

UNIVERSIDAD CARLOS III DE MADRID

Departamento de Ingeniería Telemática



Proyecto Fin de Carrera

SimuBattle: Software de recreación y simulación de batallas históricas

Autor: Alejandro Alonso Muñoz
Tutor: Dr. José Jesús García Rueda

Leganés, Diciembre de 2009

PROYECTO FIN DE CARRERA

Departamento de Ingeniería Telemática

Universidad Carlos III de Madrid

Título: SimuBattle: Software de recreación y simulación de batallas históricas

Autor: Alejandro Alonso Muñoz

Tutor: José Jesús García Rueda

EL TRIBUNAL

Presidenta: Raquel M. Crespo García

Secretario: Manuel Urueña Pascual

Vocal: Beatriz de las Heras Herrero

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 21 de diciembre de 2009 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Presidente

Secretario

Vocal

Resumen: SimuBattle es una herramienta que permite a profesores y alumnos de historia, disponer de una aplicación con la que poder estudiar con mayor profundidad, pero también desde un punto de vista más lúdico, cualquier batalla histórica y las causas y consecuencias de la misma. Esto se logra mediante la creación y simulación de las batallas. La creación no sólo permite generar los mapas explicativos, sino además comentarios que los acompañan y los dotan de un mayor didactismo. Por su parte, la simulación permite a los alumnos interactuar directamente e incluso participar en la batalla, variando no sólo los parámetros, sino decidiendo los movimientos de una o varias unidades. SimuBattle permite construir batallas de forma dinámica y cómoda para el usuario, de una manera totalmente gráfica. Por su parte, los comentarios son insertados mediante un editor de texto sencillo. Esto genera una estructura altamente familiar para el usuario, interactiva, lúdica y didáctica.

Agradecimientos

Tras muchos meses de trabajo, esfuerzo y quebraderos de cabeza, ve la luz este trabajo. Supone el fin de una carrera, años de estudio, horas de aprendizaje... No ha sido fácil, pero por el camino muchos han intentado que éste lo fuera de rosas. Es a ellos a los que agradezco su apoyo, su comprensión, su ayuda, sus ánimos y su tesón, que al final han sido los míos.

En primer lugar, siempre, mi familia. Es a ellos a quienes más debo. Mis padres me lo han dado todo, siempre más de lo que tenían. Me han enseñado a andar en la vida por mi propio pie, a esforzarme, a no conformarme con lo mínimo, ambicionar siempre un poco más, ser constante y, sobre todo, a intentar ser siempre mejor persona. Desde pequeño mi hermano mayor ha ejercido como tal, siempre ha sido el referente que necesitaba. Seguramente no siempre habré estado a la altura de las circunstancias, pero lo que sí sé es que ellos han sobrepasado con creces lo que cualquier hijo y hermano podría esperar de ellos. Nunca encontraré maneras adecuadas para agradecerlos todo, os debo demasiado.

A Diana agradecer que me haya acompañado todos estos años, y desde el principio, pues entonces fue cuando nos conocimos. Si algo he de agradecer a mi paso por la universidad, es haberla conocido a ella. Poco a poco fue fraguándose nuestra historia, en clases, laboratorios, exámenes y horas de descanso en el césped y en la cafetería, hasta llegar a San Sebastián. Momentos que no cambiaría por nada en el mundo, salvo por más días de mi vida junto a ella. Juntos estudiamos, juntos hicimos nuestros respectivos proyectos, y pido que juntos hagamos el nuestro en común.

A Saufán agradecer su mano tendida, y que tanto necesité para levantarme de mis caídas. Los comienzos en la universidad fueron muy complicados, y sin su confianza ciega en mí, sin sus palabras de ánimo, sin él como remedio, probablemente no podría haber llegado hasta aquí. Yo más que nunca: gracias.

A mis amigos agradecerles que me hagan el día a día más llevadero, más alegre, más divertido, más feliz. Gracias a todos y cada uno de ellos, y en particular a Dani y a Jorge. Gracias por hacer que la carrera resultara más larga, pero sin duda mucho mejor: noches que ganamos a cambio de días perdidos.

Y por último agradecer la ayuda en este proyecto fin de carrera a mi tutor José Jesús, así como la libertad que desde un principio me dio para poder llevarlo a cabo, a pesar de lo poco “tangible” que parecía la idea en sus comienzos.

Porque siga habiendo muchos más días junto a vosotros. Porque en cada uno de esos días haya muchas más cosas que agradeceros. Porque siga teniendo ocasiones para mostraros dichos agradecimientos... en una tesis final de carrera de Ciencias Políticas, por ejemplo.

Índice

Pliego I: Memoria	9
Capítulo 1: Introducción	11
1.1. Contexto	13
1.2. Motivación	14
1.3. Descripción de los objetivos	17
1.4. Introducción a SimuBattle	20
1.5. Estructura del documento.....	22
Capítulo 2: Bases psicopedagógicas	23
2.1. Infinitas posibilidades	25
2.2. Las bases conductistas	27
2.3. Cognitivismo y constructivismo	31
Capítulo 3: Estado del arte	37
3.1. Wargames	39
3.2. El empleo de videojuegos en el aula	43
3.3. Videojuegos de recreación histórica.....	47
3.4. Serious Games	59
3.5. Objetivos cognitivistas de SimuBattle según aplicaciones anteriores	64
Capítulo 4: Diseño	67
4.1. Introducción	69
4.2. Requisitos	70
4.3. Planteamiento general.....	72
4.4. La interfaz gráfica	75
4.5. Motor de juego	88

4.6 Algoritmo de simulación	91
Capítulo 5: Implementación.....	99
5.1. Implementación	101
5.2. La interfaz gráfica	102
5.3. Motor de juego	108
5.4. Pruebas.....	113
Capítulo 6: Conclusiones y trabajos futuros	115
6.1. Conclusiones.....	117
6.2. Trabajos futuros.....	118
Bibliografía.....	121
Pliego II: Planos UML	125
Pliego III: Manual de usuario	171
1. Sobre SimuBattle	173
2. Empezando.....	174
3. Edición de batallas	176
4. Visualización de la batalla	190
5. Parametrización.....	192
6. Parametrización.....	195
Pliego IV: Presupuesto	201

Pliego I

Memoria

Capítulo 1:

Introducción

1.1. Contexto.....	13
1.2. Motivación.....	14
1.3. Descripción de los objetivos.....	17
1.4. Introducción a SimuBattle	20
1.5. Estructura del documento.....	22

1.1. Contexto

La impresionante expansión que la informática ha experimentado en los últimos años, principalmente con la universalización del acceso a Internet, ha hecho que hoy día se emplee en numerosos ámbitos de la vida diaria. Una de las grandes posibilidades que la informática ha abierto en los últimos años es su utilización en los centros docentes de todos los niveles para ayudar en el estudio y comprensión de numerosas materias. Así, asignaturas de corte clásico como la Historia, pueden verse beneficiadas por las nuevas tecnologías para hacer llegar su comprensión de una manera más rápida, sencilla y, sobre todo, amena debido a la fuerza visual de las aplicaciones multimedia.

Hacer accesible a toda persona, sin necesidad de que sea un experto informático, la posibilidad de crear sus propias aplicaciones multimedia ha de ser uno de los objetivos de los creadores de software. El fin es lograr que toda persona, independientemente de sus conocimientos previos, tenga acceso a unos servicios básicos para poder utilizarlos en sus quehaceres diarios. Así, todo docente que desee incorporar las nuevas tecnologías en sus explicaciones didácticas, ha de poder disfrutar de dicha posibilidad sin que acabe convirtiéndose en un quebradero de cabeza más que en una solución.

Si a la posibilidad de crear aplicaciones didácticas para la impartición de la materia de Historia, añadimos el punto recreativo para así lograr el interés por parte del alumnado, se habrá logrado un objetivo clave en la educación: enseñar y aprender divirtiéndose. Es por ello que cobra mayor fuerza la idea de aunar la tecnología multimedia con los juegos tácticos de guerra clásicos denominados “wargames¹” para la enseñanza de la Historia.

Y esa es precisamente la idea en que se basa este proyecto versado en la creación de una aplicación que permita al profesor disponer de una herramienta, sencilla en su uso pero poderosa en sus fines didácticos, para complementar sus clases de historia. Una aplicación con la que el maestro pueda apoyar sus explicaciones acerca de una batalla histórica y sus consecuencias, mediante su recreación paso por paso y acompañada de comentarios multimedia para hacerlo más atractivo para el alumno. Por su parte, éste, dispondrá con esta aplicación de una herramienta capaz de simular distintas versiones de una misma batalla mediante el cambio de sus parámetros, y así adquirir consciencia de la importancia de distintos factores en la consecución de una victoria militar y sus consecuencias históricas y geopolíticas futuras.

Esta aplicación recibe el nombre de SimuBattle, y permite en una sola herramienta disfrutar de estas tres funciones básicas relativas a la explicación y al desarrollo de una batalla histórica: creación, visualización y simulación.

Pero veamos más detalladamente los motivos por los cuales se optó por abordar este proyecto.

¹ Un wargame (juego de guerra) es un juego que simula y recrea un conflicto bélico mediante el uso de una serie de normas que controlan el funcionamiento del mismo con el fin de determinar el vencedor de la contienda.

1.2. Motivación

Desde el nacimiento de la informática, ésta ha sido vista como un medio para lograr mejorar el trabajo diario y el aprendizaje de las personas en nuestra sociedad moderna. Ya en la década de los 50, algunos pioneros comenzaron a creer firmemente en el gran potencial que la informática podría llegar a tener en la docencia. Las valientes ideas de estos visionarios acabaron demostrándose con el paso de los años, como así demuestra la extensión del uso de elementos multimedia en el ámbito educativo.

Hoy día, la informática es una de las herramientas básicas de trabajo en el mundo universitario, resultando ya imprescindible. Además, su extensión ha ido completándose a todos los ciclos educativos, encontrándose ya presente en los primeros niveles.

Desde muy temprana edad, nuestros jóvenes han crecido rodeados de las nuevas tecnologías. Para ellos, el lenguaje multimedia es el idioma que mejor comprenden y al que están más habituados dado el bombardeo diario de imágenes, sonidos y textos entrelazados al que son sometidos desde los distintos medios de comunicación audiovisuales. Tanto es así, que podría calificarse este lenguaje multimedia de su auténtica lengua materna.

Así pues, si los jóvenes están tan habituados a los sistemas multimedia, ¿por qué no aprovecharlos en el aula para hacer llegar mejor los mensajes educativos que el profesorado quiere lanzar a sus alumnos? ¿Por qué no acercarse los profesores más al entorno comunicativo de sus jóvenes estudiantes? ¿Por qué no modernizar comunicativa y expresivamente materias “clásicas” como la historia?

F. Etxeberria en [18], da incluso un paso más y se aventura a recomendar la utilización de ciertos videojuegos específicos, para nada creados inicialmente con fines docentes, para potenciar algunas habilidades específicas del alumno. Advierte del alejamiento existente entre los alumnos y el estudio de la historia por su identificación con algo lejano y distante, y encuentra en los videojuegos los instrumentos eficaces que aproximan el aprendizaje a la realidad sociotecnológica del alumnado.

“...podemos sacar la conclusión de que el uso de los videojuegos en la ayuda para determinados aprendizajes y entrenamientos es muy positivo, tal y como se demuestra en el terreno del tratamiento de los problemas de aprendizaje... Los videojuegos permiten aumentar la motivación para el aprendizaje de diversas materias como las matemáticas y las ciencias, y el conjunto de las enseñanzas”.

Una de las mayores ventajas de este uso de aplicaciones multimedia es la consecución de un aprendizaje mucho más interactivo. Se favorece la creatividad del alumno y permite al docente alcanzar un alto nivel de motivación en su alumnado, pudiendo incluso él mismo crear actividades mucho más completas que los recursos docentes clásicos con los que siempre se ha trabajado.

Aunar docencia y entretenimiento, didactismo, enseñar deleitando... Estas medidas pueden ser fácilmente asumibles por el profesorado, adoptando

el uso de aplicaciones multimedia en sus enseñanzas diarias para complementar sus explicaciones. Pero, ¿cómo salvar los obstáculos de la brecha tecnológica intergeneracional que muchos profesores pueden llegar a encontrarse?

Muchos profesores, principalmente los de mayor edad, sufren la misma situación que muchos padres que ven cómo sus hijos les sobrepasan con creces en conocimientos informáticos. En estos casos, la reacción más habitual suele ser el temor a toda innovación tecnológica que se presenta ante sus ojos. Es precisamente este pánico el que hay que evitar a toda costa. Las nuevas tecnologías pueden ser consideradas desde tres puntos de vista:

- 1) Como un ataque frontal al statu quo vigente en todos los ámbitos de la vida: comunicación, información, docencia, ámbito laboral, tiempo de ocio...
- 2) Como un “invento moderno” que al final en nada afectará sustancialmente a instituciones y métodos históricamente implantados en la sociedad, como puede ser la enseñanza, y que sólo podrá emplearse en ciertos círculos laborales.
- 3) Como una oportunidad para mejorar todos estos entornos, haciéndolos más eficaces, eficientes, y por qué no, lúdicos.

Sin lugar a dudas, es este último punto de vista el que ha de imponerse en la conciencia del profesorado. Hoy día los medios audiovisuales de que dispone un profesor para complementar sus clases son prácticamente ilimitados. Estamos hablando de múltiples herramientas puestas a disposición del educador, que con un correcto uso pueden facilitar e incrementar enormemente sus posibilidades didácticas. Herramientas que en manos de los alumnos pueden servir a su vez para que asimilen conceptos de manera autodidacta, explorando ellos mismos sus propias capacidades en un entorno educativo mejor adaptado al aprendizaje humano mediante construcciones conceptuales basadas en datos y relaciones entre los mismos. Herramientas que en manos de los padres pueden hacer que se acerquen más a sus hijos, apoyándoles en la adquisición de conocimientos mediante “juegos” didácticos.

Se hace por tanto necesario introducir a los profesores en la utilización de las nuevas tecnologías, mediante herramientas didácticas fáciles en su manejo pero potentes en sus fines pedagógicos. Absolutamente todas las materias pueden ser complementadas mediante programas multimedia, y la historia no es una excepción. En muchas ocasiones, la historia es vista por los alumnos como una materia densa, repleta de datos, nombres y fechas, que impone un excesivo respeto. Las nuevas tecnologías pueden permitir derribar definitivamente este muro de ideas falsas.

Las batallas han marcado el devenir histórico de todas las civilizaciones. Las políticas de pactos entre casas reales, tratados entre países, invasiones, enfrentamientos, guerras... han encauzado la historia de tal manera, que es imposible una comprensión de una materia tan poliédrica sin el estudio de las causas y consecuencias de estas batallas, así como del desarrollo de las mismas y las características de cada una que hicieron caer la balanza de uno u otro lado.

Lo atractiva que resulta una batalla a ojos de un joven, puede servirnos de vía de acceso para lograr atraer su atención hacia el estudio de la historia. Si además, a esto le añadimos la posibilidad de que el alumno interactúe con la aplicación de manera que pueda variar el resultado de las batallas, recrear las suyas propias e incluso participar activamente en las mismas adoptando el lugar de los contendientes, tenemos los ingredientes adecuados para enseñar pasajes de la historia de una manera atractiva y lúdica.

SimuBattle viene a apoyarse precisamente en estas ideas (sencillez y usabilidad, didactismo y entretenimiento en el contexto de la enseñanza de la historia), para ayudar al docente de la materia de historia a complementar la impartición de sus clases de una manera diferente y más cercana al lenguaje audiovisual al que está habituado el alumno de hoy en día.

1.3. Descripción de los objetivos

El objetivo de este proyecto versa en el diseño e implementación de una herramienta software multimedia que permita a cualquier persona (aunque está destinada principalmente a profesores de historia) la recreación de aquella batalla histórica que desee con el fin de ser mostrada a sus alumnos.

Esta funcionalidad básica ha de permitir al “autor” de la batalla una recreación sobre el mapa que abarque los puntos más importantes de la contienda, permitiendo un paso a paso individual de cada patrulla, o una abstracción mayor. Los medios para dicha simulación han de permitir utilizar unidades de distinto tipo: terrestres, navales, aéreas... Esta libertad a la hora de representar los distintos contendientes y el terreno sobre el que se libra la batalla, se encapsularía en una representación clásica de mapa cual tablero para permitir una visualización más completa y a la vez ordenada de la batalla.

Esta representación paso a paso de la batalla mediante mapas sobre los que se sitúan las distintas unidades, se acompañaría de una serie de comentarios creados libremente por el docente mediante textos HTML. Esta elección del lenguaje HTML permite la generación de descripciones muy completas gracias a sus distintos tipos de letra, colores, enlaces a páginas web donde profundizar en la información, imágenes e incluso vídeos. Para facilitar la creación de estos comentarios, se procederá a la implementación de un editor de textos HTML muy sencillo a ojos del usuario, pues su funcionamiento es similar a cualquier editor de textos de sobra conocido por todos como puede ser el Microsoft Word, pero con una estructura de código que permite convertirlo en textos multimedia.

A esta funcionalidad de editor de recreación de batallas y visualización de las mismas, se le une el objetivo de permitir al usuario final (el alumno) interactuar con la aplicación mediante la simulación de dicha batalla a modo de wargame¹, de manera automática mediante la elección de unos parámetros como (ataque, defensa, moral, inteligencia...), o incluso directamente, siendo el propio estudiante quien decida los pasos y disparos de las unidades en liza.

Así pues, el proyecto consta de tres funciones básicas, claramente diferenciadas, pero a la postre interconectadas entre sí: creación de una batalla, visualización de la misma y finalmente simulación (con su correspondiente parametrización).

Estos objetivos se fundamentan en el deseo firme de crear una aplicación didáctica e interactiva para hacer llegar la historia de una manera amena a los alumnos más jóvenes, permitiéndoles participar activamente mediante simulaciones ucrónicas de batallas, incluso con su intervención directa en las mismas.

De forma más concreta, los objetivos en este sentido son:

- Diseñar una interfaz de usuario que permita a cualquier persona, sin unos conocimientos técnicos avanzados, hacer uso de ella, prestando por tanto especial interés a su usabilidad y su funcionalidad.

