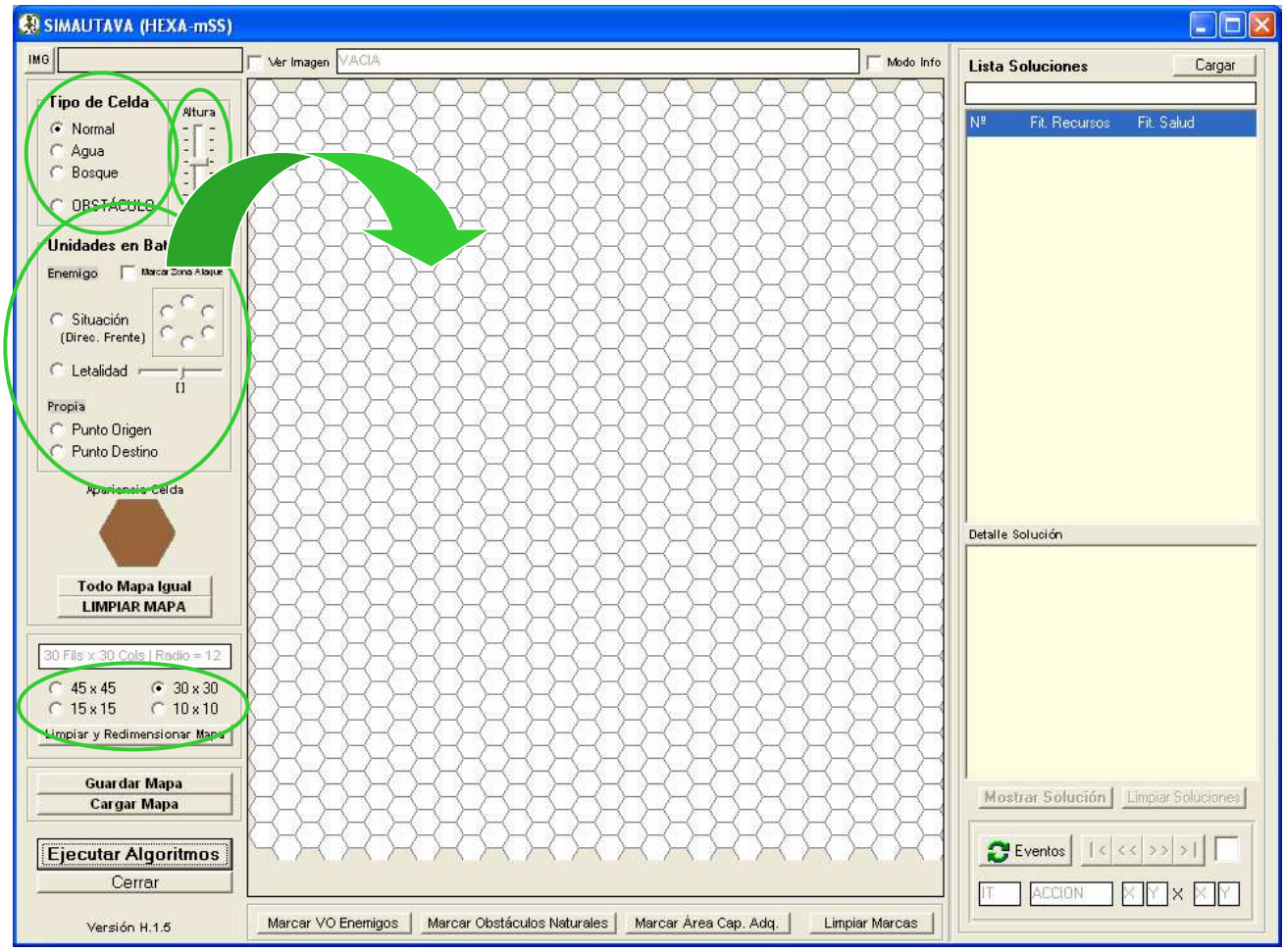
A la hora de definir completamente un escenario para el problema, es necesario indicar los puntos de origen y destino de la unidad, así como la situación de los enemigos (si los hay) y de las celdas afectadas por el impacto de sus armas. Para ello se consideran una serie de Subtipos de Celda**:**

- Enemigo → celda en la que está situada una unidad enemiga.
- Punto Origen → celda de la que parte la unidad (punto inicial del problema).
- Punto Destino → celda a la que quiere llegar la unidad (punto final del problema).
- Letalidad → se considera un subtipo y asocia a las celdas un daño (consumo de energía) por encontrarse batidas por la acción de algún arma del enemigo (o porque las queramos considerar dañinas para los efectivos de la unidad). El valor asociado a la celda estará entre 0 y 100, siendo 100 el valor más dañino.

NOTA:
La letalidad por daño de las armas de un enemigo se obtiene como conjunción de tres factores, la probabilidad de que el enemigo abra fuego si la unidad se sitúa en la celda, la probabilidad de que el fuego alcance a la unidad y el daño que produciría de impactar sobre ella.

2. EDICIÓN

Los pasos para crear o editar un mapa son muy sencillos, pues basta con elegir el tamaño del mismo, posteriormente el tipo y subtipo de la celda a dibujar y hacer clic sobre la retícula del mapa en el cuadro que queramos que ocupe dicha celda. También es posible dibujar haciendo clic y manteniendo pulsado el botón, arrastrar el ratón para que se vayan dibujando celdas de dicho tipo (siempre que sea posible), al igual que se hace en cualquier programa de dibujo.



Las celdas por defecto no tendrán tipo, subtipo ni altura asociadas, por lo que será necesario definir todas las que hay en la retícula.
Cuando se dibuja una celda, la que hubiera es esa posición se sobrescribe.
A continuación se detallarán los posibles tipos y subtipos de celdas, así como todas las funciones asociadas a la edición que se pueden usar.

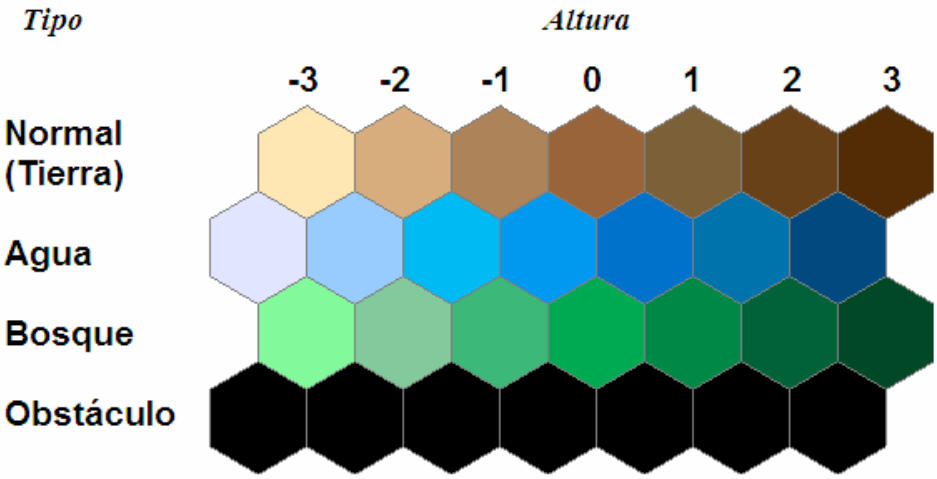
2.1. Tipos y Subtipos de Celdas

Los tipos y subtipos que se consideran están comentados en los apartados 1.3 y 1.4. Ambos son considerados a la hora de definir completamente un escenario de problema o mapa.

Los **Tipos** se pueden seleccionar en la parte superior izquierda de la pantalla de mSS-Hexa (con cada uno de los radiobuttons) y como se puede ver, hay 4, cada uno representado con un color:

- Normal (Tierra) → Color Marrón.
- Agua → Color Azul.
- Bosque → Color Verde.
- Obstáculo → Color Negro.

Del mismo modo, a cada celda se le puede asignar una **altura** dentro del rango [-3,3] como se comentó usando el slide (la barra) que hay junto al tipo. De manera que a mayor altura, más oscuro será el color del terreno y a mayor profundidad, más claro (manteniendo el mismo color del terreno), siguiendo el esquema de los extendidos tonos hipsométricos (si bien en este caso se seguirá el mismo patrón para las celdas de tipo agua, a diferencia de lo que ocurre en dichos tonos).
Por tanto, los colores asociados a tipos y alturas serían:



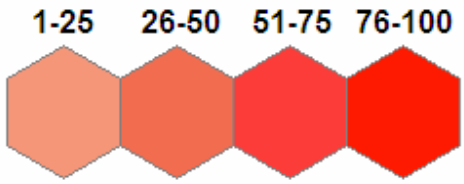
En lo que respecta a los **Subtipos**, también se pueden seleccionar en la parte izquierda, dentro del marco *Unidades en Batalla*. Estos subtipos se representarán con bordes y formas de la celda

- Enemigo → Celda con color del tipo de terreno y altura correspondiente, con borde muy grueso rojo. Para los enemigos es posible dibujar solo su situación o su situación y su zona de ataque usando plantillas (como se explica en el apartado 2.3).
- Punto Origen → Celda con color del tipo de terreno y altura correspondiente, con borde muy grueso negro. Se usará el radiobutton *Punto Origen* de la sección *Unidad Propia*.
- Punto Destino → Celda con color del tipo de terreno y altura correspondiente, con borde muy grueso amarillo. Se usará el radiobutton *Punto Destino* de la sección *Unidad Propia*.