- Proporcionar una representación icónica adecuada de la batalla a recrear.
 - Implementar un método totalmente gráfico de construcción del paso a paso de la batalla que resulte sencillo, utilizando para ello eventos de ratón.
 - Implementar un editor para el contenido audiovisual intuitivo y familiar, que permita la inclusión de texto, enlaces, imágenes, audio y videos.
- Conseguir un entorno de trabajo interactivo en el que el usuario pueda crear sus propias batallas de una manera rápida e intuitiva, permitiéndole gozar de la mayor libertad posible a la hora de generar, modificar y borrar los componentes de la batalla.
 - Crear un entorno de visualización de la batalla generada por el usuario, independiente del entorno de creación, para el uso de la misma por parte de los alumnos.
 - Crear un entorno de simulación de la batalla histórica, accesible a través de la batalla generada inicialmente por el docente. Los objetivos en este apartado de la herramienta son:
 - Permitir al usuario total libertad de parametrizar los distintos aspectos de la batalla a su gusto, y en cualquier instante.
 - Posibilitar que el usuario no sólo avance en la simulación a la velocidad deseada, sino que además pueda retroceder e incluso reanudar la simulación en un instante anterior de la batalla.
 - Permitir que el usuario tome aún parte más activa en la batalla, pudiendo seleccionar él mismo los movimientos de una unidad concreta, varias de cualquier bando, todas las de un bando o ambos.
 - Tal y como se ha explicado anteriormente, la herramienta consta de tres funciones básicas con respecto a las batallas: creación, visualización y simulación. Un objetivo claro de la herramienta es poder disfrutar de estas tres funciones en un único entorno, permitiendo al usuario libremente pasar de una a otra en cualquier momento de su utilización.
 - Aprovechar las capacidades multiplataforma del lenguaje de programación Java, optando por implementar la aplicación en dicho lenguaje.

Por otro lado, hasta el momento se ha hecho especial hincapié en el hecho de que SimuBattle nace con el objetivo de favorecer la tarea docente a los profesores, interactuando con los alumnos para hacer de la historia una materia más amena adaptando la utilización de wargames (juegos de guerra) de corte clásico. Siguiendo este razonamiento, y para lograr una mayor comprensión de los fundamentos en que se basa este proyecto, se pretende realizar un pequeño análisis de la utilización de las nuevas tecnologías en la

docencia, así como del origen y evolución de los wargames y su posible uso didáctico. De igual forma, es también objetivo de este proyecto, extraer conclusiones del análisis anterior y esbozar unas pautas de acción en este campo que allane el camino a futuros proyectos de herramientas audiovisuales con fines didácticos.

Finalmente, en una fase posterior del proyecto, se realizó, entregándoselo a usuarios noveles, una serie de pruebas para comprobar su estabilidad, su robustez, su respuesta ante eventos no esperados...

1.4. Introducción a SimuBattle

Antes de profundizar más, presentemos brevemente la aplicación SimuBattle, para así disponer de aquí en adelante de una referencia más clara. SimuBattle es una aplicación lúdico-didáctica cuyo fin es el apoyo al docente en la impartición de clases de historia en los niveles primarios y secundarios de la enseñanza.

Para ello, SimuBattle consta de varias aplicaciones interdependientes con el fin de hacer que el alumno descubra por sí mismo distintos aspectos de la historia a través de la recreación y simulación de batallas históricas, guiado siempre de manera directa o indirecta por el tutor en todo este proceso. Para ello, el docente cuenta inicialmente con una aplicación “creadora de batallas”, donde se generan los mapas y las posiciones de las distintas unidades a través de la batalla, así como los comentarios que el profesor considere oportunos para explicar aquellos aspectos que considere más destacados, mediante un editor de textos HTML. El siguiente paso a realizar por el docente puede ser “parametrizar” la batalla, dando valores iniciales a las celdas del mapa y a las distintas unidades, calificándolas de aéreas, terrestres, navales, cañones; fijar su vida, moral, fuerza de ataque, defensa, inteligencia... A continuación, el alumno puede optar por visualizar la batalla generada por su profesor, y simularla, modificando a su gusto los parámetros e incluso tomando parte en la misma, observando así directamente qué factores fueron determinantes para la resolución de la batalla, así como sus consecuencias geopolíticas.

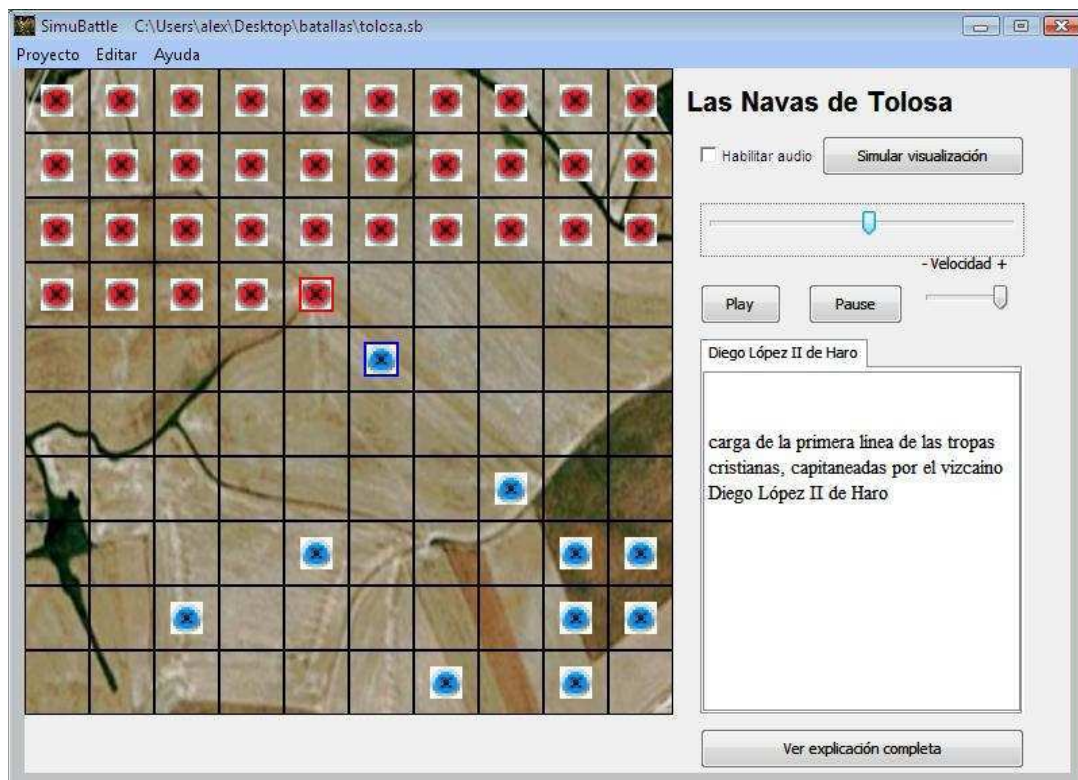


Figura 1.1. SimuBattle en funcionamiento

Para la creación de SimuBattle, se tuvieron muy en cuenta factores esenciales como la sencillez y usabilidad dado el tipo de usuarios a los que va

destinada esta herramienta. Para ello se optó por crear una única interfaz que aglutinara todas sus funcionalidades. Así, mediante un diseño único y similar para todas las opciones, se logra una mayor familiaridad con el entorno, así como un recorrido libre del usuario por la aplicación, semejante a la navegación web que tan conocida es hoy día por todos.

Posteriormente, se describirá con total detalle el proceso de diseño e implementación de la herramienta, así como las pruebas realizadas con la misma. También el estudio teórico sobre aplicaciones multimedia didácticas y su empleo efectivo en el aula. El siguiente punto muestra una guía que ayudará al lector a encontrar de forma fácil cada uno de estos aspectos a lo largo de este documento.

1.5. Estructura del documento

Este apartado consiste en una guía del presente documento, realizando un repaso por los diferentes capítulos de los que consta:

- Capítulo 1: Introducción. Es el capítulo presente, en el que se contiene una introducción breve al resto del documento, así como los objetivos y motivaciones que han llevado a la realización de este proyecto.
- Capítulo 2: Bases psicopedagógicas. Repaso al proceso seguido por las principales teorías pedagógicas y el espacio que los nuevos medios tecnológicos han pasado a ocupar, basándose en las mismas, en el aula. De esta forma, se entronca con las bases psicopedagógicas en las que se basa SimuBattle.
- Capítulo 3: Estado del arte. Recopilación de trabajos relacionados con SimuBattle para concretar la situación actual de diversas aplicaciones lúdico-didácticas que podemos encontrarnos, y observar el espacio que la herramienta SimuBattle está destinada a ocupar.
- Capítulo 4: Diseño. Describe el proceso de diseño de la aplicación, comentando todas aquellas decisiones adoptadas y que pueden aportar riqueza a la descripción de la solución final tomada.
- Capítulo 5: Implementación. Describe el proceso de desarrollo y creación software de la aplicación a un nivel puramente técnico, así como las pruebas técnicas llevadas a cabo con la herramienta final para su mejora y estudio de sus capacidades.
- Capítulo 6: Conclusiones y trabajos futuros. Incluye las conclusiones obtenidas a partir del proyecto realizado, así como las líneas en las que se puede continuar trabajando con posterioridad.

Capítulo 2:

Bases psicopedagógicas

2.1. Infinitas posibilidades	25
2.2. Las bases conductistas	27
2.3. Cognitivismo y constructivismo	31

2.1. Infinitas posibilidades

*“Creía en infinitas series de tiempos, en una red creciente y vertiginosa de tiempos divergentes, convergentes y paralelos. Esa trama de tiempos que se aproximan, se bifurcan, se cortan o que secularmente se ignoran, abarca todas las posibilidades”.
El jardín de senderos que se bifurcan (Ficciones). Jorge Luis Borges..*

Nuestra vida diaria ha sufrido drásticos cambios en los últimos años gracias a los avances tecnológicos. Nuestro trabajo, nuestro ocio, la forma de comunicarnos o de acercarnos a la información, han variado radicalmente, principalmente desde la imparable expansión de Internet, convirtiéndolo en un auténtico cambio sociológico, al que Manuel Castells ha pasado a denominar “Sociedad Red”, y que en [25] definía así:

“La difusión y desarrollo de ese sistema tecnológico ha cambiado la base material de nuestras vidas, por tanto la vida misma, en todos sus aspectos: en cómo producimos, cómo y en qué trabajamos, cómo y qué consumimos, cómo nos educamos, cómo nos informamos-entretenemos, cómo vendemos, cómo nos arruinamos, cómo gobernamos, cómo hacemos la guerra y la paz, cómo nacemos y cómo morimos, y quién manda, quién se enriquece, quién explota, quién sufre y quién se margina. Las nuevas tecnologías de información no determinan lo que pasa en la sociedad, pero cambian tan profundamente las reglas del juego que debemos aprender de nuevo, colectivamente, cuál es nuestra nueva realidad, o sufriremos, individualmente, el control de los pocos (países o personas) que conozcan los códigos de acceso a las fuentes de saber y poder.”

Se hace por tanto imprescindible no dejar escapar este tren, y aprovechar al máximo todas las posibilidades que la revolución tecnológica nos ofrece, incluido el ámbito de la enseñanza.

Desde el nacimiento de la informática, se la ha visto como un medio para lograr mejorar el trabajo diario y el aprendizaje del hombre en nuestra sociedad. Ya en la década de los 50 en EEUU empezó a visualizarse el potencial que la informática tenía en la docencia, demostrándose con el paso de los años y extendiéndose al ámbito educativo. Hoy día, la informática es la herramienta básica de trabajo en el mundo universitario, resultando ya imprescindible. Además, su extensión ha ido completando todos los ciclos educativos, aplicándose ya en los primeros niveles a muy temprana edad.

Específicamente, una variante tecnológica que históricamente ha sido empleada en la docencia es la tecnología denominada como multimedia: una forma de transmitir información a través de sistemas informáticos empleando para ello una combinación de distintos medios como son: texto, sonido, fotografías, video... Pero además, su principal característica es que son presentados de manera integrada en un todo coherente. En la mayoría de los casos, es empleado el multimedia interactivo, donde esta información así presentada en tiempo real no es del todo estática, sino que permite una interacción por parte del usuario.

Es precisamente esta interacción con el usuario la que dota a los sistemas multimedia de un mayor potencial didáctico, al permitir al alumno adquirir conocimientos de una manera continuada, pero siempre adaptada a

sus propias necesidades, pues es él quien “camina” por la aplicación descubriendo los distintos aspectos didácticos a su propio ritmo.

Es por ello que una característica que suele ser deseable en una aplicación multimedia es lograr que el usuario se sienta libre dentro de ella, es decir, que navegue a su antojo por la misma yendo de un punto a otro. Esta “libertad” no es total, pues siempre se sigue un patrón predefinido por el autor de la aplicación, permitiendo cierta libertad de elección pero siempre dentro de unos caminos conocidos y prefijados. Una buena aplicación es aquella que da la sensación al usuario de que él es el dueño y señor de su andadura dentro de ella.

Toda la información que aparece, por tanto, en un sistema multimedia, es presentada a través de numerosos canales o medios, donde los básicos son: texto, imagen y sonido. Estos medios han de permitir la interacción con el usuario para que éste sea el que decida en cada momento por qué media optar para adquirir los conocimientos deseables en una aplicación destinada a la docencia. Para su correcta representación se requiere de un hardware y un software mínimo que haga posible la reproducción en tiempo real de estos medios, e incluso su elaboración y creación para todo tipo de aplicaciones multimedia.

Resulta, por tanto, imprescindible, desterrar la añeja idea de que las nuevas tecnologías en general, y el ordenador en particular, son empleados laboralmente por altos expertos, y que su uso doméstico o cotidiano queda reducido únicamente a usos lúdicos y de ocio. Es más, es posible aunar ocio y aprendizaje en una misma aplicación multimedia, e incluso emplear una herramienta inicialmente destinada al puro entretenimiento como un videojuego, en el aula para transmitir conocimientos de historia a los alumnos.

En las próximas páginas, el lector podrá profundizar en las teorías pedagógicas en las que se basan estas ideas, realizará un trayecto histórico por la evolución de éstas hasta llegar a las bases cognitivistas y constructivistas en las que se asienta SimuBattle, y además podrá hacer un recorrido por los distintos usos en el aula que de las tecnologías pueden hacerse, que de entrada pueden calificarse de infinitas en la práctica, y el objetivo es, precisamente, empezar a descubrirlas.

2.2. Las bases conductistas

“Te han convertido en algo que ya no es una criatura humana. Ya no estás en condiciones de elegir. Estás obligado a tener una conducta que la sociedad considera aceptable, y eres una maquinita que sólo puede hacer el bien.”
La naranja mecánica. Anthony Burgess.

¿De qué manera ayudan los sistemas multimedia a asentar los conocimientos en los estudiantes? Los pedagogos han estudiado a fondo, especialmente durante los últimos años, la forma más adecuada para adaptar las características de los programas multimedia al funcionamiento del cerebro humano. En [24] se ve cómo históricamente el modelo más extendido era la concepción conductista de la enseñanza, que bebe de la anteriormente elaborada pedagogía pragmática. El conductismo consiste en la creencia de que el aprendizaje se basa en una serie de estímulos reforzados por los premios y los castigos, al igual que la existencia de modelos a partir de los cuales tallar el comportamiento humano. Los sistemas multimedia, sin embargo, permiten ir más allá y fomentar la creatividad del propio alumno, darle libertad y adaptarse a su velocidad de aprendizaje. Estos últimos son aspectos básicos de la concepción cognitiva.

Estas dos distintas teorías educativas, enfrentadas en su concepción del aprendizaje humano, nos darán una idea de cómo han evolucionado las teorías educativas en las últimas décadas, por lo que, profundizando en sus aspectos más básicos, nos permitirá más adelante conocer y comprender mejor las bases del cognitivismo en las que entroncan ciertos sistemas multimedia.

Pragmatismo

La filosofía pragmática surgió en los Estados Unidos en la segunda mitad del siglo XIX, con Charles Peirce, William James y John Dewey como principales valedores. Esta filosofía se basa en afirmar que el contenido de las ideas y conceptos se reduce a los efectos prácticos que se pueden obtener a través de ellas. Con estos postulados, el pragmatismo fue caracterizado, y criticado precisamente, por ser una teoría excesivamente subjetivista e individualista.

Como puede observarse en [26], John Dewey en las primeras décadas del siglo XX se mostró partidario de transformar la teoría y la práctica docente, a partir de considerar que el sistema imperante en aquel momento era insuficiente con relación a la preparación de los individuos para vivir en una sociedad democrática, ya que veía el desarrollo social como algo estático enmarcado en la concepción tradicional de la educación. Podría resumirse su “pedagogía de la acción” en esta frase:

“Puesto que todo saber nace de una situación problemática real, debe ponerse al niño en una situación en la que tenga que enfrentarse a problemas, para que sea capaz de inventar hipótesis, deducir consecuencias de éstas y llevarlas a la práctica. Debe ser una enseñanza de abajo a arriba.”

La pedagogía pragmática se basa, por tanto, en la obtención de resultados, centrándose únicamente en aquellas materias de las cuales espera

obtenerse una utilidad práctica, aquellas que acabarán reportando algún interés. En esta situación, el educador se convierte en un técnico que acompaña y asiste al alumno como guía durante su aprendizaje.

Conductismo

Al hablar de conductismo, frecuentemente se enlaza con palabras tales como “estímulo”, “respuesta”, “refuerzo”. Esto da una idea del carácter procedimental basado estrictamente en experimentos para estudiar el comportamiento observable (la conducta), considerando el entorno como un conjunto de estímulos.

Nació como corriente psicológica influida por las ideas de minimizar el estudio introspectivo de los procesos mentales, las emociones y los sentimientos, sustituyéndolos por el estudio objetivo de los comportamientos de los individuos en relación con el medio, siempre mediante el empleo de métodos experimentales. Basándose inicialmente en el pragmatismo en cuanto a su pilar básico de obtención de resultados, evolucionó hacia un estadio en el que la guía del profesor acaba resultando prescindible, ahondando en la idea de individualismo frente al aprendizaje.

El conductismo actual ha influido en la psicología en tres campos:

- Ha reemplazado la concepción mecánica de la relación estímulo-respuesta por otra más funcional que hace hincapié en el significado de las condiciones estímulares para el individuo.
- Ha introducido el empleo del método experimental para el estudio de los casos individuales.
- Ha demostrado que los conceptos y los principios conductistas son útiles para ayudar a resolver problemas prácticos en diversas áreas de la psicología aplicada.