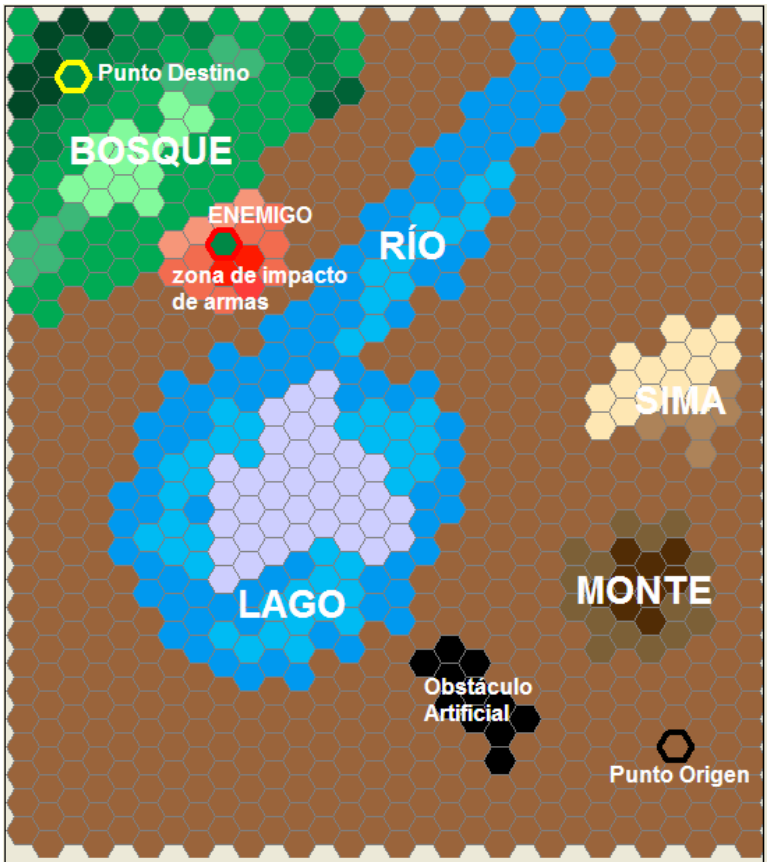
Celdas Especiales

– Letalidad → Se asignará usando el slide (barra de desplazamiento) asociado. Las celdas se marcarán con distintos tonos de rojo en función del daño asociado (entre 0 y 100 como ya se indicó), según indica la siguiente figura:

Letalidad



Un ejemplo de mapa ya definido (en el que se han sobrescrito etiquetas para aclarar) podría ser:

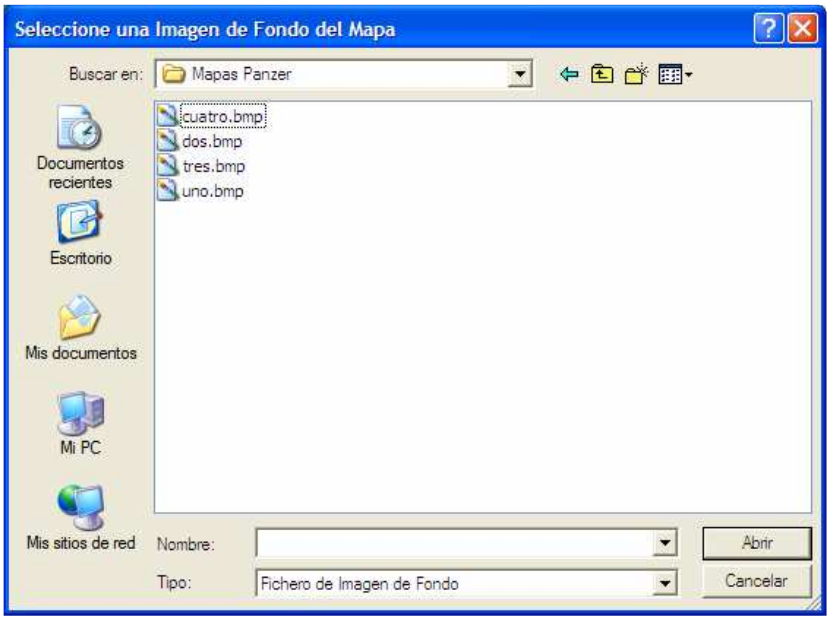


2.2. Funciones de Edición

Las funciones de edición se ofrecen al usuario mediante botones situados en la parte izquierda y en la parte inferior, dentro del marco de definición de tamaño del mapa.

Mapas con Imágenes Reales

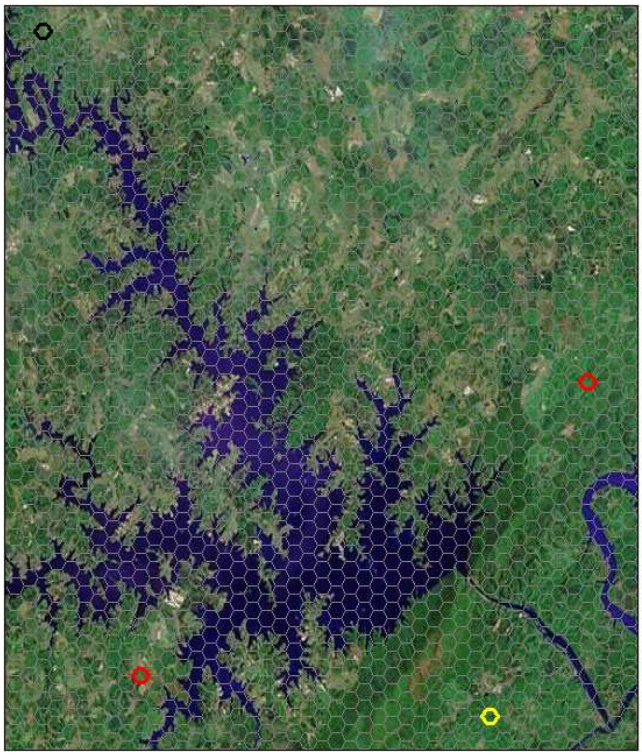
En esta aplicación es posible crear mapa a partir de imágenes reales, de manera que la definición de las celdas (tipos y subtipos), se podrá hacer para modelar dicho mapa real, editando directamente sobre la imagen del mismo (y guardándose los datos 'debajo') o alternando entre uno y otro con el checkbox *Ver Imagen*.
Para hacer esto, antes será necesario cargar la imagen usando el botón *IMG*, tras lo que aparecerá la ventana:

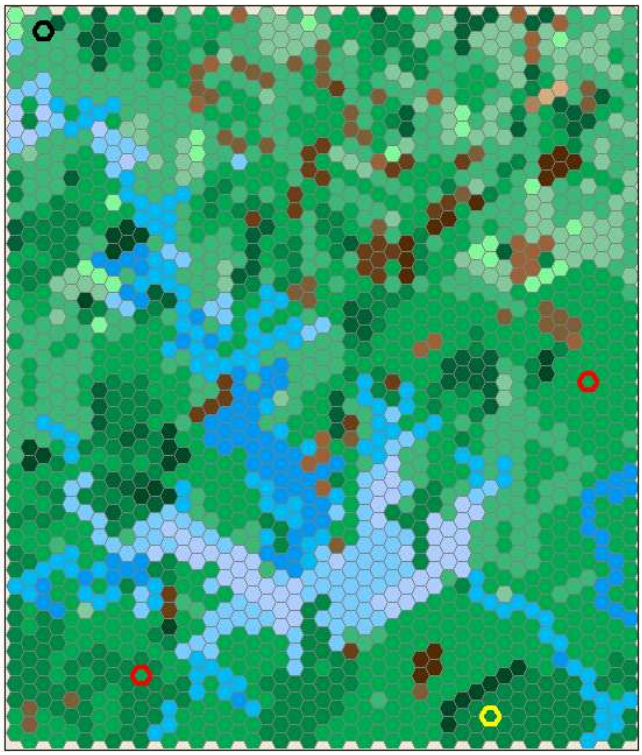


Con la que podremos seleccionar cualquier fichero de mapa de bits, '**.bmp**' cuyo tamaño óptimo a visualizar debería ser 545x640 píxeles.

Una vez cargado el mapa podremos 'dibujar' directamente sobre el mismo, eligiendo previamente los tipos y/o subtipos deseados para modelarlo de manera fiel. Las celdas irán cambiando en la capa subyacente.

De esta forma podremos modelar cualquier escenario real, por ejemplo:





Limpiar y Redimensionar Mapa

Permite determinar el tamaño que tendrá el mapa que queremos modelar en número de filas x columnas de celdas. Cuando se determine este tamaño y se pulse el botón, el mapa que haya cargado actualmente será borrado y redimensionado al tamaño elegido.

Apariencia de Celda

Se trata de un campo informativo que muestra el aspecto de la celda que hemos seleccionado (con su tipo, subtipo y altura) cuando se dibuje en el mapa. Por defecto será normal con altura 0.