John B. Watson, psicólogo americano, es considerado el padre del conductismo, al considerar que la psicología debería centrarse en la conducta humana en lugar de la mente y su conciencia, y así poder estudiar objetivamente a los seres humanos como se estudia a los animales. Esta teoría se basaba en los experimentos de Iván Petrovich Pavlov (1849-1936), fisiólogo ruso ganador del premio Nobel en 1904 por sus investigaciones sobre el funcionamiento de las glándulas digestivas, entre los que se encuentra el más conocido de ellos: la respuesta de los perros al condicionamiento.

Este experimento comenzó al observar Pavlov que los perros salivaban al mostrárseles carne en polvo. Ante ello Pavlov comenzó su experimento mediante el toque de una campana. Inicialmente este estímulo era neutral, pues no suscitaba una respuesta. Tras el sonido de la campana acercaba a los perros carne en polvo, lo que les provocaba salivación. Este proceso continuado (toque de campana, comida, salivación), fue repetido una y otra vez. Con el tiempo, ante el mero toque de campana, los perros comenzaban a salivar incluso sin mostrárseles la comida. El estímulo neutral que inicialmente era la campana, acabó convirtiéndose en el estímulo que hacía salivar a los perros, se había producido el condicionamiento. En [27], Dennis Coon lo explica así:

“Los psicólogos usan varios términos para describir estos eventos. La campana en el experimento de Pavlov es inicialmente un estímulo neutral. Con el tiempo, la campana llega a ser un estímulo condicionado (estímulo que, debido al aprendizaje, provocará una respuesta). La carne en polvo es un estímulo incondicionado (estímulo capaz de provocar una respuesta en forma innata). En vista de que un reflejo está ‘integrado’, se le llama respuesta incondicionada (no aprendida). La salivación refleja era la respuesta incondicionada en el experimento de Pavlov. Cuando la campana también producía salivación, el perro estaba emitiendo una respuesta nueva. Así, la salivación se había convertido en una respuesta condicionada (aprendida).”

Así, Watson entendía que todos los procesos psíquicos, incluso los más complejos como el pensamiento y el lenguaje, podían reducirse a meras respuestas simples (musculares, glandulares, celulares...). Por tanto el conductismo es, por tanto, naturalista, en el sentido en que el mundo material es la última realidad, y todo puede ser explicado en términos de leyes naturales. El hombre es al fin y al cabo un cerebro que responde al estímulo externo, responde al condicionamiento. En palabras de David Cohen, recogidas en [28]:

“El principio central del conductismo es que todos los pensamientos, sentimientos e intenciones, todos ellos procesos mentales, no determinan lo que hacemos. Nuestra conducta es el producto de nuestro condicionamiento. Somos máquinas biológicas y no actuamos conscientemente; más bien reaccionamos al estímulo”.

Siguiendo las teorías de Watson, B. F. Skinner (1904-1990) elaboró el llamado conductismo radical, en el cual difiere de Watson en la consideración de los fenómenos psíquicos internos, ensalzando su importancia y considerándolos objetos de estudio científico. Así, hace hincapié en la importancia subjetiva que para el sujeto tiene el estímulo que recibe, ya sea positivo o negativo.

Pero si por algo, en este análisis, destaca la aportación de Skinner a la tecnología educativa, es su desarrollo de la enseñanza programada en 1954 a partir de las máquinas de enseñar creadas en la década de 1920. Estas primeras máquinas de enseñanza fueron diseñadas por el psicólogo estadounidense Sidney Leavitt Pressey, y consistían en proporcionar una respuesta inmediata a pruebas de elección múltiple. Esa corrección inmediata de los errores era la que servía como función para la enseñanza, y permitía a los estudiantes practicar una y otra vez con los ejercicios hasta lograr que todas las respuestas fueran correctas.

La enseñanza programada es definida por varios autores de la siguiente manera (y que podemos encontrar resumidas en [26]):

- *“Modelo provisto de objetivos conductuales, contenido en forma lógica y en secuencia de unidades, métodos basados en el autoaprendizaje, (preguntas y respuestas, simulación, juegos didácticos), empleo de libros, computadoras, televisión, etc., (...). La concepción de aprendizaje es entendida como un cambio estable en la conducta del alumno, es un modelo de ensayo-error donde el sujeto produce conductas diferentes*

hasta que logra la conexión con el medio y el resultado deseado”. (Hernández Rabell, L. 2006:38).

- *“La enseñanza programada (EP) es un método pedagógico que permite transmitir conocimientos sin la medición directa de un profesor o monitor, respetando las características específicas de cada alumno considerado individualmente”. (León Fonseca, M. 2005:4).*

Así pues, este tipo de enseñanza se desarrolla mediante un modelo de aprendizaje conductista donde el alumno es el principal responsable y valedor de su propio aprendizaje, puesto que no hay intervención directa de un profesor. Existe una interacción unilateral entre alumno y medio de aprendizaje, meramente conductista pues la esfera motivacional-afectiva, la cognitiva y las interacciones entre los actuantes del proceso, quedaban al margen.

Este método de enseñanza se libera del “peso” que muchos conductistas consideraban las relaciones interpersonales, que no hacían más que entorpecer el aprendizaje. Además, reforzaban la idea de lucha psicológica entre el estudiante y el “error” para alcanzar el éxito final. Estas ideas las resume León Fonseca en [29]:

- *“La enseñanza programada libera al alumno del peso de las relaciones de simpatía y antipatía hacia el profesor y sus discípulos, lo ayuda a verificar de esta manera el proceso de aprendizaje sin perturbaciones del tipo emocional social”.*
- *“Desde el punto de vista psicológico, en el caso de los adolescentes, resulta significativa la lucha que tratan de sostener para que la máquina no les señale errores y poder salir vencedores contra ella, lo cual refuerza aspectos importantes de la personalidad, tales como la perseverancia, la constancia, el esfuerzo...”.*

Las limitaciones de este conductismo, y las críticas recibidas, resultan evidentes, como pueden ser la creación de un concepto de ser humano individualista, egoísta, alejado de la sociedad y sumergido en su propio mundo de realizaciones personales, por encima siempre de las sociales. De igual forma se ignoran valores de la socialización tan importantes como la aceptación, el respeto mutuo, saber escuchar, mantener la atención...

Tal y como hemos visto, en los inicios de la enseñanza programada, ésta se inspiró básicamente en las teorías conductistas. Poco a poco, con su evolución, se vio rápidamente que en muchos ámbitos de la enseñanza ésta no era la manera más oportuna para lograr impartir conocimientos a los alumnos, ganando cada vez más terreno las teorías cognitivistas y constructivistas que veremos a continuación, y que han logrado una gran difusión gracias a las enormes posibilidades que ofrecen las nuevas tecnologías.

2.3. Cognitivismo y constructivismo

“Me asombraba, por ejemplo, la capacidad de las palabras para encontrarse con los objetos que nombraban... Me parecía inexplicable en cambio que si al pronunciar la palabra gato aparecía un gato dentro de mi cabeza, al decir ‘ga’ no apareciera medio gato. No le dije nada a mi madre para no preocuparla...”
El mundo. Juan José Millás.

La concepción cognitiva se apoya en la idea de que no existe una verdad absoluta. Aprender no significa necesariamente ir acumulando conocimientos o establecer una colección creciente de “pequeñas verdades” de manera continuada y lineal como ocurría en la concepción conductista. Sostiene que todo conocimiento se basa, en último término, en supuestos transitorios, los que por principio están abiertos a refutación y debate. En el cognitivismo resulta clave, por tanto, la construcción del conocimiento por parte de quien aprende: “aprender a aprender”.

Jerome S. Bruner es considerado como uno de los padres y máximos valedores del cognitivismo. Nacido en Nueva York en 1915, este doctor en psicología por la universidad de Harvard es uno de los principales representantes del movimiento cognitivista y uno de los que promueven el cambio de modelo instruccional, desde el enfoque de las teorías del aprendizaje, propias del conductismo, a un enfoque más cognoscitivo y simbólico. Propone una teoría de la instrucción que intente exponer los mejores medios de aprender lo que se quiere enseñar; relacionada con mejorar más bien que con describir el aprendizaje.

La obra de Bruner se inicia en la década de los 50 con “*A study of thinking*” (Bruner, Goodnow y Austin, 1956), destacando entre sus obras principales “*The process of Education*” (1961) y “*Toward a theory of instruction*” (1966). En estas obras ya se vislumbra un interés especial por el proceso de instrucción basado en una perspectiva cognitiva del aprendizaje.

Los rasgos esenciales de su teoría, y que pueden encontrarse resumidos en [30] se refieren a:

- Importancia de la estructura: El alumno ha de descubrir por sí mismo la estructura de aquello que va a aprender. Esta estructura está constituida por las ideas fundamentales y las relaciones que se establecen entre ellas. Estas estructuras deben adecuarse a la capacidad intelectual y a los conocimientos previos del alumno, mediante una secuenciación adecuada, por tanto, la mejor manera de organizar los conceptos es encontrar un sistema de codificación que permita llegar a la estructura fundamental de la materia que se estudia. Además, la comprensión de la estructura de cualquier materia es requisito para la aplicabilidad a nuevos problemas que se encontrará el alumno fuera o dentro del aula.
- Propuesta de un diseño de currículum en espiral: Un plan de estudios ideal es aquel que ofrece materiales y contenidos de enseñanza a niveles cada vez más amplios y profundos, y al mismo tiempo, que se adapten a las posibilidades del alumno definidas por su desarrollo evolutivo. Por tanto, el currículum debe ser en espiral y no lineal, volviendo constantemente a retomar, y a niveles cada vez más elevados, los

núcleos básicos o estructuras de cada materia. Esta organización de las materias de enseñanza refleja su opinión de que el aprendizaje procede de lo simple a lo complejo, de lo concreto a lo abstracto y de lo específico a lo general, de forma inductiva.

- Aprendizaje por descubrimiento: El aprendizaje debe ser descubierto activamente por el alumno más que pasivamente asimilado. Los alumnos deben ser estimulados a descubrir por cuenta propia, a formular conjeturas y a exponer sus propios puntos de vista. Como se dijo, recomienda el fomento del pensamiento intuitivo. Entre las ventajas del aprendizaje por descubrimiento se encuentran el hecho de que se enseña al alumno la manera de aprender los procedimientos, se produce en el alumno automotivación y fortalece su autoconcepto, se desarrolla su capacidad crítica al permitirle hacer nuevas conjeturas, y el alumno es responsable de su propio proceso de aprendizaje.

A partir del cognitivismo, el psicólogo evolutivo y epistemólogo suizo Jean Piaget elaboró su propia teoría constructivista. Estudió cómo el conocimiento está representado en la mente, qué funciones u operaciones permiten el cambio de estas representaciones y qué estadios se pueden distinguir a través del desarrollo evolutivo. Para Piaget, la inteligencia tiene su origen en la acción. A través de la experiencia las acciones se van coordinando, formando estructuras cognitivas cada vez más complejas. Durante siete décadas Piaget desarrolló un programa de investigación creando la llamada “Escuela de Ginebra” y convirtiendo su teoría en el referente más importante para la explicación del desarrollo cognitivo del ser humano.

Piaget demuestra a lo largo de sus estudios que la lógica y el pensamiento se desarrollan en el niño antes que el lenguaje, y todo ello por medio de sus acciones motrices y sensoriales, los estímulos recibidos por el entorno que lo rodea, y la información que extrae de ambos y que es capaz de procesar. Deduce, por tanto, que la actividad cognitiva del niño está invariablemente ligada a la sociedad y a su entorno [31].

El conocimiento se compone de esquemas cognitivos que se van desarrollando unos sobre otros, se perfeccionan y se hacen cada vez más complejos por aplicación de la lógica inherente al pensamiento humano. Así, el niño nace esencialmente con la capacidad de la lógica y algún otro conocimiento muy básico, y sobre ella va construyendo todo su conocimiento posterior mediante asimilación de nuevos eventos a las estructuras propias de su entendimiento y comportamiento, y mediante la acomodación de las estructuras mentales ya adquiridas para dar cabida a nuevos objetos o eventos.

Se observa por tanto, en sus teorías, la importancia de la idea de que el nuevo conocimiento se construye sobre conocimiento adquirido con anterioridad, en un proceso constante de asimilación y acomodación del mismo. Los individuos se implican activamente en la construcción de una comprensión personal de los nuevos datos que van adquiriendo, para lo que resulta fundamental partir de la propia experiencia.

Esta “construcción” del conocimiento es lo que dio origen al movimiento constructivista, una evolución del propio cognitivismo a partir de la obra de

Piaget, resultando en muchas ocasiones confuso distinguir el cognitivismo del constructivismo. El constructivismo hace más hincapié en la interacción con otros alumnos en el proceso de construcción del conocimiento, con la retroalimentación como factor clave del aprendizaje, mientras que el cognitivismo es un proceso independiente de decodificación de significados que fomenta el desarrollo de estrategias que permiten al alumno pensar libremente e investigar, consiguiendo un aprendizaje continuo. Esa libertad es precisamente la que es posible fomentar gracias a las nuevas tecnologías, que aportan a la enseñanza cognitiva:

- Apoyo del aprendizaje significativo, construyendo los nuevos conocimientos a partir de los ya adquiridos anteriormente a través de procesos que pongan a prueba sus progresos y favorezcan su creatividad.
- Aprendizaje individualizado, permitiendo adaptar el ritmo de adquisición de conocimientos al adecuado para cada individuo, atendiendo a sus necesidades según sus limitaciones físicas, psíquicas, de tiempo o espacio.
- Herramienta mixta que conjuga su funcionamiento como fuente de información y como herramienta misma de trabajo y creación.
- Creación de un entorno más atractivo para el alumno, favoreciendo así la motivación por el aprendizaje.
- Permitir la retroalimentación al estudiante para que éste mismo sea consciente de sus avances y de sus puntos fuertes y débiles.
- Permitir al profesor modificar los contenidos que considere oportunos para así hacer más énfasis en la idea central que desea hacer llegar a sus pupilos.
- Lograr la recreación y la representación de situaciones mediante simulaciones y modelos que, de otra manera, sería imposible observar.

El constructivismo puede dar nuevos métodos de trabajo más afines al objetivo planteado. Pensemos que es el constructivismo social el que considera necesario que los alumnos trabajen en grupo, de manera que puedan contrastar sus ideas y los resultados obtenidos, debatiendo y profundizando así más en la materia, intercambiando los distintos conocimientos que cada uno interioriza individualmente. Así, el profesor adopta el papel de guía, realizando una presentación inicial de la información y encaminando a los alumnos durante el proceso de aprendizaje, evaluando y corrigiendo los conceptos aprendidos por cada alumno.

Este modelo pedagógico es muy fácilmente adaptable en el aula con las tecnologías multimedia, pues históricamente han ofrecido enormes posibilidades en el mundo de la docencia dentro de la concepción constructivista y cognitiva del aprendizaje, donde el auténtico protagonista de la enseñanza es el propio alumno en interacción con sus compañeros y el medio. Un ejemplo es el proyecto de investigación presentado en [13] y llevado a cabo por el Instituto de Ciencias de la Educación de la Universidad de Friburgo consistente en un entorno de aprendizaje multimedia. En él, el

objetivo era que unos alumnos de una escuela de bachillerato se familiarizaran con los ciclos económicos. La figura de un experto se encargaba de presentar un modelo de los ciclos monetarios explicando el proceso mediante ejemplos y comparándolo con modelos análogos, como el modelo del ciclo del agua. A continuación los alumnos libremente recopilaban información sobre los problemas, podían escoger entre resolverlos, escuchar la opinión de los expertos, adoptar modelos de estructuras, observar nuevos ejemplos o seguir animaciones. Finalmente los alumnos presentaban conjuntamente sus soluciones a los problemas y debatían sobre todas las alternativas posibles. Los resultados fueron altamente satisfactorios, y todos estos procesos fueron llevados a cabo en el aula con la ayuda de un ordenador y gracias a sus posibilidades multimedia.

Esta potencialidad formativa viene dada principalmente por dos factores clave para toda transmisión de información presente en el proceso educativo, y que Insa Ghisaura destaca en [17]:

- La redundancia de información.
- Complementariedad lingüística al integrar lenguajes verbales y visuales.

Con anterioridad en el tiempo, Seymour Papert, discípulo de Piaget en su Centro Internacional de Epistemología Genética de Ginebra entre 1959 y 1964, y posteriormente colaborador en el MIT de Marvin Minsky en el campo de la inteligencia artificial, creó el lenguaje LOGO, que supuso un cambio sustancial en la escuela, un cambio en los objetivos escolares acorde con el elemento innovador que supone el empleo del ordenador.

El lenguaje LOGO es el primer lenguaje de programación diseñado y destinado para niños [32]. Emplea instrucciones muy sencillas cuyo fin es desplazar una tortuga por la pantalla, pudiendo así construir, a través de sus movimientos, cualquier figura geométrica. Aunque su fin básico es que los alumnos lleguen a dominar los conceptos básicos de geometría, tal y como Crevier (1996, 86) afirma, detrás de ello existe una “herramienta pedagógica mucho más poderosa”, fundamento de todo aprendizaje: el aprendizaje por descubrimiento.

Partiendo de los postulados de Piaget, Papert los replantea desde una perspectiva “más intervencionista” (en sus propias palabras), incidiendo principalmente en el desarrollo de las estructuras mentales potenciales y los ambientes de aprendizaje. (Papert, 1987)

Para ello, Papert se sirvió del lenguaje LOGO, pues entendía que la programación favorece las actividades metacognitivas, pues el niño, mediante ella, puede pensar sobre sus propios procesos cognitivos, sobre sus errores y aprovecharlos para reformular sus programas. De esta manera, y tal y como apunta Martí (1992), Papert pone en juego los siguientes postulados de Piaget:

- La necesidad de un análisis genético del contenido.
- La defensa de la visión constructivista del conocimiento.
- La defensa del aprendizaje espontáneo y, por tanto, sin instrucción.

- El sujeto es un ser activo que construye sus teorías sobre la realidad interactuando con ésta.

De esta manera, Papert creía en el ordenador como una herramienta, tan necesaria y funcional para llevar a cabo sus proyectos como un lápiz; y se erigió como auténtico valedor del uso de la informática en el aula, tal y como afirman sus siguientes palabras (Papert, 1995, 72):

“La medicina ha cambiado al hacerse cada vez más técnica; en educación el cambio vendrá por la utilización de medios técnicos capaces de eliminar la naturaleza técnica del aprendizaje escolar.”