Todo Mapa Igual

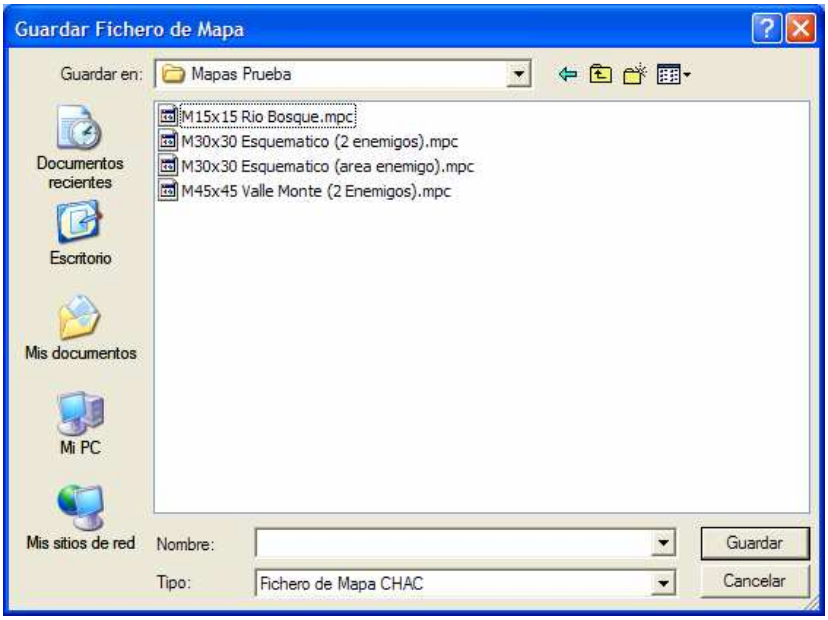
Copia en todas las celdas del mapa la que esté seleccionada por defecto (la que se muestra en *Apariencia de Celda*), con excepción de algunos tipos y subtipos.
Esta utilidad facilita mucho la creación de mapas grandes en los que predomine algún tipo (cosa muy común, por otra parte).

Limpiar Mapa

Elimina todas las celdas del mapa y las deja todas vacías (sin tipo, subtipo ni altura), como al inicio. No redimensiona el mapa.

Guardar Mapa

Permite guardar el mapa que estemos definiendo (o que hayamos definido). Tras pulsarlo aparecerá una ventana:

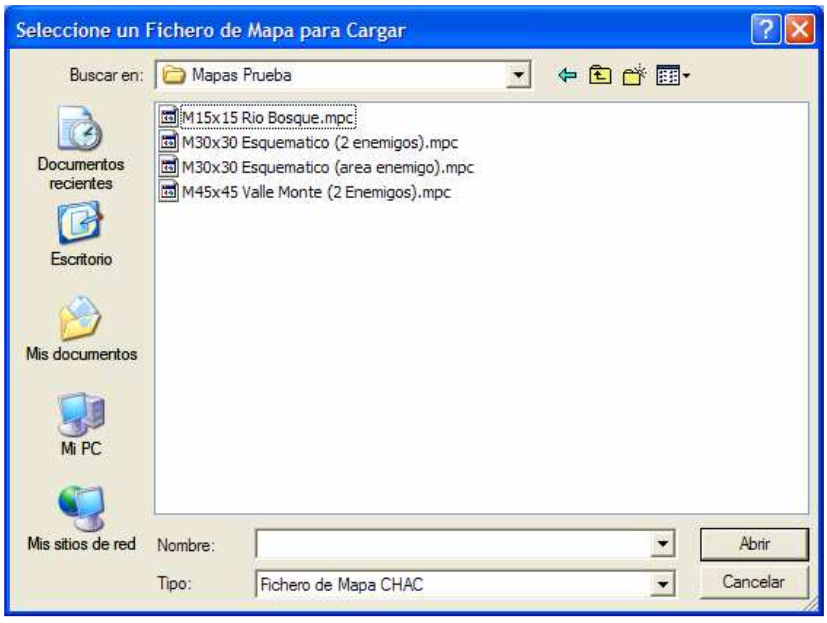


En la que podremos elegir el directorio y nombre de fichero a guardar. La extensión será **'.mpc'**. Un mensaje nos confirmará si se ha guardado correctamente.

NOTA:
En esta versión se guardará también un fichero de eventos (de extensión '.eve'), el cual actualmente no se utiliza y se considerará en versiones futuras para incorporar dinamismo en el simulador.

Cargar Mapa

Permite cargar un mapa que estuviésemos definiendo (o ya definido). Al cargarlo se perderán los cambios hechos sobre el actual. Nuevamente aparecerá una ventana:



En la que podremos seleccionar de entre los que tengamos guardados. La extensión será **'.mpc'**. Un mensaje nos confirmará si se ha cargado correctamente.

Modo Info

En la parte superior de la pantalla se mostrará toda la información de las celdas sobre las que pasemos con el ratón. Si se pulsa el checkbox *Modo Info*, no se podrá modificar el mapa (el cursor del ratón se convertirá en una interrogación).
Tras cargar un mapa, este modo se activa por defecto para evitar cambios no deseados sobre el mismo.
La información que se muestra es:

[X,Y] tipo celda , altura , [Subtipo (letalidad si tiene)] | [daño por defecto – recursos por defecto]

donde *X* es la columna e *Y* es la fila, la altura se muestra como *Alt*, la letalidad viene descrita como *Impacto Arma (valor daño)*, el daño por defecto es *d0* y los recursos consumidos por defecto son *r0*.

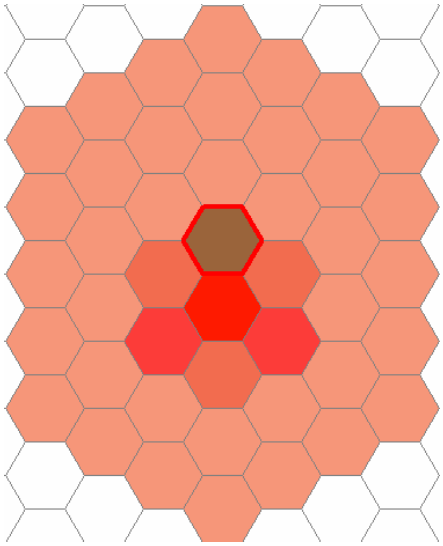
2.3. Situación de unidades Enemigas. Plantillas de Zona de Impacto

A la hora de fijar la posición de un enemigo en el mapa, el programa ofrece una utilidad, que es el dibujado automático de la zona de impacto de las armas del mismo. Para ello se permite al usuario la creación y carga de plantillas de estas zonas.
Estas plantillas permiten definir de manera gráfica y sencilla la letalidad de cada una de las celdas en la zona circundante a una unidad enemiga (en base a sus armas y al alcance de las mismas).

Crear Plantilla

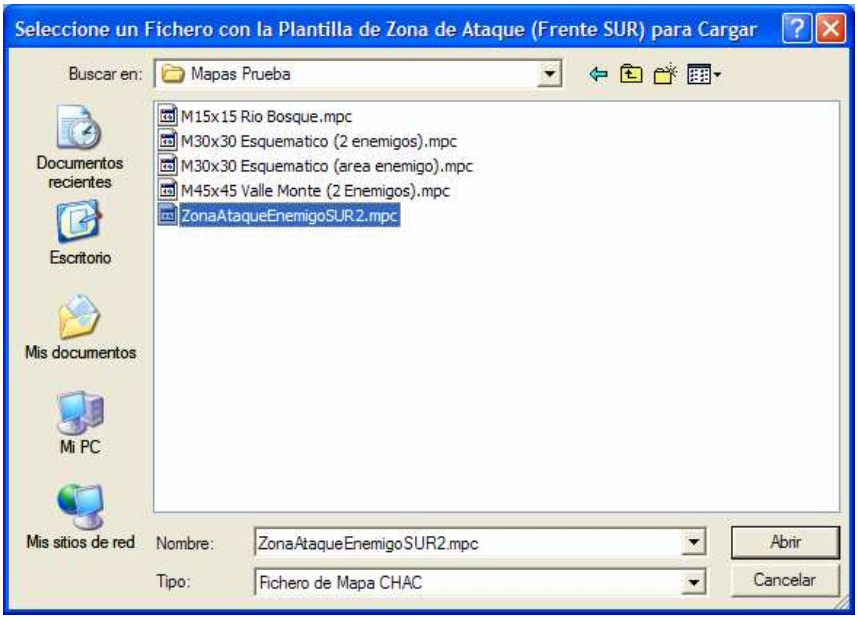
Para crear una plantilla, se siguen los mismos pasos que para crear cualquier otro mapa. Se suele usar un mapa de 10x10 porque la zona de impacto no suele ser demasiado amplia, pero se puede usar cualquier tamaño.
Sobre el mapa (con las celdas vacías si se quiere) situaremos la unidad enemiga e iremos asignando y dibujando la letalidad de las celdas alrededor del enemigo (pueden ser de cualquier tipo).
La orientación del enemigo se supondrá que es con el frente hacia el SUR.

Una vez hayamos definido su zona de impacto, guardaremos el mapa como si fuese uno más (con el botón de Guardar Mapa y con extensión ‘.mpc’).
Un ejemplo de plantilla (la que se suele utilizar) podría ser:



Usar Plantilla

Para usar una plantilla existente, bastará con pulsar sobre el checkbox *Marcar Zona Ataque* que hay en la sección de datos de *Enemigo* del marco *Unidades en Batalla*. Tras pulsarlo, aparecerá una ventana que nos pedirá elegir la plantilla a cargar:



Elegiremos la plantilla a cargar y la abriremos. Un mensaje nos confirmará si se ha cargado correctamente.

Una vez cargada, marcaremos el radiobutton *Situación* y elegiremos la dirección del frente de la unidad con uno de los radiobuttons de orientación (NORTE, NORESTE, NOROESTE, SUR, SURESTE, o SUROESTE), después bastará con seleccionar sobre el mapa la posición de la unidad enemiga y el resto se dibujará automáticamente siguiendo la plantilla. Solo se dibujará dentro de los límites del mapa y se evitarán los obstáculos (no se dibujará sobre ellos).
Al dibujar *solo se considerará la letalidad de las celdas de la plantilla*, por lo que el tipo de las mismas no se verá afectado.
La dirección por defecto será SUR, la cual copiará la plantilla de forma idéntica a como se haya definido.

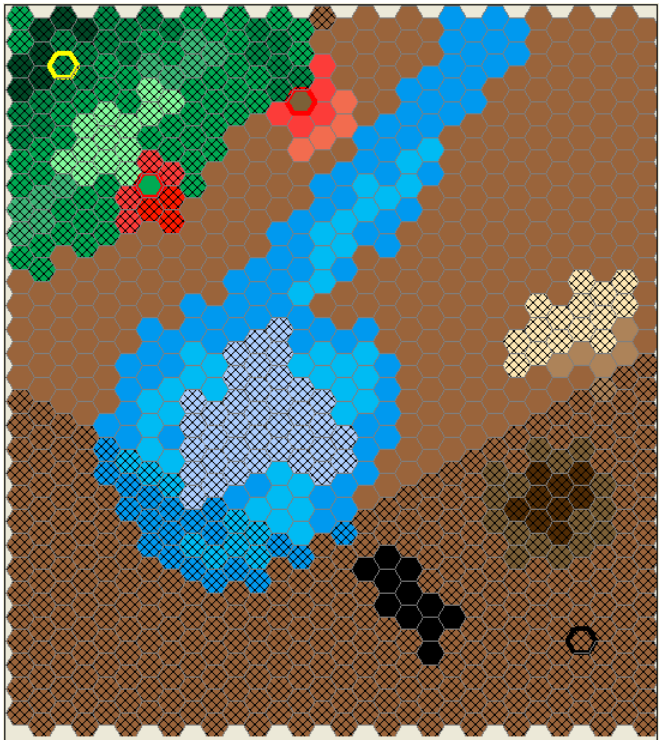
NOTA:
En esta versión solo están activas las orientaciones SUR y NORTE, siendo la primera idéntica a la plantilla que creemos y la segunda mostrando la plantilla invertida según el eje horizontal. En futuras versiones se ofrecerán todas las posibilidades.

3. HERRAMIENTAS DE VISUALIZACIÓN

En esta versión del mini-simulador, se han incluido 3 herramientas que facilitarán la visualización de varias de las restricciones que se tienen en cuenta en la resolución del problema, en concreto, se ofrecen las posibilidades de marcar las celdas vistas y ocultas para los enemigos, marcar las celdas que determinan las capacidades de adquisición y marcar los obstáculos naturales para la unidad (ver sección 6 para ver las definiciones de estos conceptos). Estas posibilidades se ofrecen como botones en la parte inferior de la pantalla principal (bajo el mapa), junto con el botón *Limpiar Marcas*, que elimina las celdas marcadas con cada una de esas opciones.

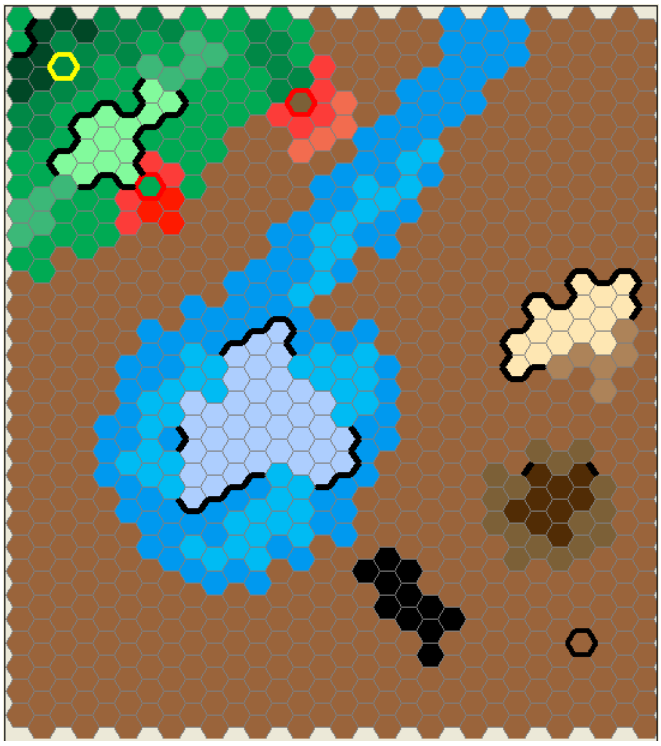
3.1. Marcar Celdas Vistas y Ocultas

Se usa el botón *Marcar VO Enemigos* y tras pulsarlo, se señalarán mediante un sombreado aquellas celdas que sean ocultas a todos los enemigos del mapa, bien por estar fuera de su capacidad de adquisición o por tener celdas que obstruyan la línea de visión (ver sección 6) de los enemigos.
Aquí podemos ver un ejemplo:



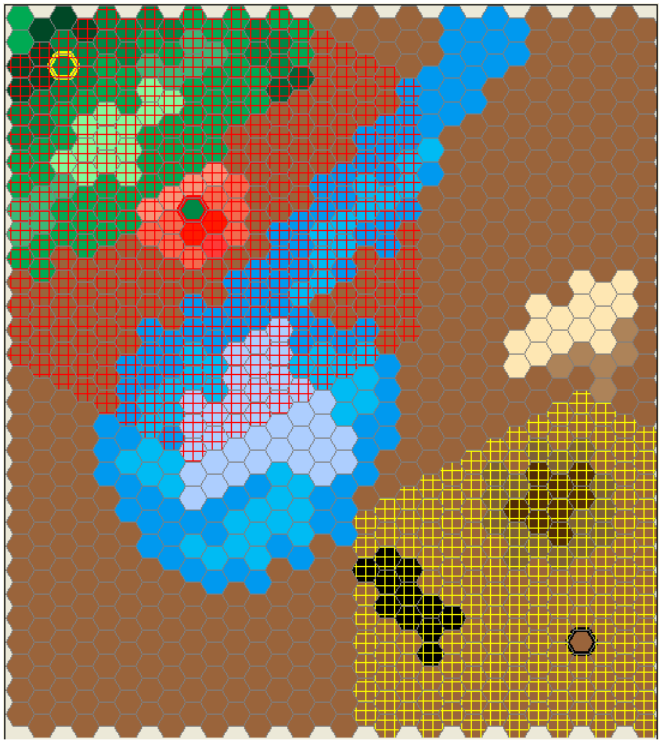
3.2. Marcar Obstáculos Naturales

Se usa el botón *Marcar Obstáculos Naturales* y tras pulsarlo, se señalarán mediante una línea negra las fronteras entre celdas que la unidad no puede atravesar por haber una diferencia de altura entre ellas mayor de la que la unidad puede superar (por defecto 2).
Un ejemplo podría ser:



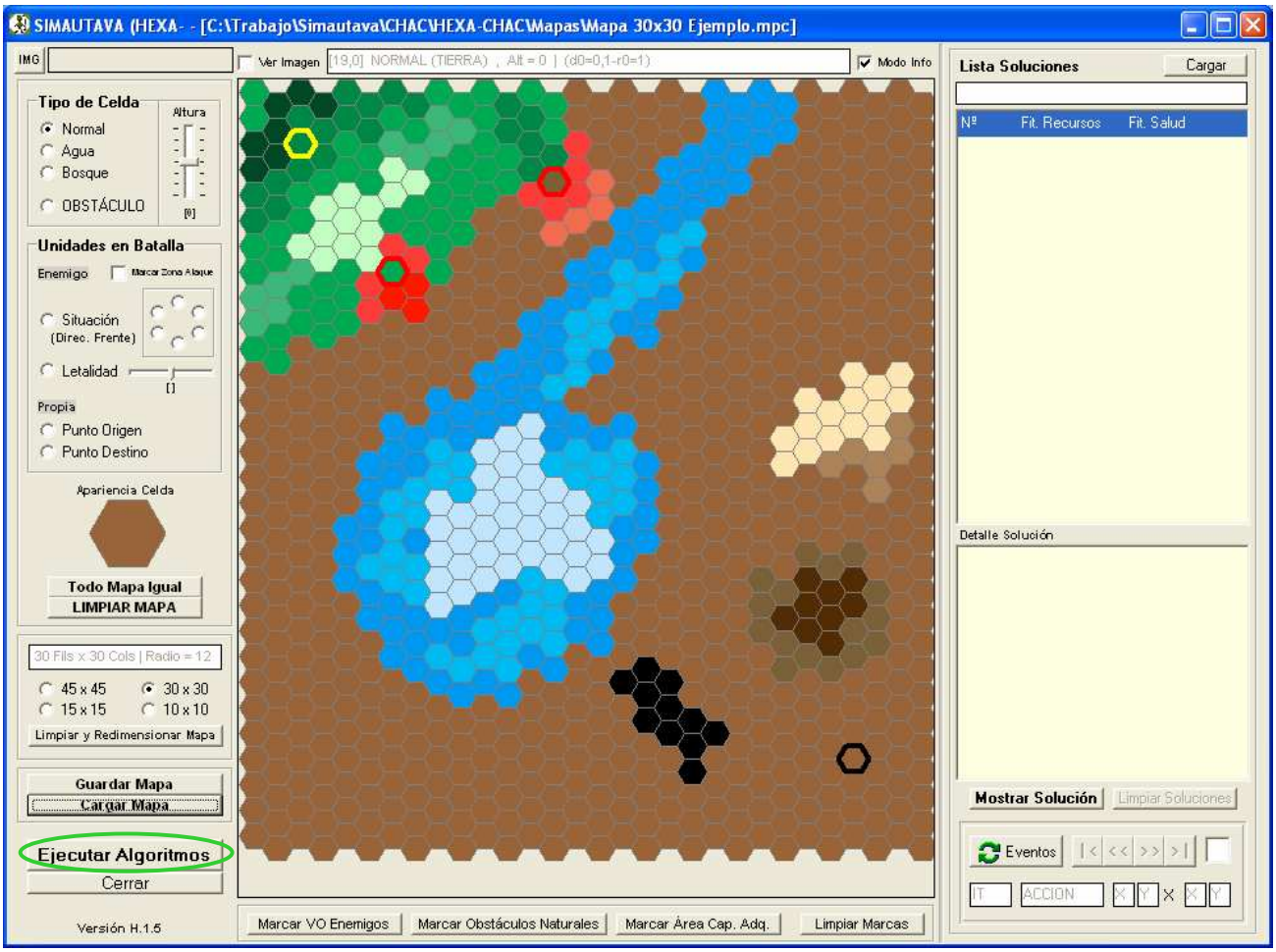
3.3. Marcar Área de Capacidad de Adquisición

Se usa el botón *Marcar Área Cap. Adq.* y tras pulsarlo, se señalarán con tramas de color las celdas que están incluidas en el área de capacidad de adquisición. La de los enemigos se marcará con trama roja y la de la unidad con trama amarilla. Aquí se puede ver un ejemplo en el que se considera 10 celdas como radio del área de enemigos y unidad del problema.