Ésta es una manifestación perfecta de la corriente pedagógica constructivista, de la que Papert fue firme defensor a lo largo de su trayectoria profesional. Tanto es así que, en el MIT, fundó el grupo “Epistemology and learning group”, desarrollando la teoría del construccionismo a partir del constructivismo, y cuyo trabajo es definido en [33] como:

“Diseñamos el construccionismo como una teoría de aprendizaje y educación. El construccionismo se basa en la idea de que la gente aprende mediante la activa construcción de nuevos conocimientos en lugar de “verter” información en sus cerebros”.

Capítulo 3:

Estado del arte

3.1. Wargames	39
3.2. El empleo de videojuegos en el aula	43
3.3. Videojuegos de recreación histórica	47
3.4. Serious Games	59
3.5. Objetivos cognitivistas de SimuBattle según aplicaciones anteriores	64

3.1. Wargames

Un wargame es un juego que simula y recrea un conflicto bélico mediante el uso de una serie de normas que controlan el funcionamiento del mismo con el fin de determinar el vencedor de la contienda. Así, ha de quedar claro que es un juego y no una recreación fidedigna de hechos históricos, pues el fin es dar al jugador la posibilidad de cambiar el resultado de la batalla recreada mediante el uso de tácticas a través de su propio ingenio y, cómo no, del inevitable azar.

Acudiendo a la historia de estos juegos de guerra, recogida en [22], los wargames modernos se originaron inicialmente ante la necesidad militar de estudiar las tácticas y de reconstruirlas con fin instructivo. La victoria del ejército de Prusia sobre el Imperio Francés en la guerra Franco-Prusiana (1870-1871) se debió en parte gracias al entrenamiento que los oficiales prusianos disfrutaron a través del juego Kiregspiel que fue inventado en torno a 1811 y que ganó rápidamente popularidad entre ellos. Estos wargames ya empleaban dados para recrear los cambios azarosos que podían darse durante una batalla real: moral de las tropas, meteorología... aunque en ocasiones eran reemplazados por un árbitro que usaba su propia experiencia en el combate para determinar los resultados.

La primera utilización de un wargame para fines no militares fue en Oxford en el siglo XIX por parte del aficionado a todo lo relacionado con el mundo naval Fred T. Jane, que creó una serie de reglas para determinar las acciones navales empleando miniaturas de barcos alrededor de 1898. En 1905 la edición de "*Jane's Fighting Ships*" incluía una edición revisada para "*The Naval War Game*".

Los libros "*Floor Games*" (1911) y "*Little Wars*" (1913) de H.G. Wells incluían reglas para enfrentar a soldados de miniatura en batallas haciéndolas accesibles al público general. Ya en 1940 se publicó "*Fletcher Pratt's Naval War Game*". Éste presentaba un sistema más arbitrario que el de Jane, pero que generalmente daba resultados más realistas, y muchos clubs empezaron a jugar a él. Pratt invitó a Jack Coggins a participar en batallas donde los resultados de las mismas eran calculados a través de complejas fórmulas matemáticas. Como se ha dicho, muchas de las normas eran relativamente arbitrarias, pero estaban profundamente basadas en los conocimientos de estrategia naval de Pratt, por lo que aportaban unos resultados increíblemente cercanos a la realidad.

Todos estos juegos nacieron con el propósito de ser accesibles al público general, pero esto no fue posible debido al elevado precio que alcanzaban las miniaturas de guerra. Pero este adictivo hobby ya había sido creado y en 1955 Jack Scruby comenzó a producir miniaturas reduciendo los costes, lo que provocó una mayor expansión de este tipo de juegos, proliferando los entusiastas de los wargames.

El primer wargame moderno basado en cartas, dados y mapas fue diseñado y publicado por Charls S. Roberts en 1952, fundando la compañía Avalon Hill Game destinada a la creación de juegos para adultos. Seguidamente, en 1964 Avalon Hill sacó al mercado la publicación "*The General Magazine*", acercando esta afición al gran público. Antes, en 1961 se

creó “*Robert’s Gettysburg*”, el que es considerado hoy por hoy el primero wargame de tablero basado enteramente en una batalla histórica. Ese mismo año se publicaron “*D-Day*” y “*Chancellorsville*”, que fueron los primeros juegos en emplear un mapa hexagonal.

El creciente negocio hizo florecer nuevas compañías en el sector, convirtiendo en muy populares a los wargames en la década de los 70, convirtiéndola en la edad de oro de los wargames. En paralelo a este fuerte crecimiento, surgieron los primeros juegos de miniaturas basados en mundos fantásticos, lo que acabaría derivando en el nacimiento de los juegos de rol.

A partir de los 80, los hábitos de vida del gran público hicieron disminuir el entusiasmo por este tipo de juegos, y ya, posteriormente, con la proliferación de las nuevas tecnologías, los wargames pasaron a aplicarse a los ordenadores personales. Estos ofrecían mayor simplicidad a la hora de jugar, ya que permitían jugar directamente sin tener que memorizar complicadas reglas, reduciendo el espacio de juego necesario (los mapas de los wargames clásicos ocupan mucho sitio), reducen los gastos enormemente al no tener que adquirir costosas miniaturas, y permite gracias a internet encontrar contrincante en cualquier lugar del mundo o incluso jugar contra la propia máquina.

Se puede diferenciar de manera general, según el nivel del conflicto, entre wargames estratégicos y wargames tácticos. Los primeros recrean grandes conflictos y los jugadores tienen bajo su mando un importante número de fuerzas: ejércitos, divisiones... y controlan además aspectos políticos y económicos. En cambio, los tácticos se centran en escenarios más pequeños con un menor número de unidades controladas.

Como ya se ha explicado, lo que hace complejo a un wargame es su cúmulo de reglas y nada tiene que ver con el nivel de representación. Hay juegos de guerra que tienen un nivel de detalle excesivo centrándose hasta en el más mínimo aspecto, y otros que de una manera más general recrean el espíritu del conflicto captando su esencia.

Según los elementos que emplean para el desarrollo del juego, se pueden distinguir los siguientes tipos de wargames, clasificación que puede encontrarse en [23]:

- De tablero: El juego se desarrolla en un tablero-mapa que representa a través de una vista aérea el conflicto. Generalmente presenta una retícula hexagonal (figura geométrica capaz de ocupar de manera continua todo el espacio ocupando a su vez el menor área posible, de ahí su utilización por parte de las abejas en sus panales de miel), que sirve para delimitar los movimientos. Las fichas empleadas suelen ser sencillas figuras de cartón. También es común la utilización de cartas a modo de elemento de apoyo, al igual que el uso de dados, empleados mayoritariamente para la representación azarosa de la contienda. Ejemplos de este tipo de wargames son: “*World in Flames*” o “*Third Reich*” (estratégicos) y “*Advanced Squad Leader*” (táctico).

- Con figuras: Se utilizan miniaturas de metal, madera, plomo o plástico, representando principalmente soldados o incluso unidades de guerra. Se juega sobre una mesa mediante mapas a escala relativamente acordes al tamaño de las figuras. También se utilizan dados e incluso cintas métricas para medir el alcance de tiro, el avance de las tropas... El nivel de representación queda reducido a grandes batallas a lo sumo, nunca una guerra completa, pues cada jugador controla un número muy reducido de unidades. A este tipo de juegos de guerra suelen pertenecer los wargame tácticos anteriormente mencionados. Ejemplos de wargames con figuras son: “*Command Decision*” y “*Warhammer*”.
- Ordenador: Gracias a la gran capacidad de los ordenadores, se presentan wargames tanto tácticos como estratégicos. Algunos son meras recreaciones de los clásicos juegos de tablero, pero otros utilizan conceptos propios como la acción en tiempo real. Un tipo de juegos como “*Civilization*” no son considerados puros wargames, pues no priman el conflicto bélico en sí, sino los mecanismos de producción y control. Algunos wargames de ordenador clásicos son: “*Panzer General*” o “*Steel Panthers*”.

Las batallas representadas son muy diversas, aunque las más populares mundialmente son: batallas de la antigüedad clásica (desde las primeras civilizaciones hasta la Edad Media), el Renacimiento, Guerra Civil Inglesa (impulsada su afición por los jugadores ingleses, de los primeros en popularizar los wargames), Guerra de los 7 años, Guerra de Independencia Americana (popularizado por el gran poder de la cultura estadounidense), periodo Napoleónico (sin lugar a duda el más popular junto a la Segunda Guerra Mundial), Guerra Civil Americana, batallas coloniales (principalmente de Gran Bretaña), Primera Guerra Mundial, Segunda Guerra Mundial y Guerras Modernas como Vietnam o Corea (con una implantación menor que otros periodos).

Otra clasificación de los wargames podría ser según dónde se desarrolle la acción de la batalla:

- Tierra: Recrean las batallas terrestres y son los mayoritarios así como los primeros en inventarse.
- Mar: Las batallas navales son también unas de las primeras en recrearse en los wargames de la mano de Fred T. Jane primeramente y Fletcher Pratt con posterioridad.
- Aire: Dado que los combates aéreos son muy recientes (II Guerra Mundial), hay relativamente pocos wargames que se centran en este tipo de batallas, siendo obviamente los pocos que existen de tipo táctico.
- Combinación de armamentos: Es una mezcla de todos los anteriores, lo que repercute en wargames con reglamentos relativamente complejos dada su gran variedad. Este amplio espectro de elementos a emplear durante el juego hace que sean de tipo estratégicos dado lo completos que son.

Como ya se ha dicho con anterioridad, los wargames han dado el salto de los tableros de mesa a los ordenadores personales, permitiendo encontrar oponentes para las batallas en cualquier lugar del mundo a través de Internet e incluso, gracias a la inteligencia artificial de las computadoras, tener como contrincante al propio ordenador. A los juegos de guerra se ha añadido además la posibilidad de jugar en tiempo real, en detrimento de los turnos entre jugadores de los wargames clásicos. Además, se ha posibilitado realizar juegos estratégicos mucho más completos sin que por ello se conviertan en más complejos. Inicialmente, las nuevas tecnologías ofertadas por los ordenadores fueron aplicadas a los juegos clásicos de guerra para poder comunicar los movimientos y estrategias a sus oponentes. Gracias a internet, una persona en un punto del planeta podía enviar vía e-mail los movimientos al igual que en una partida de ajedrez a distancia. Posteriormente la inclusión de archivos adjuntos y la proliferación de wargames electrónicos permitió enviarse entre oponentes una partida grabada tras realizar cada uno sus movimientos. Otra posibilidad es la de jugar una partida clásica remotamente, donde los ordenadores actúan como meros tableros con piezas, siendo los jugadores de uno y otro extremo los que realizan los movimientos sin necesidad de que el programa que corre en ambos lados conozca con detalle las reglas del juego que, al fin y al cabo, son las fijadas por los oponentes humanos.

3.2. El empleo de videojuegos en el aula

Hoy día en las aulas comienza a ser relativamente habitual la utilización de recursos multimedia para complementar las explicaciones habituales y clásicas de los docentes. Sería incluso posible el empleo de programas no destinados en principio para la docencia, sino con un importante carácter lúdico como son los videojuegos de estrategia ambientados en guerras históricas.

Analizar los objetivos que persigue el videojuego puede resultar provechoso para su uso en el aula. El hecho de que estos videojuegos de contenido histórico se basan en la consecución de unos objetivos (conquista, fin de una batalla...), permite a los docentes aprovecharlo para programar una serie de actividades vinculadas al uso de un videojuego, introduciendo así su empleo en las aulas. María del Carmen Gálvez de la Cuesta (coordinadora de Proyectos del Centro Nacional de Información y Comunicación Educativa del MEC), en [14], propone las siguientes estrategias docentes a emplear con este tipo de videojuegos:

- *“Consecución de objetivos básicos: Tras establecer un objetivo básico, que no conduzca a la finalización del proceso lúdico, sino a lograr un objetivo que dé paso a otra fase, se solicitará del alumno el reconocimiento de determinados conceptos, tanto geográficos como históricos. Estos conocimientos podrán comprender espacios geográficos de desarrollo de la acción, ubicación de lugares, ciudades o poblaciones e identificación de personajes históricos.*
- *Consecución de objetivos intermedios: Por definición, los videojuegos ofrecen objetivos intermedios, que permiten progresar en la estructura del entorno audiovisual de tal forma que el usuario adquiere un conocimiento exhaustivo de los principales elementos que lo componen, alcanza situaciones de responsabilidad sobre los personajes y ha de proceder a la ampliación de las dotaciones, suministros o bienes que poseía al principio de la partida para poder continuar. En esta fase el alumno podrá alcanzar un mayor entendimiento del sistema organizativo del modelo a través del que se desarrolla el videojuego (ejército, civilizaciones, estados...), por lo que será posible que defina las principales estructuras jerárquicas que se representan. De igual forma podrá establecer relaciones de semejanza o antítesis entre los personajes destacados (héroes, generales, nobles, monarcas, jefes de gobierno...), y a su vez referenciar los principales rasgos de las actividades socioeconómicas que se detallen (abastecimientos, compras, saqueos...).*
- *Consecución de objetivos finales: La representación del hecho histórico en torno al que se desarrolla la acción requiere del usuario un nivel de comprensión elevado de la estructura general del videojuego, por lo que el alumno habrá podido ejecutar un porcentaje muy elevado de acciones. A través de este objetivo final, habrá adquirido un conocimiento exhaustivo del sistema organizativo, de las características de la civilización o pueblo, y en especial una perspectiva secuencial de los acontecimientos que le han dirigido a lograr la victoria o premio final. Es desde este punto a partir del cual se interrelacionará el proceso histórico real con los hechos plasmados en el videojuego. Desde los conocimientos adquiridos por el alumno de forma no*

proactiva, sino a través del juego y con una finalidad distinta al aprendizaje, el docente iniciará una fase de percepción comparativa, que logre concretar un esquema lógico del hecho histórico objeto del videojuego. A través de ese esquema es posible completar el conocimiento a través de otros instrumentos o estrategias, entre las que pueden incluirse la puesta en común de los procesos de juego con el resto del grupo de compañeros o la aportación de datos complementarios por parte del profesor.”

María del Carmen Gálvez de la Cuesta advierte del cuidado en el uso de estos videojuegos de carácter histórico ante la posibilidad de que el juego derive en situaciones diferentes a las ocurridas verdaderamente, produciéndose ucronías. En muchos casos estas ucronías que muestran la culminación de un hecho real con un acontecimiento inesperado cambiando el proceso real de la historia son deseadas para mostrar al alumno la importancia de un hecho concreto acaecido. En muchos otros el docente ha de controlar y explicar adecuadamente, cuáles fueron los hechos reales que se sucedieron en el pasado para no confundirlos con los derivados de una táctica seguida durante la marcha del videojuego.

El alumno, se aproxima a los videojuegos no con un deseo educativo, sino meramente lúdico, por tanto es siempre necesaria la intervención activa del profesor, orientando el juego y enfocando las actividades propias del mismo a fin de que el alumno sea consciente de lo que está aprendiendo.

Desde el curso 1992/93, viene funcionando en Cataluña el grupo de trabajo F9, dedicado al aprovechamiento didáctico de los juegos de ordenador en la escuela primaria y en ESO. Formado por 8 profesores de escuelas e institutos del Vallès y del Maresme, y asesorado por Begoña Gros Salva, profesora de la Facultad de Pedagogía de la Universidad de Barcelona, ha estudiado la utilización de los videojuegos por parte de los alumnos en el aula, tutelados y guiados por el profesor, obteniendo unos excelentes resultados. En [19], encontramos los objetivos alcanzados por los alumnos con este proyecto:

- *“Ser capaz de construir activamente el conocimiento en lugar de recibirlo elaborado.*
- *Ser capaz de decidir el camino más adecuado en la resolución de una situación, teniendo en cuenta las relaciones causa-efecto.*
- *Tener la necesidad de tomar decisiones a fin de ejercitar de forma directa la elaboración de estrategias cognitivas.*
- *Desarrollar la capacidad de síntesis a través de los juegos a fin de tener más facilidad a la hora de entender fenómenos complejos.*
- *Valorar los diferentes aspectos de la realidad de una forma más global e interdisciplinar.*
- *Valorar la interacción entre otros alumnos, pues sabe que le puede servir en la consecución de mejores resultados a fin de aumentar su capacidad de diálogo y argumentación.*
- *Favorecer la estructura de los contenidos debido a la necesidad de memorización de procesos, acciones, reglas...”*

En resumen, la utilización de sistemas multimedia, incluyendo en esta categoría a los videojuegos, como medios de aprendizaje en distintos ámbitos de la docencia, presenta las siguientes ventajas:

- La utilización de distintos canales comunicativos (texto, imagen, sonido) conduce a la redundancia y a la complementariedad lingüística, dotando así al alumno de un amplio abanico de posibilidades a la hora captar los conocimientos.
- Un aprendizaje más lúdico y atractivo para los estudiantes al emplear distintos medios y más modernos para hacer llegar la información deseada.
- Flexibilidad y adaptación a la velocidad de aprendizaje de cada alumno, convirtiéndolo así en un aprendizaje individualizado. Además, este aprendizaje es adaptado a las necesidades y deseos de cada individuo permitiendo adquirir los conceptos en el orden preferido por el alumno y haciendo hincapié en aquellos aspectos deseados por el docente en cada momento.
- Se logra un aprendizaje mucho más interactivo. Esta interactividad consiste en recrear situaciones que favorezcan la creatividad del alumno y logrando el docente un alto nivel de motivación en su alumnado creando él mismo aplicaciones y actividades, mucho más completas que los recursos docentes clásicos, en las que los alumnos ponen a prueba sus propios progresos en el aprendizaje.
- Si además de multimedia estos sistemas incluyen también características hipermedia (algo muy común), la información se presenta de una manera no secuencial, no hay un orden fijo y único preestablecido de acceder a ella. Esto permite una mejor adaptación a la manera en que funciona la mente humana, pues los estudios realizados acerca de su funcionamiento afirman que el aprendizaje en los seres humanos se produce sobre construcciones conceptuales basadas en datos y las relaciones entre los mismos. Por tanto, la organización no secuencial de la información presente en las aplicaciones multimedia, permite contrastar rápidamente conceptos y las relaciones entre ellos, permitiendo un aprendizaje más rápido, pero sobre todo, más eficaz.
- Los sistemas multimedia son capaces de presentar y almacenar una ingente cantidad de información y, hoy día gracias a su alojamiento en sitios web, se hace accesible a todos en cualquier lugar y en cualquier momento. Así se logra una globalización del conocimiento y permite contrastar diversos puntos de vista ante todo tema a asimilar, permitiendo así una estimulación de la investigación, el debate y la creatividad de todo individuo.

El mayor inconveniente presente en la utilización de estas nuevas tecnologías en el mundo educativo y formativo es su empleo como mero factor de modernidad sin tener clara su adecuación y buen uso en cada momento dependiendo de las características propias de cada proceso de aprendizaje. Es necesaria una buena gestión de las nuevas tecnologías en el aula, siendo siempre recomendable emplearlas integrándolas en un programa educativo completo y bien fundamentado para conseguir que sean un complemento necesario al proceso de enseñanza y aprendizaje y nunca un lastre o una carga más para el profesor e incluso para el propio alumno.

En la misma línea, uno de los mayores problemas es utilizar las nuevas tecnologías para reproducir los tradicionales modelos de enseñanza. Esto no es más que una manera de desaprovechar el inmenso abanico de posibilidades y de no lograr la educación individualizada y adaptada al propio ritmo del estudiante. Emplear un esquema clásico de estímulo-respuesta completamente secuencial es no realizar una auténtica aplicación multimedia didáctica.

3.3. Videojuegos de recreación histórica

Los lazos entre videojuegos y educación son cada vez más fuertes, surgiendo en los últimos tiempos diversas iniciativas a favor de educar y entretener a la vez mediante el ocio electrónico. En el mercado podemos encontrar una gran variedad de videojuegos que se basan en la recreación histórica. He aquí un breve muestrario de ellos, que permitirá hacernos una idea de las posibilidades que nos presentan, tanto de ocio como de posible uso con fines educativos.

Imperium Civitas II

Un claro ejemplo de videojuego fácilmente utilizable en el aula con fines didácticos, es Imperium Civitas II, un juego de estrategia para PC del tipo “city-builder” básicamente, aunque también con un importante contenido de recreación de guerras y batallas, diseñado por la compañía italo-española FX Interactive, y que ha contado con el apoyo del Parlamento Europeo, el Ayuntamiento de Roma y la Universidad Complutense de Madrid. En este juego el usuario toma el rol de un gobernador romano cuyo objetivo es lograr la prosperidad de su ciudad, gestionando los recursos a su alcance de una manera eficaz y eficiente, actuando sobre el comercio de la zona y protegiendo a sus conciudadanos de las agresiones externas, también se ha de actuar sobre la seguridad ciudadana intentando mantener la paz social interna mediante un abastecimiento adecuado. El juego hace especial hincapié en las materias primas dominantes de la época: oro, hierro, mármol..., la importancia de la mano de obra de los esclavos, alimentación para los habitantes: trigo, pan, carne, aceite, vino..., e incluso el tipo de edificios típicos de la época como el foro, calzadas, templos, tabernas, comercios, campos de cultivo, termas, academias, arcos del triunfo, coliseos, circos... lo que permite dotar al juego de un alto cariz educativo al tener estos edificios asignados unas funciones muy concretas y equilibradas, consiguiendo comprender exhaustivamente su utilidad en aquella época.



Figura 3.1. Juego Imperium Civitas II

Como la mayoría de city-builders, un objetivo claro es el reparto por todo el territorio de los recursos necesarios para la supervivencia, para ello se utilizan “áreas de influencia”, que suelen consistir en circunferencias

alrededor de las construcciones para conocer su radio de alcance. Con el fin de hacer más verídica la acción de este videojuego, esto ha sido sustituido por lugares donde se pueden escuchar las quejas de los habitantes en forma de rumores en tabernas o ruegos en los templos.

Como se ha dicho, este juego aporta además la necesidad de proteger la ciudad de ataques bárbaros, por lo que se tiene a su disposición tres tipos de unidades de batalla: guerreros a pie (hastati), guerreros a caballo (équites) y arqueros. Éste es un juego del tipo city-builders y no un wargame al uso, pues estas unidades de batalla no son controladas por entero por el jugador capitaneando a los ejércitos en diferentes guerras, sino que se han de producir unidades gestionando los recursos de la ciudad para intentar lograr el equilibrio saciando todas las necesidades de los habitantes, incluida la seguridad ciudadana.

Hay que destacar el ambiente recreado a través de sonidos muy reales de la propia vida de la ciudad, melodías que acompañan al jugador y que lo sumergen en la historia y sus gráficos con una calidad muy aceptable, resaltando principalmente el nivel de detalle de los principales edificios de la ciudad.

Medieval 2: Total War

En el modo campaña se presenta un mapa bidimensional con Europa y parte de Asia y África por el que poder mover las tropas a cargo de cada usuario así como gestionar las poblaciones, incluyendo el adiestramiento de tropas y la construcción de edificios. En el momento del combate, el mapa desaparece y se da paso a una visión detallada en tres dimensiones del campo de batalla, permitiendo mover la cámara y seguir la acción con todo lujo de detalles. Se incluye, al margen del juego por turnos, la posibilidad del modo multijugador para poder competir contra oponentes de todo el mundo mediante el servicio Gamespy.

El juego introduce diversos personajes que convierten el juego en más realista y logran crear la atmósfera perfecta para recrear la situación histórica de la Edad Media: princesas que ejercen función diplomática e incluso con la posibilidad de auspiciar matrimonios concertados para mejorar las relaciones entre países, diplomáticos que ejercen tareas de negociación, mercaderes que crean nexos de unión entre pueblos exportando materias primas al extranjero, e incluso los personajes de Papa al que seguir en cruzadas so pena de excomulgación en caso de contradecirle, obispos con los que convertir a los pueblos paganos y mejorar los índices religiosos... El juego permite además escoger entre diversos bandos en las cruzadas a la hora de jugar: católicos, ortodoxos, islámicos...

Al igual que muchos juegos, entre batalla y batalla tridimensional, se trabaja sobre un mapa-tablero en dos dimensiones que presenta, eso sí, numerosas y complejas posibilidades para dotar al juego de las características estratégicas deseadas con apasionantes subtramas de traiciones e intrigas. Cabe destacar que el juego también presenta la posibilidad de recrear batallas navales, pero se resuelven de forma automática sin la intervención plena del usuario. También, una de las mayores quejas por parte de los usuarios es la

“estática” inteligencia artificial del juego, que no llega a ser lo deseablemente impredecible.

El diseño gráfico es nuevamente uno de los factores clave para esta nueva hornada de videojuegos estratégicos, donde la acción presenta un cuidado aspecto cinematográfico con numerosos detalles en los soldados, los entornos naturales (bosques, vegetación, agua) y la presencia de factores climatológicos clave a la hora de estudiar una batalla con detalle: niebla, lluvia, diferencia entre noche y día...

Sombras de Guerra: La Guerra Civil Española

En noviembre de 2007, salió a la venta en España este videojuego de estrategia basado, por primera vez, en la Guerra Civil Española. Este lanzamiento provocó un inusitado revuelo en los medios de comunicación ajenos al mundo de los videojuegos por lo sensible del tema, y es que los medios especializados prácticamente no han prestado interés por este juego de estrategia en tiempo real debido a que lo califican de “anacrónico, sin profundidad táctica o estratégica, mapas confusos, inteligencia artificial torpe, gráficos pobres y desproporcionados...”.

Aún así este título sirve de claro ejemplo de la importancia que los videojuegos de estrategia están tomando en nuestro país, así como del amplio abanico de posibilidades que presentan para recrear la historia, incluso la más reciente y polémica y no sólo la historia clásica de Roma de la que el mercado de los videojuegos está especialmente inundada.

Este título cuenta con tres modos de juego: campaña, escaramuza y multijugador. El primero de ellos narra la historia desde el punto de vista de ambos bandos (nacional y republicano). La narración se divide en episodios que pueden ser, por tanto, visualizados desde distintas perspectivas, lo cual ayuda a comprender mejor el enfrentamiento.



Figura 3.2. Sombras de guerra

El juego cuenta cuatro niveles de dificultad (principiante, normal, difícil y experto) que varían no en las tácticas del rival, sino en la resistencia del mismo; y con una gran variedad de unidades de guerra y personajes (médicos, conductores, pilotos...) que representan los 5 tipos básicos de facciones durante la guerra civil española (bando nacional, republicano, soviético, Legión Cóndor y brigadistas internacionales en forma de tropas voluntarias italianas).

El modo multijugador permite hacer combatir hasta a 14 jugadores simultáneamente en red o a través de Internet. Este modo, a su vez, incluye tres tipos de juego: Melee, Matar al General y Victoria Total.

El juego recrea batallas y puntos históricos importantes de la contienda, como el levantamiento de Melilla, la batalla del Ebro, el bombardeo de Gernika o el asalto al Alcázar de Toledo. Además, antes de las batallas la ambientación se acompaña de vídeos con imágenes de archivo. Aunque después, durante el juego, se echa en falta una mayor rigurosidad histórica evitando tópicos y diversas descontextualizaciones. Esta crítica es una de las que más han pesado sobre el juego desarrollado por Legend Estudios, lo que nos aporta una idea de la importancia actual al carácter didáctico que se le da a los videojuegos actuales por parte, ya no sólo de los padres, sino también del usuario medio, que busca en esta especie de wargames un juego ameno, entretenido y que a la vez desarrolle la capacidad estratégica aprendiendo a la vez, en este caso, historia.

Company of Heroes

Este juego es una de las últimas novedades en videojuegos basados en la II Guerra Mundial. Plantea la posibilidad al usuario de meterse en la piel del ejército aliado y liberar Europa del fascismo tras desembarcar en la playa de Omaha. Unidades de infantería, aire y blindaje, armas fijas, carros de combate... cada uno con sus propias características a desplegar en el momento oportuno de cada batalla para lograr vencer. El juego tiene un marcado y profundo carácter estratégico, aportando la novedad de líneas de transporte de recursos que hacen el juego más real y no sólo centrado en las batallas, pero eso sí, sin dejar atrás la acción.

Este videojuego posee una inteligencia artificial muy desarrollada que se aprecia en la forma de actuar de los personajes durante las batallas, adaptándose a los terrenos físicos o edificaciones que los rodean, y obedeciendo las órdenes que se le dan de manera inteligente y respondiendo a las mismas de una manera ordenada.

El juego consta de quince misiones en las que se han de completar los objetivos marcados a través de presentaciones muy cinematográficas. También hay un modo de juego llamado "skirmish" que consiste en cumplir hasta 19 desafíos independientes distintos basados en eventos históricos, lo que le aportan, junto a la realidad de las acciones del videojuego, el carácter didáctico deseado para convertirlo en un juego muy completo. Además permite, nuevamente, el modo multijugador, en este caso para hasta 8 jugadores. Se puede escoger entre el bando aliado o decantarse por el eje y

batallar en el conocido como modo “escaramuza” consistente en eliminar al enemigo (prescindiendo del cumplimiento de objetivos del modo de juego individual). Una importantísima novedad es la inclusión de una herramienta que, a modo de editor, permite crear toda clase de mapas sobre los que jugar con posterioridad, convirtiendo al usuario en un auténtico diseñador.

Es el apartado gráfico el más desarrollado y en el que mayor novedad aporta este juego. Los gráficos, detalles, texturas, el permitir mover la cámara dónde se desee, realismo en todas las figuras (hombres, edificios, carros de combate)... En este videojuego auténticamente prima la acción y la espectacularidad, lo que puede ocasionar que no todos los ordenadores tengan una capacidad tal para hacer correr el juego sin sufrir pequeños parones en la acción del mismo.

Imperial Glory

Imperial Glory es un juego que ha sido desarrollado por el equipo español de Pyro Studios (creadores del mundialmente famoso Commandos). Ha sido creado tras dos años y medio de trabajo de un equipo de 30 personas y una inversión de cuatro millones de euros.

Es un juego que combina el modelo de gestión imperial por turnos en 2D con un entorno 3D donde se representan multitudinarias batallas, tanto navales como terrestres, en tiempo real. Se centra en las guerras napoleónicas, comenzando la historia del juego en 1789, fecha del asalto francés a la Bastilla. Así, el programa permite optar entre 5 naciones distintas: Francia, Gran Bretaña, Rusia, Austria o Prusia. El objetivo del juego es conquistar toda Europa, lo cual se aleja bastante de la rigurosidad histórica deseada para poder calificar este software de didáctico. Aún así, a pesar de estas licencias tomadas, sí logra una recreación eficaz de los modos de vida de la época, y permite gestionar un imperio a través de su política, rutas comerciales, diplomacia, construcción de edificios, comunicaciones, formación de tropas, firma de tratados...



Figura 3.3. Imperial Glory

Cuando dos países entran en guerra, el juego pasa a un modo de batallas en tiempo real, pudiendo ser estas navales o terrestres. La inteligencia artificial es aceptable y las batallas gozan de gran espectacularidad aunque no por ello sufren de complejidad excesiva. Especialmente, las batallas navales son las que mayor novedad aportan al género, teniendo en cuenta factores como la dirección del viento, tiempo de carga de los cañones, tipo de naves empleadas (corbetas, fragatas o buques de línea). Los gráficos son excepcionales en cuanto a paisajes, diversidad del terreno, vestuario y construcciones, así como la animación de las unidades.

El juego presenta cuatro modos distintos de juego, dotándole así de mucha variedad y grandes posibilidades:

- Modo historia: Jugando turno a turno se ha de cumplir el objetivo final de conquistar toda Europa (representada en los mapas por países y provincias).
- Batallas históricas: Recreación de batallas reales, permitiendo cambiar la historia auténtica. No es posible cambiar las características de los ejércitos pues el objetivo es recrear lo más exhaustivamente posible la situación real en la que se produjo la batalla, aunque el final de la misma pueda cambiarse según la estrategia que se siga durante la contienda.
- Batalla rápida: Se puede escoger entre 80 escenarios posibles (país y batalla particular). Pura estrategia en una batalla directa, prescindiendo de toda gestión estratégica previa a la contienda.
- Modo multijugador: permite hacer competir hasta a 4 jugadores en red local o a través de internet.

Age of Empires

Esta trilogía (por el momento) de juegos de estrategia, con extensiones incluidas, es quizá una de las más famosas en el panorama de los videojuegos estratégicos de guerra. Desde su primera entrega hasta la última se ha pasado por la Edad Media, la colonización americana, el oriente lejano del segundo milenio... Se han convertido en los juegos más populares debido a su ambición de lograr unos gráficos espectaculares en todas sus ediciones consiguiendo de esta manera unos resultados visuales muy detallados. Además, su jugabilidad es uno de los valores más apreciados por los usuarios, pues sus variados modos de juego al igual que sus complejas estrategias a seguir para lograr los objetivos aportan múltiples horas de entretenimiento a los usuarios.

Didácticamente hablando, esta saga de videojuegos se permite numerosas licencias históricas a la hora de plantear el desarrollo del juego, pues los creadores prefirieron primar en todo momento su carácter lúdico. Aún así es cierto que según se fueron desarrollando las últimas entregas se quiso potenciar el realismo histórico hasta el punto de que el último título publicado (Age of Empires III: The Asian Dynasties) replica modos de juego que recrean, de manera bastante cercana a la realidad, la historia de los imperios japonés e indio.

Resumen de características

Todo este tipo de juegos tienen una serie de características en común fácilmente identificables, formando así un género de videojuegos propio. A continuación se pueden ver tablas explicativas con las distintas características de los seis videojuegos que aquí han servido de muestra del amplio número de juegos basados en batallas históricas existentes en el mercado actual.

	City-builder	Gestión	Diferentes unidades	Calidad gráfica
Imperium Civitas	Sí, es la base del juego.	Gestión de materias primas y fomento tropas	Sí	Excelente. Ambientación muy lograda.
Medieval 2	Sí, es una de sus opciones	Gestión de poblaciones	Sí	Espectacular durante las batallas.
Sombras de guerra	No	No	Sí	Media-baja, pocos detalles.
Company of héroes	No	No	Sí	Gráficos espectaculares
Imperial Glory	No	No	Sí	Muy alta
Age of Empires III	No	No	Sí	Excelente

Tabla 3.1. Resumen de características de videojuegos 1

	Inteligencia artificial	Personajes	Fiabilidad histórica	Imágenes reales
Imperium Civitas	Media-baja al no haber batallas	Pocos	Muy elevada	No
Medieval 2	Buena	Muchos e importantes	Muy alta	No
Sombras de guerra	Mala	Sí, de 5 nacionalidades	Baja, durante el juego	Sí
Company of héroes	Buena	Pocos	Muy alta	No
Imperial Glory	Aceptable	Pocos	Muy alta	No
Age of Empires III	Buena	Pocos	Media-alta	No

Tabla 3.2. Resumen de características de videojuegos 2

	N° de bandos para escoger	Tiempo real ó juego por turnos	Editor de batallas (mapa y parámetros)	Tipos de juego
Imperium Civitas	1	No aplicable	Sólo cambio de dificultad	City-builder, sin batallas “jugables”. Campaña histórica, Roma o elección de una ciudad.
Medieval 2	3	Ambos	Sólo cambio de dificultad	Juego normal y multijugador
Sombras de guerra	2	Tiempo real	Sólo cambio de dificultad	Juego normal, batalla independiente y multijugador
Company of héroes	2	Tiempo real	Editor de mapas y cambio de parámetros	Juego normal, batallas reales y multijugador
Imperial Glory	5	Tiempo real	Sólo cambio de dificultad	Juego normal, batallas reales, batalla rápida y multijugador. (incluye batallas navales)
Age of Empires III	3	Tiempo real	Sólo cambio de dificultad	Juego normal, multijugador

Tabla 3.3. Resumen de características de videojuegos 3

Como podemos ver en las anteriores tablas donde se resumen las principales características de este tipo de juegos de estrategia, suelen mezclarse las batallas históricas con otra serie de elementos que complementan a la perfección su función recreativa, como es la función de city-builder. Este tipo de videojuego vivió su boom inicial con el juego Sim City y continuó su creciente éxito con la serie de los Sim, mundialmente famosos. Consiste, en su versión más básica, en la construcción de edificios y servicios para la población sobre la que se gestiona, a partir de unos recursos de los que se dispone y que se ha de saber administrar. Esta opción es muy interesante en los juegos de recreación histórica sobre todo si se piensan con un fin didáctico, pues logra en el usuario el desarrollo de la capacidad de la responsabilidad.

Aún así, en los wargames puros, lo que prima es la recreación de las batallas históricas. Una de las principales características es la presentación de diferentes unidades de batalla para lograr una mayor aproximación a la realidad de estas contiendas. Las distintas características de cada unidad es lo que las diferencia de las demás, y su comportamiento durante la batalla permite al jugador crear distintas estrategias. Es obvia, en este sentido, la completa similitud entre los wargames clásicos y su antecesor natural: el ajedrez. Las distintas unidades tienen distintos comportamientos y características durante el juego, al igual que las piezas del ajedrez cada una tiene un movimiento que las diferencia de las demás: el caballo no se mueve igual que el peón, y el rey o la reina no tienen el mismo valor material que la torre o el alfil...