4. EJECUCIÓN DE ALGORITMOS

Antes de ejecutar un algoritmo para resolver el escenario, **debe haber cargado un mapa y deben estar guardados todos los cambios que se le hayan hecho. Es obligatorio que haya definido un punto de origen y un punto de destino que determinen el problema a resolver**.
Una vez esté correcto, se pulsará el botón *Ejecutar Algoritmos*, con lo que aparecerá la ventana de parámetros que se describe en el apartado 4.2.



4.1. Algoritmos Implementados

Actualmente mSS-Hexa incluye 4 algoritmos para ejecutar sobre cualquier mapa, varios de ellos configurables de varias formas, con lo que operarán de muy diferentes maneras, actuando realmente como muchos más algoritmos y dando más posibilidades de elección al usuario.

CHAC (Hexa-CHAC)

Se trata de un algoritmo original multiobjetivo que se ha desarrollado para resolver este problema y del que se han hecho varias publicaciones tanto en congresos nacionales, como internacionales.

Como apunte decir que al ser un algoritmo multiobjetivo, muchas veces se obtendrá más de una solución, siendo igual de buenas todas considerando los dos criterios del problema (rapidez y seguridad).
Se trata de un algoritmo de optimización basada en colonias de hormigas que puede usar dos reglas de transición diferentes y sus hibridaciones con un algoritmo de búsqueda local.
Ofrece soluciones con el mejor compromiso entre rapidez y seguridad, siguiendo de manera bastante fiel las decisiones del usuario.

CHAC-4

Implementación del algoritmo CHAC para la consideración de 4 objetivos, obtenidos como subdivisión de los dos criterios principales en otros dos. De forma que la rapidez se divide en la minimización de recursos consumidos y en la reducción de la distancia media al destino. La seguridad por su parte, se divide en la minimización de salud consumida y en la minimización de la visibilidad de la unidad por parte de los enemigos.
Al igual que en el caso de CHAC, se tienen dos reglas de transición diferentes y se puede determinar la prioridad de los objetivos.

GreedyMO

En este caso se trata de un algoritmo greedy clásico, pero adaptado para trabajar con este problema, por lo que también es multiobjetivo, puede devolver más de una solución y emplea mecanismos de backtracking para obtener soluciones en todos los casos.
No se ve afectado por la prioridad de los objetivos.

monoCHAC

Se trata de una implementación del algoritmo CHAC como un monoobjetivo clásico, agrupando los términos relativos a los dos objetivos en una única expresión con sus correspondientes ponderaciones.
Al ser monoobjetivo, solo ofrece una solución que intentará satisfacer a la vez los dos criterios (rapidez y seguridad), sin que se pueda configurar la prioridad de cada uno de ellos.

MOACS

Se trata de una implementación del algoritmo multiobjetivo propuesto por Barán et al. para resolver otro tipo de problema (Enrutamiento de Vehículos con Ventana de Tiempo), adaptado a este problema.
Es también configurable y obtiene buenas soluciones, pero CHAC obtiene mejores y simplemente se ofrece como otra alternativa para comparar.

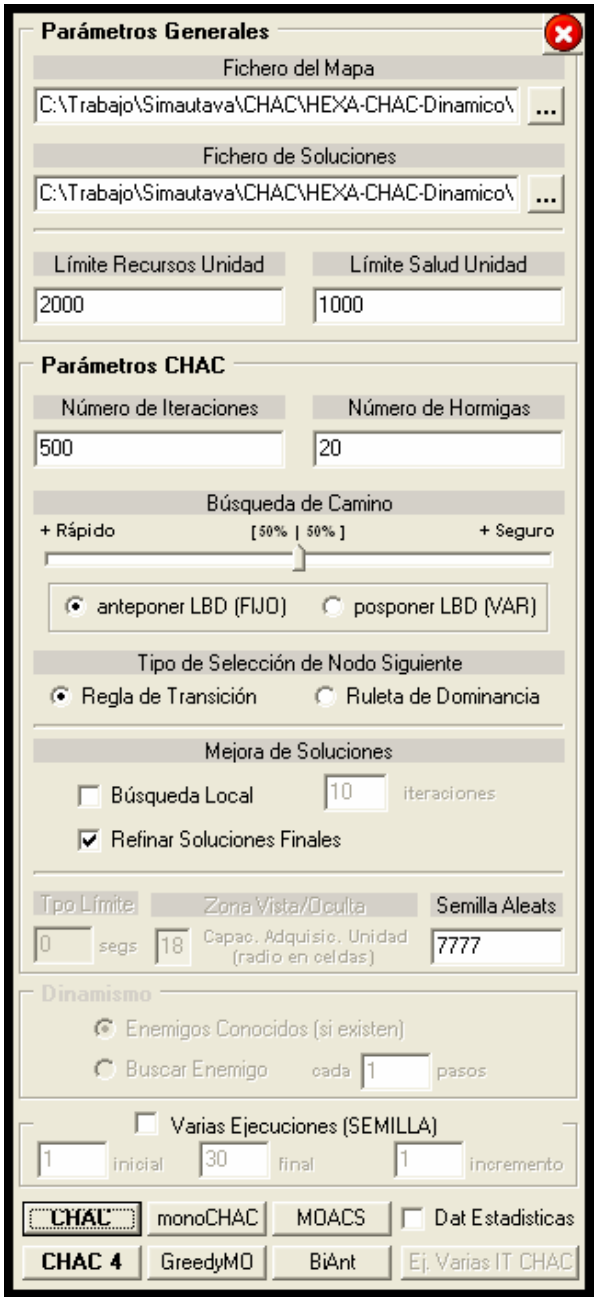
BiAnt

Adaptación del algoritmo multiobjetivo propuesto por Iredi et al. para resolver el SMTTP, para la resolución de este problema.
Es también configurable y obtiene buenas soluciones si se le dan iteraciones suficientes, pero las de CHAC son mejores, es otra alternativa de comparación.

→ Para extender estos conceptos se puede leer la tesis desarrollada sobre ellos (Tesis Antonio Mora) o los artículos sobre CHAC incluidos en el Apéndice 1.

4.2. Ventana de Parámetros

Esta ventana nos permite configurar casi totalmente los algoritmos a ejecutar para resolver el problema. Tiene el aspecto:



Como se puede ver hay dos tipos de parámetros: *Generales*, que son los que influyen en todos los algoritmos y *parámetros de CHAC*, que afectan a CHAC en sus tres implementaciones y algunos al GreedyMO, MOACS y BiAnt.

Una vez establecidos los parámetros, se pulsará el botón correspondiente al algoritmo deseado y tras un tiempo (depende de los parámetros, pero irá desde algunos segundos a pocos minutos), se mostrará un mensaje con el número de soluciones encontradas.

NOTA:
En muchas ocasiones no aparece el icono con el reloj de arena indicando procesamiento, este fallo no afecta al procesamiento, por lo que tras varios segundos aparecerá el mensaje de fin de ejecución.

A continuación se comentarán los parámetros.

Parámetros Generales

- Fichero del Mapa → se trata del fichero que contiene la definición del mapa en formato *'.mpc'*. Este campo se rellena automáticamente con la ruta del mapa que se tiene cargado actualmente en mSS-Hexa.
- Fichero de Soluciones → es el fichero en el que se guardarán las soluciones que encuentren los algoritmos en un formato propio de mSS-Hexa también. Por defecto tiene como nombre y ruta la misma del mapa, pero con extensión *'.sol'*. Este fichero podrá ser abierto posteriormente.
- Salud Unidad → puntos de salud (energía) de los que dispone la unidad para recorrer el camino desde el origen al destino. Debe ser un número entero. Por defecto tiene un valor de 1000.
- Recursos Unidad → puntos de recursos de los que dispone la unidad para recorrer el camino desde el origen al destino. Debe ser un número entero. Por defecto tiene un valor de 2000.