Donde más hincapié hacen los actuales juegos de estrategia histórica es en su elevada calidad gráfica. En los últimos años la industria de los videojuegos ha avanzado a pasos agigantados en cuanto a calidad visual, siendo hoy día ésta una de las características más importantes para el éxito de un título, si no la más importante. En la tabla resumen efectuada para el estudio de la breve muestra de juegos de estrategia aquí estudiados, se puede observar cómo, el mayor esfuerzo de los estudios de animación se centra en la calidad gráfica de sus productos, alcanzando hoy día una calidad cinematográfica comparable a las grandes superproducciones de Hollywood. Las cifras que se manejan para la creación de estos juegos es comparable al dinero gastado en las últimas cintas de animación, y un buen número de animadores dan el salto de la pantalla de videojuego a la gran pantalla, existiendo una importante bolsa de empleo en este sector.

La inteligencia artificial de los juegos aquí presentados es también un factor clave para la llamada “jugabilidad” de los mismos. Se busca un comportamiento eficaz y lúcido del propio juego, que no existan puntos de escape o fallos en el mismo y que el desarrollo del mismo sea el más lógico y aproximado a la realidad. Una mala inteligencia artificial del juego puede echar por tierra todo el trabajo realizado de los animadores, pues se convertirá en un producto con un envoltorio espectacular pero vacío en su contenido.

La introducción de personajes a los que utilizar según sus características en cada momento de una manera estratégica, es similar a las distintas unidades de batalla explicadas con anterioridad, cual piezas de ajedrez sobre el tablero. Este caso es más característicos de los juegos de estrategia que incluyen gestión al estilo city-builder, pues el empleo de estos personajes suele ser ajeno a las propias batallas, aunque su comportamiento previo puede variar los parámetros de la batalla futura, pues sólo cobran protagonismo en los “momentos de entreguerra”.

Otra opción que incluyen la mayoría de videojuegos de este estilo es la posibilidad de escoger bando con el que recrear la batalla. Esta es una opción sencilla inicialmente de llevar a cabo y que revierte rápidamente en un gran abanico de posibilidades para el jugador al disponer del mismo juego pero participando en él desde distintos puntos de vista, pudiendo cambiar el prisma desde el que vislumbrar la época recreada.

Igualmente se ofrece distintos modos de juego, donde destacan tres principalmente:

- Modo normal: Se escoge un bando desde el que llevar a cabo la recreación total del juego, pasando “pantallas” hasta lograr un objetivo con el que finaliza del juego. Esta es la opción que más tiempo lleva al jugador, pues recrea toda una época y todas las batallas presentes en el juego.
- Modo batalla: Es un modo de juego rápido en el que se juega una única batalla donde, en muchos casos, se permite cambiar los parámetros de la misma y escoger distintas opciones para presentar una “batalla a la carta”. También es destacable que esta opción presenta en muchas ocasiones la recreación de una batalla histórica real, dotando así al juego de un marcado carácter didáctico y, su mayor innovación, cambiar el curso de la historia cambiando el resultado de la contienda dependiendo de las habilidades del jugador.

- Modo multijugador: La gran expansión de internet y la velocidad en su envío de datos, ha permitido en los últimos años lograr esta interconexión entre jugadores de distintos puntos del planeta. Este tipo de juegos, similares a las partidas de ajedrez, permiten competir fácilmente con otros jugadores, quedando la máquina como un mero tablero recreador de la contienda.

La mayoría de estos juegos son en tiempo real. En lugar de seguir el esquema de los wargames clásicos que consisten en juego por turnos, las capacidades de las nuevas tecnologías permiten recrear las batallas históricas de tal forma que los jugadores pueden realizar sus movimientos sin interrumpir el transcurso normal del juego. Con anterioridad todo este tipo de juegos habían seguido un esquema de juego por turnos similar al del ajedrez. Como se ha podido ver con anterioridad, aún este esquema está vigente en algunos de los nuevos videojuegos publicados.

Finalmente, un punto altamente importante para el fin de este estudio, es la comprobación de la libertad que el usuario final tiene para la construcción de sus propias batallas. Todos los videojuegos aquí presentados tienen en común su funcionamiento cerrado e inamovible. La mayoría de ellos únicamente permiten cambiar la dificultad del juego, que en muchos casos no repercute en el comportamiento o las tácticas del oponente, sino únicamente en la resistencia de los enemigos. Tan sólo uno de ellos introduce un editor de mapas y la posibilidad de cambiar los parámetros característicos de la contienda para crear batallas a medida del usuario.

Como muestra del cariz didáctico presente en estos juegos, basta reseñar que el videojuego Imperium Civitas II ha obtenido el premio Marco Aurelio que otorga el Ayuntamiento de Roma “por su contribución a la difusión de la cultura clásica”, premio que hasta ahora había sido siempre concedido a escritores, historiadores o cineastas. Y es que el fin de este juego es puramente recreativo, pero también aporta una alta dosis de capacidad educativa. Manuel Moreno es uno de los integrantes del equipo de FX Interactive, y en su presentación del juego ante el mismísimo Parlamento Europeo dejaba claras las intenciones de este juego de estrategia: “Además de un juego entretenido, intentamos desarrollar, con la ayuda de documentalistas e historiadores, una herramienta que pueda acompañar a los jóvenes en el aprendizaje de la historia y transmitir de forma eficaz algunos valores: el sentido de la responsabilidad, el respeto de la ley o la capacidad de gestión”. Precisamente el sentido de la responsabilidad es, según los expertos, uno de los principales valores que pueden fomentar los videojuegos. Inmaculada Marín, educadora y asesora pedagógica y directora de la consultora Marinva afirma sobre los videojuegos: “Pueden conjugar el reto, la sorpresa, incluso la transgresión, con la educación, y por eso son una herramienta poderosísima”.

Los creadores de este imperio virtual realizaron un año antes de sacar a la venta el videojuego un proyecto con estudiantes de secundaria en el Instituto Santa Eugenia de Madrid y el Liceo Highlands de Roma, donde los estudiantes probaban el juego en clases de historia. Los alumnos admitían que el videojuego les había ayudado a comprender conceptos como el sistema de población según su categoría y clases sociales, o familiarizarse con el nombre de distintos emperadores, e incluso los más escépticos afirmaban que aunque no se pueda aprender mucho del videojuego sí que ayudaba a asimilar

algunos conceptos de una manera amena y entretenida. Por otro lado, los profesores destacaban la posibilidad de que los alumnos se sumergieran dentro de la historia y les dotara de poder de decisión, mejorando la lección puramente formal que puede darse de la historia en el aula.

El grupo F9 en [20] hace un estudio de un gran número de videojuegos que pueden emplearse en el aula para fomentar diferentes capacidades del alumno. En el caso de videojuegos como los aquí presentados, propone las siguientes habilidades que pueden ser promovidas adecuadamente por el docente:

Contenido	Habilidades
De asimilación y retención de la información	Es imprescindible recoger la máxima información que vamos descubriendo en el desarrollo del juego.
De organización	Para progresar en la cavilación hay que trazar un plan y organizar los recursos que vamos obteniendo y saber utilizarlos apropiadamente.
Creativas	El alumno debe trazar un plan de actuación y plantearse más de una hipótesis de trabajo para continuar avanzando en su aventura. Después de aplicar diversas tácticas, debería llegar a normas generales de actuación, desarrollando así su razonamiento inductivo.
Analíticas	El alumno va evaluando continuamente todas las hipótesis que se plantea. Desarrollando así el razonamiento deductivo.
Para tomar decisiones	En su relación con las otras civilizaciones debe tomar decisiones, teniendo en cuenta, sobretodo, los recursos de los que dispone.
De resolución de problemas	Trabaja el razonamiento lógico y va acumulando a través de la experiencia de juego diferentes estrategias que le permiten plantearse las situaciones previendo las consecuencias.
Metacognitivas	El alumno percibe que cada situación es producto de las circunstancias anteriores y eso facilita la evaluación continua de su propia evolución para cambiar estrategias o transferir este conocimiento a otras situaciones.

Tabla 3.4. Resumen de habilidades promovidas por el tutor gracias a los videojuegos

Por su parte, Cuenca López en [21] concluye sobre la utilización de este tipo de videojuegos de estrategia basados en hechos histórico:

“Es evidente que el uso en el aula de estos juegos no está exento de dificultades, implica disponer de una serie de recursos informáticos que

posibiliten su aplicación, así como contar con alumnos interesados en participar en una propuesta didáctica que fácilmente puede motivarlos para trabajar contenidos de corte histórico. En este sentido, estos juegos se pueden articular como motivadores en los primeros pasos del proceso de enseñanza-aprendizaje, o bien, como recursos para la extracción significativa de información sociocultural durante su desarrollo. Finalmente, podrían ser empleados como sintetizadores de los aprendizajes producidos a través de la aplicación práctica, simulada virtualmente, de los procesos socialmente relevantes desarrollados en una cultura pasada o presente.”

Como se ha visto, este tipo de videojuegos pueden resultar muy útiles en la transmisión de conocimientos históricos, no sólo por los datos de fechas y sucesos que pueden encontrarse en ellos, sino también por la recreación, lo más cercana a la realidad posible, de una sociedad y unos comportamientos socioculturales que explicados de otra manera serían más difíciles de transmitir a los alumnos.

3.4. Serious Games

¿Qué es un serious game? María Sánchez Gómez, en su artículo “Buenas prácticas en la creación de serious games” [34] presentado en el IV Simposio Pluridisciplinar sobre Diseño, Evaluación y Desarrollo de Contenidos Educativos Reutilizables, celebrado en 2007 en Bilbao, nos da una definición acertada y concisa de los mismos:

“Los serious games o juegos serios son objetos y/o herramientas de aprendizaje que poseen en sí mismos y en su uso objetivos pedagógicos, didácticos, autónomos, autosuficientes y reutilizables, que posibilitan a los jugadores a obtener un conjunto de conocimientos y competencias predominantemente prácticos”.

La primera referencia a los juegos serios la encontramos en el libro de Clark C. Abt “*Serious Games*” [35] donde hacía referencia a este tipo de juegos, pero no a los actuales videojuegos con los que asociamos el término, sino a juegos de mesa clásicos con cartas y tablero. Aún así, su definición sigue resultando aplicable para el término actual de videojuegos:

“Reducido a su esencia formal, un juego es una actividad entre dos o más tomadores de decisiones independientes que buscan alcanzar sus objetivos en algún contexto limitado. Una definición más convencional podría decir que un juego es un contexto con reglas entre adversarios tratando de conseguir objetivos. Estamos hablando de juegos serios en el sentido en que esos juegos tienen un claro y explícito propósito final educativo, y no están ideados en un principio para ser jugados por mero entretenimiento.”

Por tanto, se considera serious game una aplicación de software desarrollada con tecnología y principios de diseño de videojuegos para fines no convencionales, es decir, no precisamente para entretenimiento, sino que han sido específicamente diseñados para enseñar a la gente sobre un determinado tema, ampliar conceptos, reforzar su desarrollo, entender un acontecimiento histórico o una cultura determinada o ayudarles a desarrollar una habilidad concreta.

La idea de utilizar juegos para la educación, como ya vimos con anterioridad, se remonta incluso a antes de la aparición de la informática, pero en el caso que nos ocupa, el considerado por muchos como primer Serious Game, fue “*Army Battlezone*”, un proyecto que no acabó de ver la luz del todo, dirigido por Atari en 1980, consistente en el entrenamiento militar. Y es que incluso en este, la industria militar ha sido uno de sus pioneros, aunque con posterioridad la educación, formación profesional, la sanidad, la publicidad e incluso la política pública se han interesado en la materia y han fomentado su creación. Tanto es así, que grandes empresas como American Express, IBM, Nokia, JP Morgan Chase, o el propio Departamento de Defensa de los EEUU han utilizado serious games para la formación de sus empleados.

Así pues, actualmente podemos considerar a los serious games como videojuegos con un doble objetivo: entretenimiento y aprendizaje. Son utilizados mayoritariamente con adultos para su formación en empresas, y también en el aula con alumnos, logrando así el desarrollo de habilidades y aptitudes específicas a su tarea profesional o con respecto a una materia dada

en el caso del estudiante, mediante la práctica de actividades lúdicas. Pero también, cada vez es más frecuente encontrar este tipo de videojuegos con un uso más comercial, o incluso de concienciación y/o denuncia social y política.

Un experto en la materia, Noath Falstein, afirmó durante la XMediaLab Conference celebrada en junio de 2009 [36], que el sector de los serious games puede llegar a generar durante la próxima década una cifra de negocio mayor que la que genera todo el sector del entretenimiento interactivo. Cada vez más directivos y docentes consideran que los serious games son unas poderosas herramientas de aprendizaje que permiten experimentar y aprender de los errores de una forma segura y sin correr riesgos reales. Así, su uso se está extendiendo para simulaciones militares, sanitarias, de negocio, servicios de emergencia...

Estos juegos resultan herramientas de aprendizaje muy efectivas, pues logran una interacción total del usuario, que aprende de la resolución activa de problemas, utilizando el pensamiento estratégico, la exploración y el auto-descubrimiento, desterrando la clásica memorización rutinaria.

Como se puede apreciar, los serious games abarcan un amplio abanico de usos, desde el entorno laboral hasta el docente, desde la medicina hasta los negocios, desde la publicidad hasta la denuncia social... Por ello, la siguiente clasificación acerca de los juegos serios pretende ordenar ideas y aclarar conceptos, para así conocer con más detalle y precisión los distintos tipos de juegos serios que podemos encontrar actualmente en el mercado.

A pesar de múltiples variantes en cuanto a la clasificación de los serious games, optamos por ésta presentada por los creadores del serious game "*Technocity*" Julián Álvarez y Olivier Rampnoux, del Centro Europeo para Productos Infantiles de la Universidad de Poitiers, y que podemos encontrar en [37].

- Advergaming: El término fue acuñado por Anthony Giallourakis en 2000. Son serious games destinados explícitamente a la publicidad de un producto, una compañía o incluso una idea, en definitiva, su objetivo es la difusión de una marca. Comúnmente se suelen encontrar integrados en páginas web de numerosas compañías, que se sirven de ellos como mecanismo de publicidad, para ganar visitas y, a la vez, promocionar el producto que venden o fabrican. Algunos ejemplos que podemos encontrar actualmente en la web son:
 - Night Driver: Fabricado por la marca Rexona, consiste en un clásico juego de carreras de coches a través de una carretera repleta de anuncios de productos Rexona. [38]
 - F1: De la marca de whisky Johnnie Walker, el juego consiste en una serie de preguntas con 3 opciones, en las que la temática central es la concienciación al usuario de los peligros de la conducción bajo los efectos del alcohol, promocionando así un consumo responsable de bebidas alcohólicas. [39]
 - Rompecabezas: Creado por la Junta de Andalucía con el fin de promocionar el turismo en dicha comunidad autónoma, el juego consiste en diversas imágenes de típicas estampas andaluzas y

de diversos monumentos, en forma de rompecabezas y que hay que completar. [40]

- Edumarket game: El neologismo inglés “*Edumarket game*” fue creado en 2006 por Julián Álvarez y Olivier Rampnoux, con el fin de englobar a aquellos juegos donde la intención es educar sobre un tipo de mercado, de ahí que la palabra se componga de los términos “*education*” (educación) y “*market*” (mercado). Este tipo de juegos se inscriben en amplias estrategias de comunicación y utilizan conjuntamente aspectos de los advergames y los juegos educativos, como puede ser el ejemplo del serious game “*Food Force*” lanzado por las Naciones Unidas en 2005 con la vocación de sensibilizar a los niños acerca de las misiones humanitarias contra el hambre que llevan a cabo.
- Juegos comprometidos: Son videojuegos cuya misión es concienciar a la población acerca de situaciones sociales extremas, para lograr hacer llegar la información, en muchas ocasiones, de conflictos políticamente silenciados. La empresa “Serious Games Interactive” ha elaborado dos juegos en los que el protagonista es un periodista que llega a una zona conflictiva del planeta con el fin de descubrir la situación que se vive en ese lugar y transmitirla a través de sus relatos periodísticos. Estos dos juegos, englobados en el proyecto Global Conflicts son: *Palestine* y *Latin America*. [41]
- Juegos de entretenimiento y simulación: Son aquellos que sirven para recrear situaciones que sirvan de entrenamiento y práctica para profesionales, entrenamiento, que de no ser por simulaciones virtuales, no sería posible recrear en la realidad. Son típicos ejemplos aquellos de recreaciones militares o médicas, pero también aquellos que simulan entornos económicos o de negocios, como es el caso del videojuego “Navieros”. Este juego ha sido desarrollado por la compañía española Gamelearn [42] y por la firma especializada en el desarrollo de habilidades directivas Jabary Consulting. Navieros consiste en un programa de formación en negociación y resolución de conflictos totalmente online y basado en un videojuego ambientado en la Venecia del siglo XV, donde deberán llevar a cabo el objetivo de hacer fortuna con el flete de barcos, mediante negocios bursátiles, préstamos bancarios, acuerdos comerciales y negociaciones entre empresarios.
- Edutainment o juegos educativos: Son aquellos juegos con una marcada vocación educativa. Aunque anteriormente el término “edutainment” ya había sido utilizado por la compañía Disney para una gama de sus productos y por Bob Heyman para una serie documental de la National Geographic Society, no fue utilizado con el concepto de videojuego hasta 1983 por Spectrum Microcomputers para el paquete de software Oric 1. Algunos ejemplos que podemos encontrar online son:
 - Tangram: Basado en el clásico juego de piezas de distintos tamaños y formas con los que conformar diferentes figuras, ayudando así a mejorar las habilidades en el dibujo y en la conformación de imágenes en el cerebro. [43]

- Xuegus educativos llingua asturiana: Consiste en diferentes juegos infantiles que se encuentran completamente en bable para que los niños practiquen este idioma. [44]
- Addition Attack: Juego matemático basado en el clásico arcade de destruir naves alienígenas, pero con la particularidad de que hay que disparar aquella que contenga el número que se corresponde con la resolución de una suma. [45]

Son precisamente estos últimos serious games, de tipo educativo, los que sirven de guía para la realización de SimuBattle, el espejo en el que mirarse y de los que tomar ideas para lograr el objetivo final de una aplicación interactiva lúdica y educativa de simulación de batallas, con la que aprender diversos pasajes de la historia. Pero el fin no es un software con el que los más jóvenes practiquen individualmente, sino que la interacción con sus compañeros y, sobre todo, la realimentación con el profesor, siempre tutelando el proceso de aprendizaje, son imprescindibles para una asimilación de conocimientos más completa y eficaz. Por ello, siempre teniendo en mente esta finalidad, un último ejemplo de serious game educativo empleado en el aula, y del que se han estudiado sus satisfactorios resultados, para confirmar la auténtica utilidad del uso de este tipo de juegos por parte de los docentes para así lograr el interés y la más sencilla comprensión de su materia por parte de sus alumnos.