Parámetros de CHAC

Estos parámetros afectarán a dicho algoritmo (y algunos a MOACS). Son:

- *Iteraciones* → número de iteraciones que se desean hacer para resolver el mapa. A mayor número, mejor serán las soluciones obtenidas, pero más tiempo consumirá su cálculo. Es aconsejable unas 1000 iteraciones para mapas de 30x30 (aunque depende mucho de su dificultad). Se considera en todos los algoritmos.
- *Número de Hormigas* → se podría entender como el número de agentes que buscarán la solución. A mayor número mejores soluciones se obtendrán y más tiempo emplearán. Es aconsejable unas 30-50 hormigas para mapas de 30x30. También se considera en MOACS y BiAnt.
- ***Barra de Criterio*** → se trata del parámetro *más importante* del algoritmo, ya que es el que determina el criterio a seguir para buscar las soluciones. Puede ser buscar el camino más rápido (el que consuma menos recursos) o el más seguro (el que garantice mayor seguridad para los efectivos). Siempre se considerarán los dos criterios (si se asigna algún porcentaje a cada uno), pero el usuario decidirá la proporción entre ambos. También se considera en MOACS y BiAnt, y no en GreedyMO.
- *Tipo de Selección de Nodo Siguiente* → se trata de una regla que usarán las hormigas para elegir el siguiente nodo durante la construcción de un camino que sea posible solución. Podrá ser la clásica *Regla de Transición* (adaptada a nuestro problema y algoritmo) o la llamada *Ruleta de Dominancia*. La primera dará mejores resultados normalmente, pero la segunda hace más exploración, por lo que puede ser más útil en mapas complejos. Si bien la segunda necesitará de más iteraciones y hormigas para obtener buenos resultados. Solo lo utilizan CHAC y CHAC-4.
- *Mejora de Soluciones* → con estos parámetros se podrán mejorar las soluciones encontradas por el algoritmo normal. Con la *Búsqueda Local*, se intentará mejorar cada solución de cada hormiga en cada iteración, pero a costa de perder cierta funcionalidad del algoritmo, lo que podría llevar a obtener soluciones no tan buenas en algunas ocasiones. Si se desea hacer, se

debe indicar el número de iteraciones de la misma (número de pruebas de mejora), teniendo en cuenta que a más iteraciones, más pruebas se harán sobre las soluciones, pero también más se alejará del funcionamiento usual del algoritmo. Además, esta búsqueda conlleva un mayor tiempo de ejecución (aunque no demasiado porque se suelen hacer pocas iteraciones).
Por otro lado es posible hacer un *Refinamiento de las Soluciones Finales*, en el cual se eliminen posibles celdas sobrantes (celdas que están en contacto con dos celdas vecinas en la solución). Esta opción mejora las soluciones siempre que se pueda.

- *Tiempo Límite* → esta restricción no se considera necesaria actualmente, puesto que los algoritmos no tardan demasiado en su ejecución (pocos minutos en el caso de los mapas más grandes con muchas ejecuciones).
- *Zona Vista/Oculta* → este parámetro solo se usa cuando se quiere resolver un mapa *sin enemigos* y permite definir un radio de celdas en el que se comprobará si dicha celda es visible o no durante la construcción del camino, ya que no se conoce la situación del enemigo en el mapa. Vendría a representar una medida de la cautela que se tendrá. Un mayor radio aumentará el tiempo de ejecución. El radio que se considerará es el correspondiente a la capacidad de adquisición de la unidad del problema.
- *Semilla Aleatorios* → se trata de un número que servirá como base para el cálculo de los números aleatorios de los algoritmos (son algoritmos estocásticos). Se usa para poder replicar los resultados, ya que usando la misma semilla siempre se obtendrán las mismas soluciones. Con diferentes semillas, las soluciones serán distintas, pudiendo mejorar o empeorar.

Dinamismo

Se trata de una característica aún no implementada, pero en desarrollo actualmente.

Varias Ejecuciones

Este apartado se utilizará para realizar varias ejecuciones del algoritmo elegido sobre el mapa (problema) en cuestión. Para ello se determinarán varias semillas diferentes, una por ejecución, para lo que se seleccionará el número inicial y final del rango de semillas, así como el incremento.

5. VISUALIZACIÓN DE SOLUCIONES

Una vez haya terminado la ejecución del algoritmos elegido y si ha obtenido soluciones, tras pulsar el botón *Cerrar* de la ventana de parámetros se cargará en la parte derecha de la pantalla principal de mSS-Hexa la lista de soluciones encontradas (las encontradas por la última ejecución en caso de hacer varias de forma automática).

5.1. Ficheros Obtenidos

Tras la ejecución del algoritmo elegido, ya sea una o varias, se habrán creado varios ficheros, un fichero '**.sol**' por cada una de las ejecuciones, el cual incluirá en su nombre los parámetros con los que se ejecutó el algoritmo. De forma que su nombre tendrá el formato:

'<nombre mapa> [<algoritmo> [, bl], semilla, nº iteraciones, nº hormigas, <criterio>].sol'

'bl' se incluirá si se realiza una búsqueda local en el algoritmo. El criterio será '+r' si se buscó el camino más rápido (hasta el 50%) y '+s' en el caso de haber buscado el camino más seguro.
En el caso de los algoritmos CHAC o CHAC-4, también se incluirá el tipo de regla de selección 'RT' (regla de transición) o 'RD' (ruleta de dominancia).

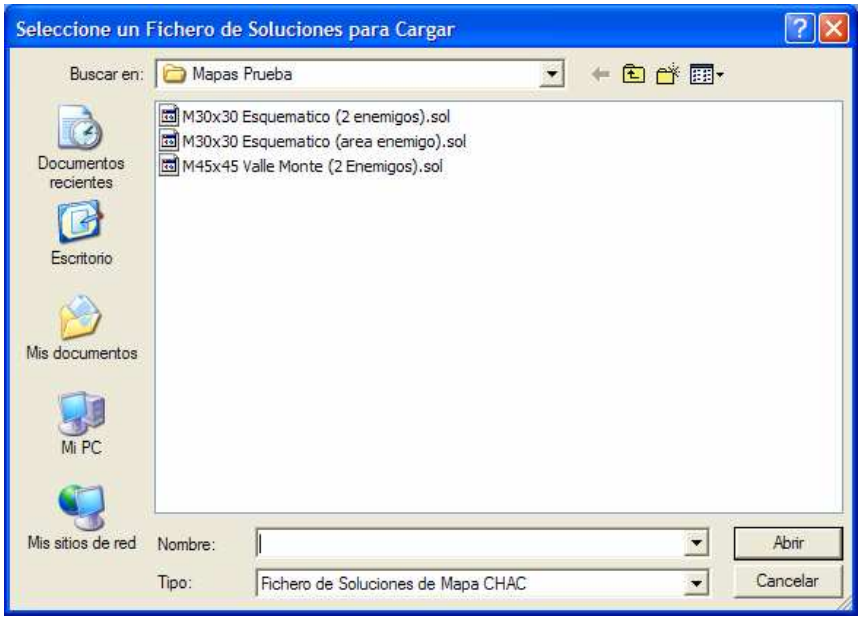
Además, se obtendrá un único fichero de tipo texto ('.txt') por cada ejecución con la misma configuración para cada algoritmo (a excepción de las semillas). En él se podrán ver todos los parámetros utilizados en el mismo y la mejor de las soluciones obtenidas en cada ejecución (según el criterio predominante).
El nombre de dicho fichero tendrá el formato:

[<nombre mapa>] MejoresF_<algoritmo>_<iteraciones><hormigas>_<criterio>.txt

5.2. Cargar Soluciones desde Fichero

Las soluciones obtenidas por los algoritmos, se guardan en el fichero **'.sol'**, de modo que es posible cargar dichos datos (y visualizarlos) en mSS-Hexa.

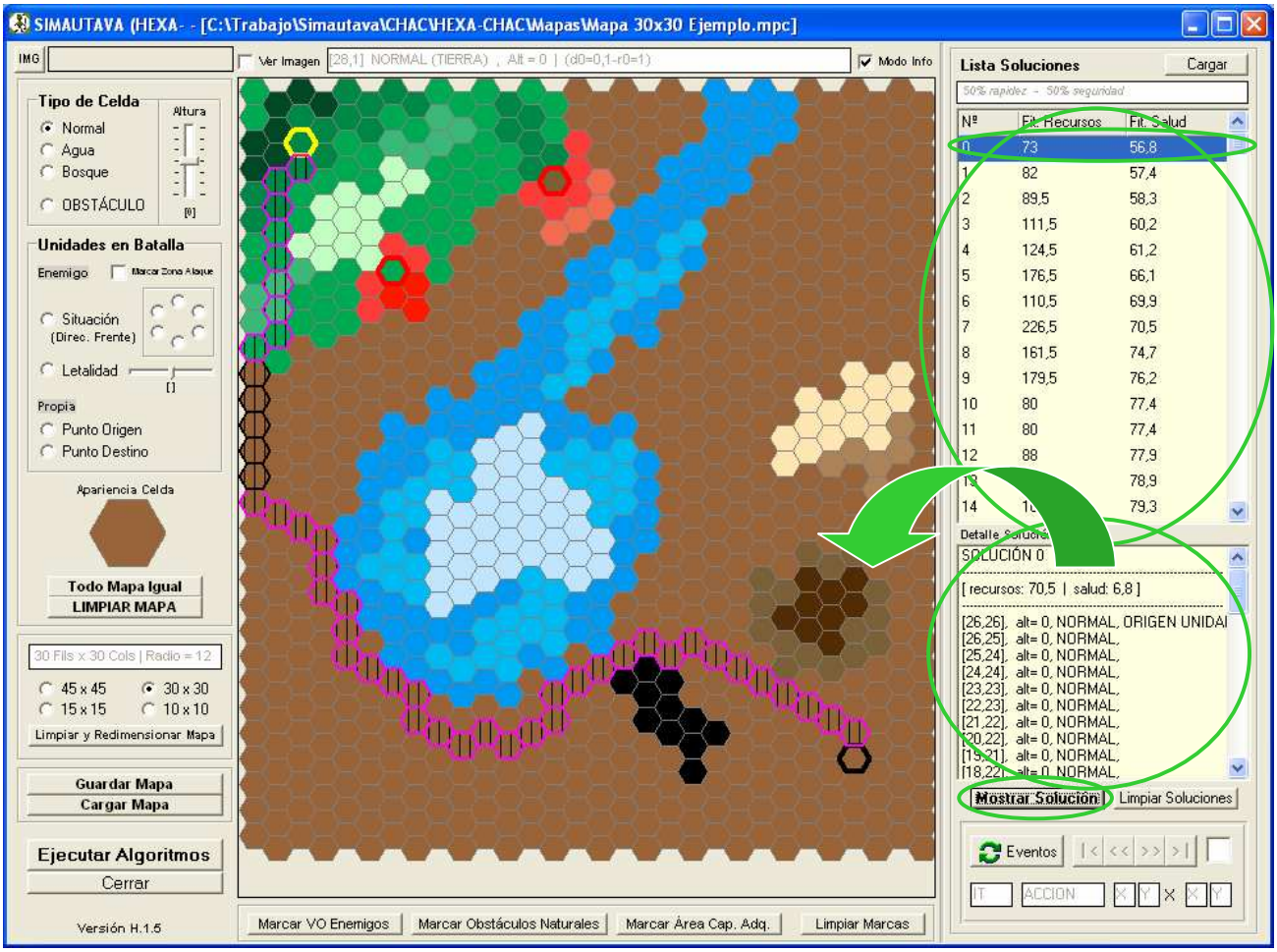
Para ello bastará con pulsar el botón *Cargar* de la parte derecha de la pantalla principal, dónde se muestra la lista de soluciones. Aparecerá una ventana:



En la que podremos elegir el fichero deseado (con extensión '.sol'). Hay que tener en cuenta que no hay ninguna comprobación de que un fichero corresponda con el mapa cargado, por lo que el usuario debería asegurarse de cargar el fichero asociado al mapa que haya cargado previamente.

5.3. Datos de Soluciones

Una vez cargado el fichero de soluciones (o tras acabar la ejecución actual), se mostrarán los datos de las mismas, ordenadas según el criterio que predomine. Entre dichos datos se podrá ver el coste asociado a cada objetivo (salud/seguridad y recursos/rapidez) de la solución, para poder valorarlas, aunque según la teoría de los problemas multiobjetivo, ninguna será mejor que otra (el usuario elegirá la que mejor le parezca si es que tiene que elegir una), ya que las que mejoren en un objetivo empeorarán en el otro.
La pantalla tendrá la forma:



Como se puede ver en la figura, en la parte superior de la lista de soluciones, se muestra el criterio con el que se buscaron dichas soluciones, es decir la relación de rapidez y seguridad con la que se buscó.

Los costes que se muestran en la lista superior no son estrictamente la salud y recursos consumidos, sino unos costes asociados a la solución que tienen en cuenta ambos factores, pero también la visibilidad de las celdas del camino desde las posiciones enemigas, sobretodo en el caso del camino más seguro (en el que penalizan mucho las celdas vistas por los enemigos). Dichos costes serán siempre 2, incluso para los algoritmos que hayan considerado 1 objetivo en la búsqueda (monoCHAC) o 4 (CHAC-4).

La solución se mostrará como celdas redondas con los colores del tipo y altura y con el borde negro en las vistas y rosa/fucsia en las ocultas. Esto se comentará en el apartado 5.4.

NOTA:
En un problema multiobjetivo suelen obtenerse miles de soluciones y este no era una excepción, pero por comodidad para el usuario se han omitido todas que tuvieran el mismo coste (serían prácticamente iguales, una celda en lugar de otra y todas las combinaciones posibles de este caso) y solo se muestra una de ellas.

Lista de Soluciones

Como se puede ver en la imagen anterior, las soluciones se muestran en la parte derecha de la pantalla principal a modo de lista. En dicha lista aparecen 3 valores:

- *número de solución* → para poder identificarla
- *Fitness Recursos* → coste de la solución en cuanto a rapidez. Se consideran tanto los recursos consumidos, como la visibilidad de la celda desde las posiciones enemigas (pero esto en poca medida) en las celdas del camino.
- *Fitness Salud* → coste de la solución en cuanto a seguridad. Se consideran tanto la salud consumida (bajas de no combate y letalidad), como la visibilidad de la celda desde las posiciones enemigas (en gran medida) en las celdas del camino. La visibilidad es muy importante en el coste en seguridad porque las celdas vistas son susceptibles de ser atacadas por el enemigo y, por tanto, mucho más inseguras.

Detalle de Soluciones

Al seleccionar una solución de la lista (cuadro superior de la parte derecha), en el cuadro inferior se muestran sus detalles (ver figura al principio del apartado 4) estos detalles son:

- número de solución
- recursos y salud consumida en el camino (en esta ocasión si es la suma de recursos y de salud real que se consume)
- lista de celdas del camino desde el origen al destino. Se muestran los detalles de cada celda, coordenadas, tipo, altura, subtipo, etc.

5.4. Descripción de Visualización de Soluciones

Una vez se tenga seleccionada una solución (y se muestren sus detalles en el cuadro inferior), es posible verla representada sobre el mapa, para ello bastará con pulsar el botón *Mostrar Solución* (debajo del cuadro de detalle de soluciones).

Tras eso, se dibujará dicha solución sobre el mapa que se tenga cargado (es importante asegurarse de que el fichero de soluciones cargado es el correspondiente a ese mapa).

El camino estará formado por celdas **con borde grueso** que tendrán el color correspondiente al tipo y subtipo que tuviese esa celda en el mapa normal. El borde tendrá diferentes colores y grosores:

- *negro y grueso* → celda origen del camino
- *amarillo y grueso* → celda destino del camino
- ***negro fino*** → celda del camino que es vista desde alguno de (o todos) los enemigos.
- ***rosa grueso*** → celda del camino que es *oculta a todos* los enemigos (tienen obstaculizada su línea de visión hasta esa celda o está fuera de la zona que determina su capacidad de adquisición).

Es posible visualizar diferentes soluciones al mismo tiempo, a fin de poder comparar varias entre si del mismo algoritmo e incluso entre diferentes algoritmos, pues cada una estará rellena con una trama (de rayas negras) en un sentido diferente (horizontal, vertical, diagonales, etc).

Cuando se desee limpiar las marcas de las soluciones en el mapa, bastará con pulsar el botón *Limpiar Soluciones*.

6. FUNCIONALIDAD DE SIMULADOR DE mSS-HEXA

Como ya comentamos, mSS-Hexa es un simulador, aparte de un editor de mapas y visualizador de soluciones. Esto lo ponen de manifiesto varios aspectos ya descritos, como la definición de una serie de estructuras que representan entidades del mundo real, como pueden ser el terreno (con sus tipos, alturas, etc), las unidades militares, que cuentan con características comunes a la realidad (despliegues, recursos, armas, etc) y varias restricciones o condiciones como los consumos de salud (bajas de no combate) y recursos (gasto de combustible) asociados a las celdas.

Estos consumos por defecto definidos son:

	SALUD	RECURSOS
NORMAL	0.1	1
AGUA	0.5	5
BOSQUE	0.3	3
OBSTÁCULO	0	0

Pero además de estos factores, hablamos de mSS-Hexa como simulador porque implementa varias funciones que tienen su reflejo en la realidad (al igual que lo hacen muchos otros) y que hacen más realista el tratamiento del problema y sus soluciones.
Estas funciones no se aplican en una simulación 'en tiempo real' como podría pensarse, sino que se usan a la hora de buscar las soluciones (tanto durante la construcción, como al asignar un coste a la solución) de modo si la unidad pasase a seguir el camino marcado por una de esas soluciones dentro de un simulador dinámico, los resultados se asemejarían mucho a los estimados en los costes de las soluciones que obtienen nuestros algoritmos.

Entre las funcionalidades implementadas se encuentran:

- *Distancia en número de celdas* → en lugar de obtener la distancia entre dos celdas con una fórmula matemática, ésta se calcula trazando una línea entre las celdas y contando el número de ellas que habría que atravesar para llegar de una a otra, lo cual es más realista, puesto que realmente para llegar de una celda a otra se dan un número entero de saltos, no un número real como obtendríamos con otras fórmulas.

- *Obstáculos Naturales* → se considera que hay un obstáculo natural, para la unidad, entre dos celdas si la diferencia de alturas es mayor que la que dicha unidad es capaz de superar. Por defecto se considera que este límite es 2, por lo que se considerará que hay un obstáculo natural entre celdas entre las que haya una diferencia de alturas mayor de 2.

- *Capacidad de Adquisición* → máxima distancia a la que pueden 'ver' (localizar a otras) las unidades, ya sean enemigas o la unidad del problema. Siguiendo los datos reales, esta distancia se fija en 9 Kilómetros, lo que correspondería a 18 celdas.

- *Línea de Visión* → para determinar la visibilidad entre 2 celdas, se tiene esta función que comprueba dicha línea. Se considera que esta línea está obstruida (cada celda es oculta para la otra) si:
  - Hay una celda de tipo bosque en la línea que las une (incluyéndolas a ellas) con una altura igual o superior a la más alta de las dos.

- Ambas celdas tienen la misma altura y existe una celda entre ellas con una altura 2 niveles superior.
- Hay una celda que corta una línea trazada desde la parte superior de una a la parte superior de otra (considerándolas en vista lateral de perfiles). Si es de tipo bosque, se considera que la celda intermedia es un nivel más alta (por la vegetación).

Además, esta línea está limitada por la capacidad de adquisición de la unidad que 'mira'.

- *Consumo de salud (energía)* → se considera como consumo de salud al atravesar una celda a la suma de la penalización que tiene asignada la celda (por su tipo y altura) y la letalidad de la misma.