En el documento de B. D. Coller y M. J. Scott “*Effectiveness of using a video game to teach a course in mechanical engineering*” [46] se relata cómo en la Universidad del Norte de Illinois, comenzó en 2005 a utilizarse para la asignatura de Métodos numéricos, un modelo de enseñanza basado en la utilización de un videojuego como fundamento principal del curso. Optaron por un videojuego en lugar de por otros métodos más comunes como los libros o los videos, por el factor clave de la interactividad. Diseñaron este serious game como un modelo computacional del mundo real simulado, donde las acciones de los alumnos (jugadores) afectan al devenir de dicha simulación. Y no sólo eso, sino que también los progresos del usuario son encauzados por la propia aplicación y mostrados al instante al alumno, quien observa el juego, además, como un reto a superar. Los sumerge en un mundo simulado, pudiendo recrear situaciones que podrían vivir profesionalmente con posterioridad en la vida real. Además, gozan de total libertad para experimentar y conocer de primera mano las consecuencias de las decisiones que ellos mismos toman en dicho entorno.

Así, el juego utilizado para este propósito didáctico fue el titulado NIU-Torcs, basado en juegos de carreras de coches. Y es que el fin del juego es construir y diseñar su propio vehículo, empleando para ello sus conocimientos mecánicos y matemáticos a través de programas en lenguaje C++ que ellos mismos escriben para dotar de comandos de funcionamiento a su coche: pedales de aceleración y frenado, cambio de marchas, radios de rotación de las ruedas... Tras ello, los estudiantes compilan y simulan en tiempo real su diseño, observando el comportamiento de su vehículo mediante un grafismo tridimensional, tal y como se aprecia en la figura 3.4.

Los primeros días, en un primer contacto con la aplicación, los alumnos no efectúan cálculos matemáticos de métodos numéricos, sino que se familiarizan con el programa y con la forma de hacer funcionar su vehículo. Con posterioridad, van dotando al coche de mejoras mecánicas, para las que

necesitan desarrollar los cálculos correspondientes. Así, mediante la práctica, progresan en sus conocimientos de forma paralela al perfeccionamiento de su vehículo. Este método mejora notablemente el clásico de libros de problemas, pues se observa in situ la utilidad de los cálculos que el alumno elabora, así como el correspondiente aprendizaje por ensayo y error.



Figura 3.4. NIU-Torcs

La primera medición que se realizó para comprobar el éxito de este método de enseñanza, fue calcular la media de horas dedicadas a la asignatura fuera del horario lectivo, resultando para este curso de métodos numéricos impartido a través de este juego el que más horas de estudio acumulaba en comparación con el resto de asignaturas, siendo esta diferencia muy notable, de más del doble con respecto a la media de todos los cursos. Esto da una idea del interés mostrado por los alumnos hacia esta asignatura con este nuevo método de enseñanza. Un estudio más específico fue el que se llevó a cabo para comparar este curso impartido con la ayuda de un serious game con otros cursos clásicos de métodos numéricos. Se pidió a los alumnos de diversos cursos que elaboraran un mapa conceptual con las ideas claves enseñadas en la asignatura, así como las relaciones entre ellas y la importancia de las mismas. Tras realizarse distintas medidas y comparaciones entre los mapas obtenidos en los distintos cursos de la asignatura, puede observarse en [46] que los conceptos asimilados por los alumnos en el curso llevado a cabo con este novedoso método se asemejaban más a los considerados más importantes por los profesores de la materia, llegando así a la conclusión de que se había logrado una impartición de la asignatura más amena, práctica y, sobre todo, eficaz, profundizando más en la materia.

3.5. Objetivos cognitivistas de SimuBattle según aplicaciones anteriores

Ya hemos visto sobradamente la utilidad del empleo de videojuegos y juegos serios en el aula y en el hogar para lograr impartir conocimientos por parte de los profesores y tutores. Vemos, por tanto, que nuestra idea de generar un programa para ordenador con el fin de facilitar la enseñanza de la historia tiene una base importante en anteriores proyectos educativos.

Vista la evolución de las distintas teorías pedagógicas en el último siglo, estamos ya en disposición de argumentar qué objetivos pedagógicos busca alcanzar la herramienta SimuBattle.

Tal y como hemos visto con anterioridad, el conductismo ha tenido históricamente efectos en la educación tradicional mediante el empleo de calificaciones, basadas así en motivaciones y estímulos positivos y negativos. La estructuración de los materiales de estudio de una manera secuencial también es innata a las teorías conductistas.

Esta forma pedagógica puede hacer que sea el propio alumno el que tome las riendas de su propio aprendizaje, gracias también a la organización secuencial y perfectamente estructurada de los materiales de estudio. Esta estructuración secuencial, es una de las características deseables para SimuBattle, pues el propio docente es el que puede organizar y adaptar los materiales de estudio a las necesidades de sus alumnos.

Por el contrario, el conductismo adolece de la falta de motivación hacia otros conocimientos evolucionados a partir de los impartidos. Es decir, el alumno aprenderá a realizar una serie de ejercicios y resolución de problemas básicos, sin profundizar en las bases teóricas que permitan extraer y extrapolar a otro nivel de razonamientos, trasladando esta responsabilidad por entera al alumno. De igual manera, se premia la individualidad, perdiendo las ventajas de la socialización y del razonamiento colectivo.

Pero el deseo es dar a SimuBattle unas posibilidades más acordes a nuestros días, que sobrepasen el clásico esquema tutorial, dotando al alumno de libertad para profundizar en las ideas, creando nuevas estructuras y así no actuar como mero espectador, buscando la interacción con otros alumnos y la realimentación de información con el profesor, enriqueciendo así la adquisición de conocimientos.

Vistos los empleos prácticos en el aula de las teorías constructivistas y cognitivas, veamos las características deseables para la herramienta SimuBattle:

- Total libertad del alumno para modificar los datos mostrados por el profesor en la herramienta y así investigar y descubrir distintos resultados finales.
- Posibilidad de participar activamente y jugar con la propia aplicación, logrando así captar un mayor interés por parte del alumno: enseñar deleitando.

- Posibilidad de que los resultados sean intercambiables con otros alumnos y con el propio profesor, consiguiendo así compartir información y la realimentación de la misma, redundando sobremanera en la investigación y en el debate de ideas.

Capítulo 4:

Diseño

4.1. Introducción	69
4.2. Requisitos.....	70
4.3. Planteamiento general	72
4.4. La interfaz gráfica	75
4.5. Motor de juego	88
4.6 Algoritmo de simulación	91

4.1. Introducción

Ha llegado por fin el momento de profundizar más en la aplicación SimuBattle como entidad propia y diferenciada. En este apartado se pasará a describir el proceso de diseño de la misma. Partiendo de los objetivos iniciales planteados y de los requisitos previos, se guiará al lector por las consideraciones y razonamientos que se fueron sucediendo, hasta concluir en las decisiones finales adoptadas para su consiguiente implementación. Será éste un camino, en el que comenzando desde los planteamientos generales iniciales, se pretende llegar a puntos esenciales de la aplicación para así alcanzar un conocimiento pleno de la misma.

4.2. Requisitos

Recapitulando, observemos los fines que perseguimos con SimuBattle. El propósito de esta aplicación es proporcionar una herramienta que sirva de complemento en la impartición de sus clases a los profesores de historia en colegios e institutos, específicamente en la explicación de batallas y guerras y de las causas y consecuencias geopolíticas y sociales de las mismas.

Pero aunque el fin es explicar pasajes de la historia partiendo de algo concreto y particular como es una batalla, terminando en algo más general como es el entorno histórico que la rodea, el medio es, sin embargo, muy diferente al habitual. Se busca, como objetivo clave junto a la enseñanza, el entretenimiento. Intentar captar la atención del alumno mediante un juego, pero que encierre en sí mismo mucho más que su carácter meramente lúdico. El fin es la enseñanza, en cambio el medio es el entretenimiento y el juego.

A todo esto hay que añadirle la intervención, a modo de guía, del profesor, para que tutele y dirija el proceso de aprendizaje. Por ello, la aplicación va dirigida a dos tipos claros y diferenciados de usuario: alumno y profesor. Ambos se encuentran en un mismo entorno, pero salta a la vista su disimilitud en cuanto al rol de usuario que juegan: dos perspectivas diferentes, dos objetivos diferentes (enseñar frente a aprender o más bien jugar), dos hábitos de uso del ordenador diferentes...

Así pues, con estas premisas, y teniendo siempre en mente las bases psicopedagógicas descritas en capítulos anteriores, en las que asentamos los cimientos de nuestra aplicación, con fines principalmente constructivistas, nuestra aplicación SimuBattle de recreación y simulación de batallas históricas, parte de los siguientes requisitos:

- Proporcionar un entorno en el que el profesor de historia pudiera recrear, representar y explicar con todo detalle el paso a paso de una batalla histórica, así como de sus causas y consecuencias. Esto aporta al docente la posibilidad de generar, al margen de los clásicos libros de historia, pasajes de la historia directamente creados y dirigidos por él, redundando en los beneficios que esta libertad e independencia de enseñanza.
- Mantener un enfoque claramente atrayente, intuitivo y simple. Por esta razón, es deseable que la construcción de este material didáctico en forma de recreación de batallas, se lleve a cabo de una manera totalmente gráfica la representación en el mapa de las distintas unidades de guerra, complementando la edición de explicaciones detalladas mediante un editor de textos muy similar a los ya existentes en el mercado y tan generalizados en su uso, que además incluya funcionalidades específicas adaptadas al diseño de estas batallas.
- Como ya se ha podido imaginar, se pretende crear una aplicación donde el canal del mensaje sea completamente multimedia, entremezclando imágenes, texto, audio e incluso vídeo, lográndolo hacer mucho más completo, atractivo e interesante.
- Ofrecer un alto grado de interactividad a los alumnos. La actitud de los alumnos ante la recreación de la batalla no ha de ser simplemente

pasiva mediante la visualización de la misma. Se persigue que el alumno “personalice” y “parametrice” las características de la batalla, cambiando las realizadas inicialmente por el profesor. Así, mediante su posterior simulación, puede observar las consecuencias y las diferencias existentes al cambiar las particularidades de una batalla. Con esto, el alumno puede realizar un ejercicio de razonamiento e imaginación, intentando vislumbrar de qué manera hubiera cambiado la historia si el desarrollo de esta o aquella batalla se hubiera desarrollado de una manera diferente. Un ejercicio de ucronía o creación de una historia alternativa que ayuda a la concepción de las ideas básicas de cada época y lugar, siempre con la ayuda y guía del docente.

- Además de esta intervención directa por parte del alumno, se busca una parte mucho más lúdica, permitiendo al alumno participar activamente en el desarrollo de la batalla tomando el lugar de algún bando o cualquier unidad, dirigiendo sus movimientos. Esto además permite que el estudiante “compita” contra el ordenador, o incluso contra otro compañero, intentando lograr la victoria final
- Finalmente, completar el círculo de la interactividad y de la realimentación alumnos-profesor mediante la posibilidad de que otros estudiantes y el propio docente varíen las simulaciones y las batallas creadas por otros. Consiguiéndose así generar un ejercicio colectivo de creación de una batalla, intercambiando opiniones respecto a la misma, a sus causas y sus consecuencias inmediatas.

4.3. Planteamiento general

A la hora de diseñar la estructura de creación de esta aplicación, decidimos basarnos en un Modelo Vista Controlador (MVC). Este modelo consiste en un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control o negocio en tres componentes distintos.

La principal ventaja del Modelo Vista Controlador, al separar la parte gráfica de la funcional y de los datos, es lograr una independencia tal que permita cambiar cualquiera de las tres partes de una manera sencilla sin que repercuta en el resto. En nuestro caso, nos encontramos ante una aplicación nueva, por lo que resulta obvio que podrán plantearse futuras mejoras en la misma, por lo que esta separación redundaría próximamente en la sencillez para que otras personas puedan continuar el trabajo aquí iniciado, desarrollando nuevas implementaciones y progresos.

Al margen de esta razón, otra aún más importante es, dada la funcionalidad de esta aplicación, tener completamente separada la parte gráfica del motor de juego. Con motor de juego (*“game engine”*) se entiende la serie de rutinas de programación que permiten el diseño, la creación y representación de un videojuego, es decir, la capa no visible pero que dota de funcionalidad a una aplicación. En este caso, el considerado motor de juego será el algoritmo de simulación de las batallas históricas. Al encapsular todo este funcional en un solo bloque, conseguimos que este pueda modificarse, alterarse, mejorarse en cualquier momento, sin necesidad de tocar la parte gráfica.

Es por tanto esta separación de bloques entre la parte gráfica y el motor de juego, inspirada en el Modelo Vista Controlador, la que nos marca, finalmente, la estructura de nuestra aplicación, tal y como podemos ver en el siguiente gráfico:

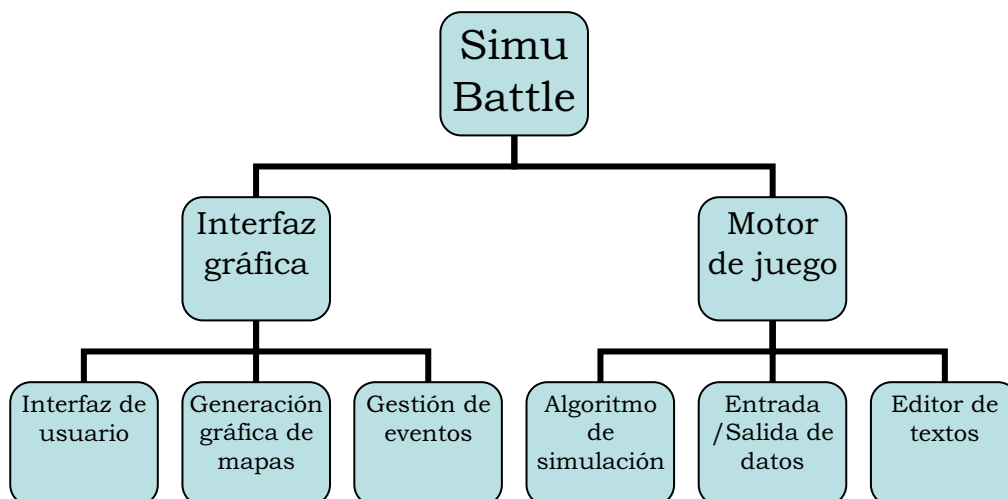


Figura 4.1. Arquitectura de SimuBattle

En él podemos observar, por un lado, que el bloque denominado Interfaz Gráfica, contiene a su vez varias subestructuras que son:

- Interfaz de usuario: Incluye todo aquello que participa directamente en la interacción con el usuario: botones, menús, cuadros de diálogo...
- Generación gráfica de los mapas de la batalla: Incluye aquellas funciones gráficas que permiten dibujar los mapas y las unidades de batalla en pantalla, así como los resúmenes de los comentarios a las batallas.
- Gestión de eventos: Se incluye en el bloque general de Interfaz Gráfica por lo sumamente ligados que van los eventos a la propia interfaz de usuario, tanto que es prácticamente imposible entenderlos por separados.

En la otra rama del gráfico, podemos observar el bloque denominado Motor de Juego, compuesto a su vez por:

- Algoritmo de simulación: Formado por todos los métodos encargados de realizar el comportamiento del programa durante la simulación, siguiendo una serie de reglas predefinidas y que componen la inteligencia del mismo.
- Entrada/Salida de datos: Incluye aquellas funciones encargadas de la lectura y escritura de datos e información en ficheros físicos para su almacenamiento persistente en disco, ya sea de manera temporal durante la ejecución del programa o de manera más duradera para su posterior utilización en un nuevo uso de la aplicación.
- Editor de textos: Aplicación que corre en paralelo a la principal y que consiste en un editor gráfico de textos para acompañar la visualización de los mapas de la batalla con comentarios explicativos.

Antes de explicar con detalle cada uno de los bloques, vamos a aclarar una serie de conceptos a modo de nomenclatura que se utilizará de ahora en adelante, y que servirán para una mejor descripción de los distintos componentes de la aplicación:

- Unidad de batalla: Cada uno de los componentes físicos de una batalla destinados a la lucha. Éstas pueden ser de distintos tipos: terrestres, navales, aéreas o misiles.

- Bando: Cada uno de los dos contendientes en una batalla, compuestos a su vez por unidades de batalla que se enfrentan durante la misma.

- Mapa: Es la unidad mínima de una batalla. Consiste en una imagen fija del terreno donde se desarrolla la batalla, dividida a su vez en celdas o casillas, a modo de escaques de un tablero de ajedrez, y que éstas a su vez pueden contener en su interior una unidad de batalla.

- Celda: Cada una de las zonas en que se divide el mapa. Puede ser de tipo terrestre o aérea.

- Batalla: Sucesión de mapas que recrean un conflicto histórico. Puede contener además comentarios explicativos en forma de textos HTML.

- Parametrizar: Asignar valores a las distintas variables asociadas a las celdas y a las unidades, y que sirven para dotarlas de diferentes características funcionales, dominando unas sobre otras a elección del usuario.

- Visualizar: Recorrer visualmente los distintos mapas de una batalla, de manera secuencial, así como los comentarios que la acompañan.

- Simular: Hacer correr el algoritmo de simulación, mediante el cual las distintas unidades interacción entre sí de manera autónoma con respecto al usuario, y comportándose de manera inteligente según la parametrización de las mismas.

- Capitanear: Dirigir personalmente, el propio usuario, los movimientos y acciones de una o varias unidades durante la simulación de una batalla.