- *Consumo de recursos* → se considera como consumo de recursos al atravesar una celda *j* viniendo de otra *i* a la suma de recursos por cambio de altura:
  - ninguno si la altura(*i*) = altura(*j*)
  - un valor mayor *c* a mayor cambio de altura, que será *c/2* si altura(*i*) > altura(*j*)

  más el coste en recursos que tenga la celda *j* por defecto.

7. Fichero .INI

El programa dispone de un fichero '**.ini**' en el que se pueden ajustar diversos parámetros, tanto de simulación, como de los algoritmos. Dicho fichero se llama ***mSS_HEXA.ini***.

Cada uno de los parámetros se localiza gracias a una etiqueta entre corchetes '[',']'
Algunos parámetros toman valores reales, utilizando como separación de decimales la coma ','. Dichos parámetros SIEMPRE tendrán decimales (podremos ',0' si queremos que no tenga).
Los comentarios en los ficheros '.ini' se ponen con ';'.

A continuación se describen dichos parámetros, indicando el tipo de los valores que recibirá:

[Unidad] (datos relativos a la unidad del problema)

Capacidad_Adquisicion → (entero). Número de celdas correspondiente a la capacidad de adquisición (cada una es de 500x500 metros).

Maxima_Diferencia_Altura_Puede_Atravesar → (entero). Diferencia de altura que supone un obstáculo natural para la unidad (no lo puede atravesar).

[Enemigos] (datos relativos a las unidades enemigas)

Capacidad_Adquisicion → (entero). Número de celdas correspondiente a la capacidad de adquisición (cada una es de 500x500 metros).

[AlgoritmoCHAC] (parámetros de configuración del algoritmo CHAC y los demás)

Son valores que influyen en la forma en que trabajan los algoritmos y son característicos de los OCH. El más determinante se podría considerar q0, ya que Cuanto mayor sea su valor, mayor explotación se hará de las soluciones, pero menos se explorará en la búsqueda.

ALPHA → (entero). Ponderación feromona
BETA → (entero). Ponderación heurística
RHO → (real). Evaporación de feromona
q0 → (real). Explotación/exploración en SCH
PHI → (real). Evaporación local de feromona en SCH monoobjetivo

Las Constantes de Ponderación de las Heurísticas se encargan de asignar un peso a cada uno de los términos de las funciones que dirigen la búsqueda (hay una función por objetivo y cada una tiene 3 términos).

Heurística del Camino más Rápido:
CTEPOND_N1_RECURSOS → (real). Ponderación del consumo de recursos
CTEPOND_N1_VO → (real). Ponderación de las zonas vistas y ocultas
CTEPOND_N1_DISTANCIA_OBJ → (real). Ponderación de distancia al punto objetivo

Heurística del Camino más Seguro:

CTEPOND_N2_DANIO → (real). Ponderación del consumo de salud (energía)
CTEPOND_N2_VO → (real). Ponderación de las zonas vistas y ocultas
CTEPOND_N2_DISTANCIA_OBJ → (real). Ponderación de distancia al punto objetivo

Las Constantes de Ponderación de las Funciones de Evaluación determinan la importancia del término de la visibilidad, en cada función objetivo, que se considera para evalúar cada una de las soluciones obtenidas.
CTEPOND_EV1_ZVO → (real). Zonas Vistas/Ocultas en objetivo 1: ruta más rápida
CTEPOND_EV2_ZVO → (real). Zonas Vistas/Ocultas en objetivo 2: ruta más segura

8. INSTALACIÓN Y CÓDIGO FUENTE

INSTALACIÓN:

El programa *mSS_Hexa* (o el que se genere si se recompila de nuevo el proyecto) no necesita de instalación alguna, basta con copiar el *.exe* y el *.ini* a la máquina deseada, y disponer del S.O. Windows.

Lo que si es aconsejable es copiar los mapas de ejemplo y plantilla que se adjunta con el programa para poder hacer las pruebas iniciales.

CÓDIGO FUENTE:

Como se comentó anteriormente, el mSS-Hexa ha sido desarrollado en Delphi 7, por lo que los ficheros que contienen el código fuente tienen la extensión *'.pas'*.

En concreto se tienen 5 ficheros:

- *Principal.pas* → contiene las clases, objetos, variables, funciones y procedimientos que controlan la funcionalidad de la pantalla principal.
- *ParamsCHAC.pas* → contiene los elementos que gestionan la funcionalidad de la ventana de parámetros.
- *Simulador.pas* → contiene las constantes, tipos, clases, funciones y procedimientos que son propios de la parte de simulador de mSS-Hexa.
- *Tipos.pas* → constantes y tipos comunes a varios módulos.
- *CHAC.pas* → contiene el código de los algoritmos implementados para resolver el problema.
- *CHAC4.pas* → contiene el código del algoritmo CHAC-4.
- *HexMap.pas* → contiene el código del componente que maneja el grid hexagonal que utiliza la aplicación y que ha sido desarrollado a partir de un componente creado para libre distribución, mejorándolo y añadiendo las funcionalidades necesarias para nuestros requerimientos.

Además se tienen 2 ficheros con extensión *'.dfm'* que son los que contienen la definición de los frames del programa (las ventanas principal y de parámetros), en concreto:

- *Principal.dfm* → definición ventana principal
- *ParamsCHAC.dfm* → definición ventana de parámetros de ejecución.

Estos ficheros están agrupados en un proyecto llamado *mSS_HEXA.dpr*, que se puede abrir y compilar 'en teoría' sin problemas. No obstante es más aconsejable crear un nuevo proyecto e incluir los ficheros nombrados para evitar problemas con las rutas de archivos que haya grabados en el fichero de proyecto que se adjunta.

APÉNDICE I: PUBLICACIONES

Lista de publicaciones relacionadas con la investigación en torno a CHAC:

(Revista Internacional) A.M. Mora, J.J. Merelo, J.L.J. Laredo, C.Millán, J. Torrecillas. **CHAC. A MOACO Algorithm for Computation of Bi-Criteria Military Unit Path in the Battlefield: Presentation and First Results**. International Journal of Intelligent Systems. Aceptado para publicación en 2007. Publicación en 2009.

(Congreso Internacional) A.M. Mora, J.J. Merelo, C.Millán, J. Torrecillas, J.L.J. Laredo. **CHAC. A MOACO Algorithm for Computation of Bi-Criteria Military Unit Path in the Battlefield**. I Workshop in Nature Inspired Cooperative Strategies for Optimization (NICSO'06), D. Pelta y N. Krasnogor Eds., pags. 85-98, Granada (España), Junio, 2006.

(Congreso Nacional) A.M. Mora, J.J. Merelo, C.Millán, J. Torrecillas, J.L.J. Laredo, P.A. Castillo. **CHAC. OCHMO para la Resolución del Problema Bicriterio del Camino Óptimo de una Unidad Militar en el Campo de Batalla**. V Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'07), F. Almeida, B. Melián, J.A. Moreno y J.M. Moreno Eds., pags. 207-214, Tenerife (España), Febrero, 2007.

(Congreso Internacional) A.M. Mora, J.J. Merelo, C.Millán, J. Torrecillas, J.L.J. Laredo, P.A. Castillo. **Enhancing a MOACO for Solving the Bi-Criteria Pathfinding Problem for a Military Unit in a Realistic Battlefield**. First European Workshop on Evolutionary Computation in Transportation and Logistics (EVOTransLog), dentro de EVO* 2007, Lecture Notes in Computer Sciences, Mario Giacobini Ed., pags. 712-721, Valencia (España), Abril, 2007.

(Congreso Internacional) A.M. Mora, J.J. Merelo, C.Millán, J. Torrecillas, J.L.J. Laredo, P.A. Castillo. **Balancing Safety and Speed in the Military Path Finding Problem: Analysis of Different Multi- and Mono-objective Heuristic Functions for an Ant-colony Optimization Method**. Workshop on Defense Applications of Computational Intelligence (DACI 2007), dentro de GECCO 2007, ACM Pres., Dirk Thierens et al. Eds., pags. 2859-2864, Londres (Inglaterra), Julio, 2007. \textbf{- 1 Cita -}

(Congreso Internacional) A.M. Mora, J.J. Merelo, C.Millán, J. Torrecillas, J.L.J. Laredo, P.A. Castillo. **Comparing ACO Algorithms for Solving the Bi-Criteria Military Pathfinding Problem**. 9th European Conference on Artificial Life (ECAL 2007), Springer, Lecture Notes in Artificial Intelligence, F. Almeida et al. Eds., pags. 665-674, Lisboa (Portugal), Septiembre, 2007.

(Congreso Internacional) A.M. Mora, J.J. Merelo, J.L.J. Laredo, P.A. Castillo, P.G. Sánchez, J.P. Sevilla, C. Millán, J. Torrecillas. **hCHAC-4, an ACO Algorithm for Solving the Four-Criteria Military Pathfinding Problem**. Nature Inspired Cooperative Strategies for Optimization (NICSO 2007), Springer, Studies in Computational Intelligence Intelligence, G. Rudolph et al. Eds., vol. 129, pags. 73-84, Sicilia (Italia), Noviembre, 2007.

(Congreso Internacional) A.M. Mora, J.J. Merelo, P.A. Castillo, J.L.J. Laredo, C. Cotta. **Influence of Parameters on the Performance of a MOACO Algorithm for Solving the Bi-Criteria Military Path-finding Problem**. IEEE Congress on Evolutionary Computation (CEC'08), dentro de WCCI 2008, IEEE Pres., pags. 3506-3512, Hong-Kong (China), Junio, 2008.