4.4. La interfaz gráfica

Tal y como ha sido explicado con anterioridad, la interfaz de usuario de SimuBattle resulta una pieza clave en este proyecto, de acuerdo con los objetivos asumidos en el desarrollo de esta aplicación y dado el enfoque mediante el cual se pretenden abordar. En la interfaz de usuario de una aplicación es donde recae en mayor medida el peso de la usabilidad de la misma, el grado de aprovechamiento de las funcionalidades que oferta y, sin lugar a dudas, gran parte de la opinión del usuario final. Al igual que con los buenos platos gastronómicos, una buena aplicación entra primero por los ojos. Es por ello que el éxito final de una aplicación está intensamente subordinado al aspecto de la interfaz gráfica de la misma. Además, la interfaz gráfica de SimuBattle debía cumplir una serie de objetivos adicionales, que son explicados con mayor detenimiento tras ser enumerados brevemente:

A. Construcción gráfica de la batalla.

La construcción de los mapas y unidades de guerra, es decir, la estructura de la batalla en sí, ha de ser un proceso totalmente gráfico. Debía proporcionar una representación adecuada de estas estructuras, para que su visualización fuera fácilmente comprensible por cualquier usuario. El mayor atractivo que se pretende otorgar a esta aplicación, tal y como se ha venido explicado hasta el momento, descansa principalmente en este punto.

B. Acceso simple a las estructuras

El acceso y uso de estas estructuras que conforman la batalla ha de ser inmediato y sencillo, de modo que el proceso sea lo más intuitivo posible para el usuario.

C. Interfaz gráfica familiar

A pesar de resultar una herramienta con un uso muy específico, debía asemejarse lo más posible a cualquier aplicación convencional que los usuarios finales hubiesen podido usar en un ordenador. Por ello se optó por una visualización similar a la de cualquier procesador de textos para la inclusión de comentarios a las batallas mediante código HTML. Por otro lado, se buscó que la estructura de las diferentes pantallas de la aplicación, fueran lo más homogéneas posibles, conservando siempre el mismo formato. Por tanto, la presentación es siempre una estructura en dos planos, manteniendo a la izquierda la representación gráfica del mapa de la batalla, y a la derecha todas las posibilidades ofertadas al usuario mediante distintos botones. Con este uso de estructuras visuales de sobra conocidas por el usuario medio, se logra concentrar la atención del usuario en la información que ofrece la aplicación y no gastar tiempo en el aprendizaje de su uso.

D. Conexión de las dos dimensiones de diseño (estructural y de contenido)

La interfaz debe implementar mecanismos para conectar la dimensión estructural con la dimensión de contenido, de modo que

el usuario pueda interactuar con las dos dimensiones de la herramienta a partir de una misma interfaz.

E. Una interfaz única para todas las funcionalidades

La aplicación ha de permitir implementar todas sus funcionalidades (creación, parametrización, visualización y simulación de la batalla) en un único entorno, empleando para ello una interfaz lo más homogénea y uniforme posible para aprovechar al máximo la familiarización del usuario con el grafismo empleado.

Cada uno de los puntos expuestos con anterioridad ha hecho obligatoria la toma de una serie de decisiones que serán detalladas, por orden, a continuación.

A. Construcción gráfica de la estructura

Para representar gráficamente los mapas y las unidades de guerra, se decidió optar por una representación clásica de los mismos. Esto es así para aprovechar la familiaridad de dichos elementos gráficos por parte del usuario, redundando en la sencillez del uso de la aplicación.

Empleamos por tanto una representación del mapa de la batalla a modo de tablero clásico de ajedrez, es decir, una representación bidimensional de una imagen de fondo del terreno donde se desarrolla la batalla, y una división del mismo en escaques, con un número de los mismos escogidos por el propio usuario. Inicialmente se sopesó el crear un tablero de casillas hexagonales. Esta forma geométrica ha sido muy utilizada en todo tipo de wargames de mesa, ya en 1961 los juegos “*D-Day*” y “*Chancellorsville*” la empleaban, y desde entonces su uso ha sido muy difundido. La utilización de hexágonos permite conectar cada casilla con otras seis, siendo la única figura geométrica capaz de ocupar de manera continua todo el espacio, ocupando a su vez la menor área posible, redundando así en el mayor aprovechamiento del tablero. Esta posibilidad fue descartada, optando finalmente por el uso de casillas cuadrangulares por su más sencilla representación gráfica, y porque el uso de hexágonos no aportaba ninguna utilidad suplementaria.

Al igual que el fondo del mapa-tablero, las imágenes de las unidades de batalla pueden ser escogidas por parte del usuario entre un grupo de posibilidades que se le ofrecen por defecto, o escogiendo de su ordenador la imagen que desee. En la figura 4.1. se muestran las imágenes de los mapas por defecto.

Para las unidades se optó por una representación también clásica a modo de ficha de juego de mesa. Por ello, su forma es también cuadrangular y ocupa la cuarta parte de la celda. En el caso de las unidades terrestres, éstas se sitúan en el centro de la celda. Para la representación del plano aéreo del mapa y sus correspondientes unidades aéreas, se sopesó varias posibilidades. Una de ellas era representar el mapa en dos “tableros” distintos, uno para el plano aéreo y otro para el terrestre. Esta opción fue rápidamente descartada por lo complicado que resultaría su visualización y comprensión por parte del usuario. Puesto que partíamos de un diseño bidimensional del mapa, finalmente se apostó por tener un único mapa, donde las unidades aéreas, en lugar de ocupar la parte central de la celda como las terrestres, se situarían en la esquina superior izquierda de cada una, obteniendo así una “ilusión” de perspectiva tridimensional, tal y como se aprecia en la figura 4.2.:

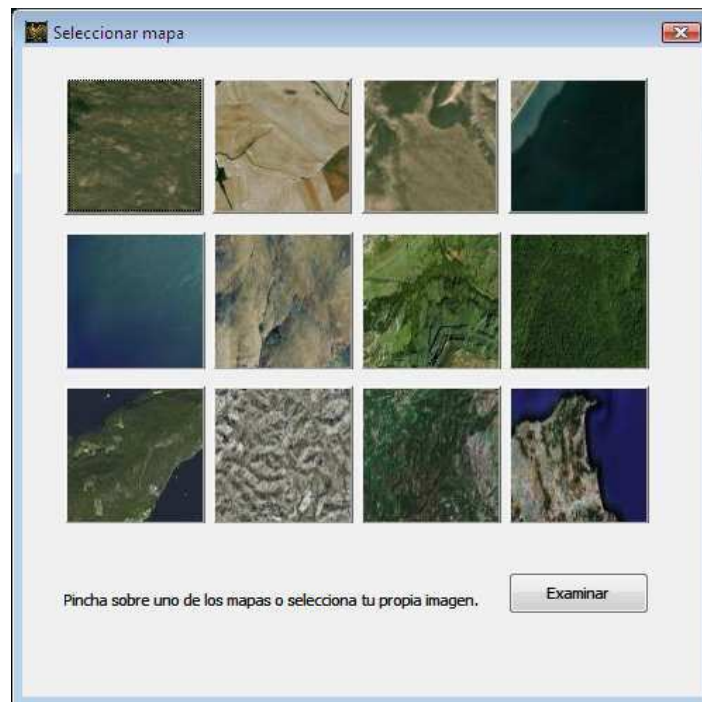


Figura 4.2. Mapas por defecto

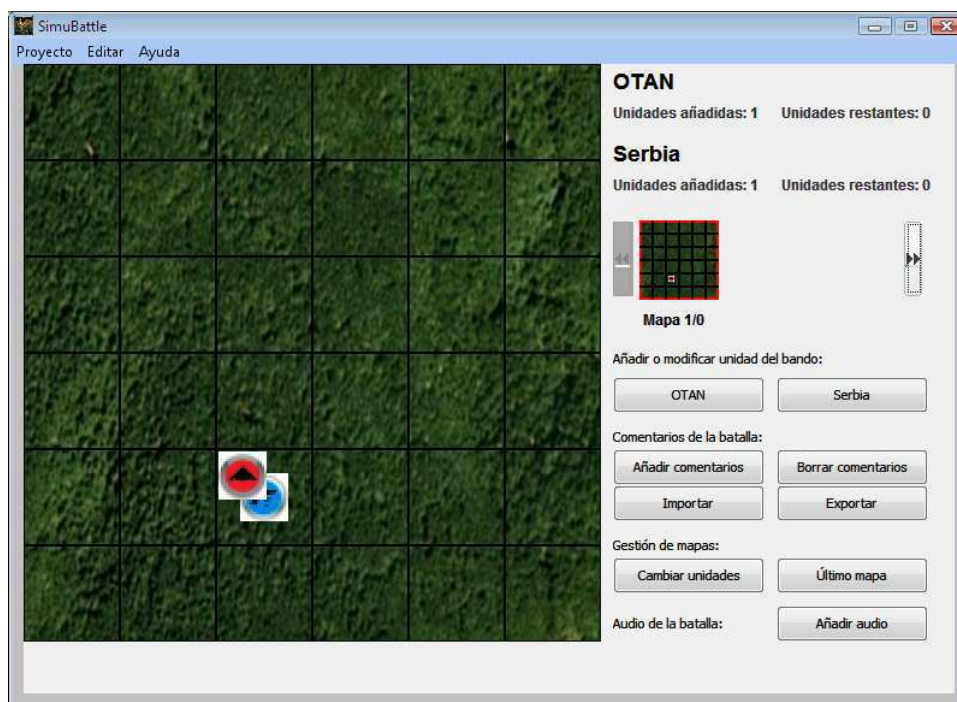


Figura 4.3. Unidades terrestre y aérea

Tal y como se ha dicho con anterioridad, se ofrece al usuario la posibilidad de seleccionar las unidades a su antojo, mostrándole como posibilidades una serie de imágenes por defecto, que para cada uno de los dos bandos son como las mostradas en la Figura 4.3.



Figura 4.4. Iconos de unidad por defecto

La selección de celdas se realiza mediante clicado del ratón sobre las mismas, para ello, se marcan gráficamente los bordes de las mismas de color rojo. Por su parte, el marcado de unidades en la visualización y la simulación, sigue un proceso similar: en el caso de marcar la unidad que en ese momento efectúa el movimiento, el borde de la misma se volverá azul, mientras que si una unidad recibe un disparo o impacto, se marcará de color rojo. Así, a lo largo de la aplicación, este código de colores se repite en el marcado de las unidades: azul para la unidad que posee el turno y rojo para aquellas que reciben un impacto. A continuación puede verse un ejemplo del marcado de celdas y unidades, en la Figura 4.4.

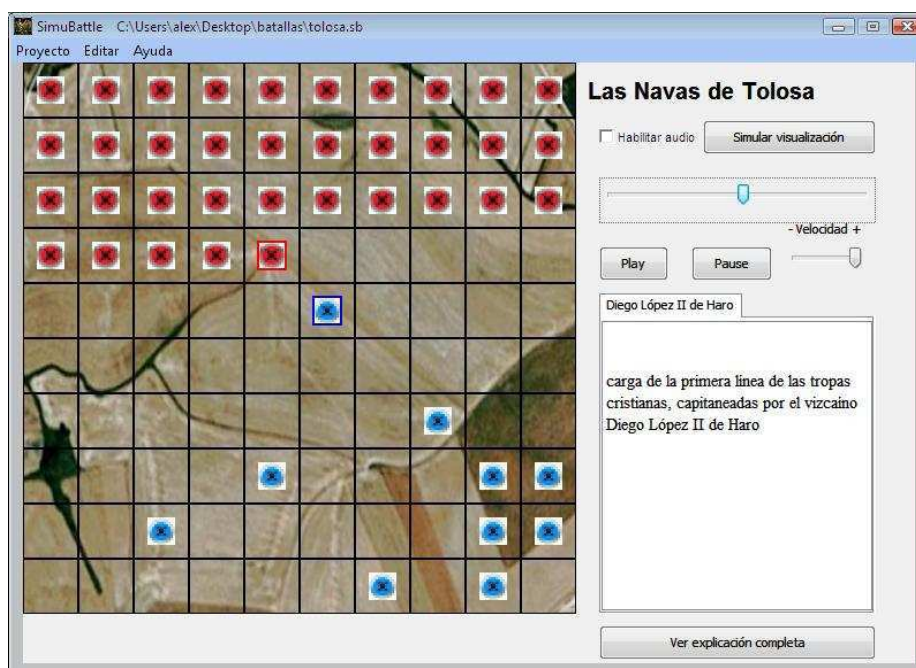


Figura 4.5. Marcado de unidades

B. Acceso simple a las estructuras

Tal y como ya se ha explicado, la selección de las distintas celdas se realiza sobre el mismo mapa sobre el que se trabaja. Mediante movimientos de ratón y clicados del mismo, se escoge la celda sobre la que se quiere posicionar o borrar una unidad, la celda que se desea parametrizar... Las opciones que pueden realizarse sobre el mapa se muestran siempre a la derecha de la aplicación en forma de botones clásicos, botones de radio (“radio buttons”) y dispositivos deslizantes (“sliders”). La Figura 4.5. muestra esto con claridad.

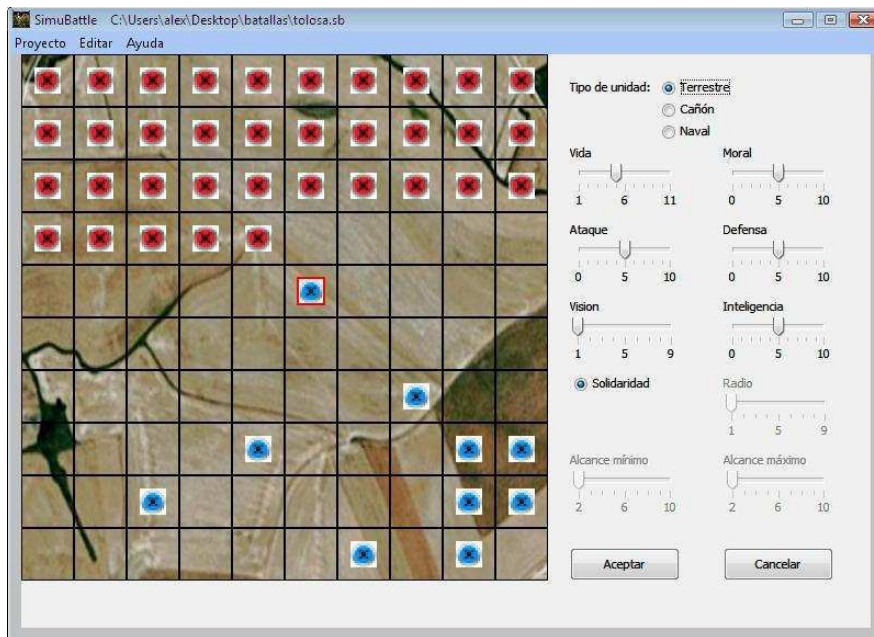


Figura 4.6. Pantalla SimuBattle

Mención aparte merece la participación del usuario en la simulación de una batalla. Si el usuario decide “capitanear” alguna unidad, cuando sea su turno, le aparecerá un combo con las distintas opciones de movimiento o disparo. En caso de seleccionar disparo, se mostrará un mapa de menor tamaño donde aparece representada dicha unidad junto a las unidades enemigas que puede atacar dadas sus características. Al posicionarse sobre las mismas, para poder ser seleccionadas, aparece una representación de las características de la misma a la derecha en forma de barras, de distinto tamaño y color según su valor, siendo el verde el máximo y el rojo el mínimo, pasando por el intermedio amarillo. Estos colores son también utilizados en el marcado de las unidades durante la simulación al seleccionar la visualización de alguna característica, tal y como vemos en las figuras 4.6. y 4.7.

C. Interfaz gráfica familiar

Tal y como se ha comentado más de una vez con anterioridad, SimuBattle consta de cuatro funciones básicas: creación de batalla, parametrización, visualización y simulación. Cada una de las cuatro funciones tiene asociada una representación gráfica propia, siendo por tanto independientes unas de otras, pero eso sí, con una estructura similar. En el lado izquierdo de la pantalla, siempre se muestra el mapa de la batalla con las unidades correspondientes. A la derecha, se presentan mediante sencillos botones todas las posibilidades presentes para el usuario, dependiendo de la

funcionalidad en la que se encuentre en ese momento. Con esto logramos una continuidad en el comportamiento gráfico a lo largo de todo el uso de la aplicación, lo cual redundará en el beneficio de quien utiliza por primera vez esta herramienta. Así, el hecho de que cada función conlleve un diseño independiente, refrescándose la pantalla, permite al usuario ser consciente de que va a pasar a desempeñar una funcionalidad diferente a la que estaba realizando hasta ese momento, pero como se ha dicho, sin perder la continuidad en el diseño gráfico en ningún momento.



Figura 4.7. Características de unidad y enemigo

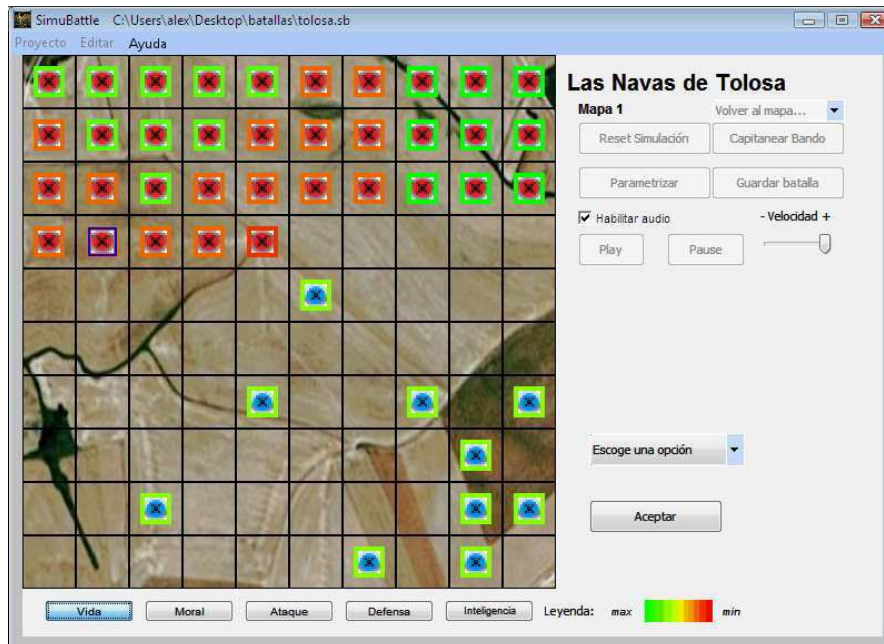


Figura 4.8. Marcado de características

Las figuras 4.8. y 4.9. muestran dos ejemplos de funciones distintas (creación de batalla y visualización de la misma), donde se pueden apreciar con claridad las características aquí explicadas.

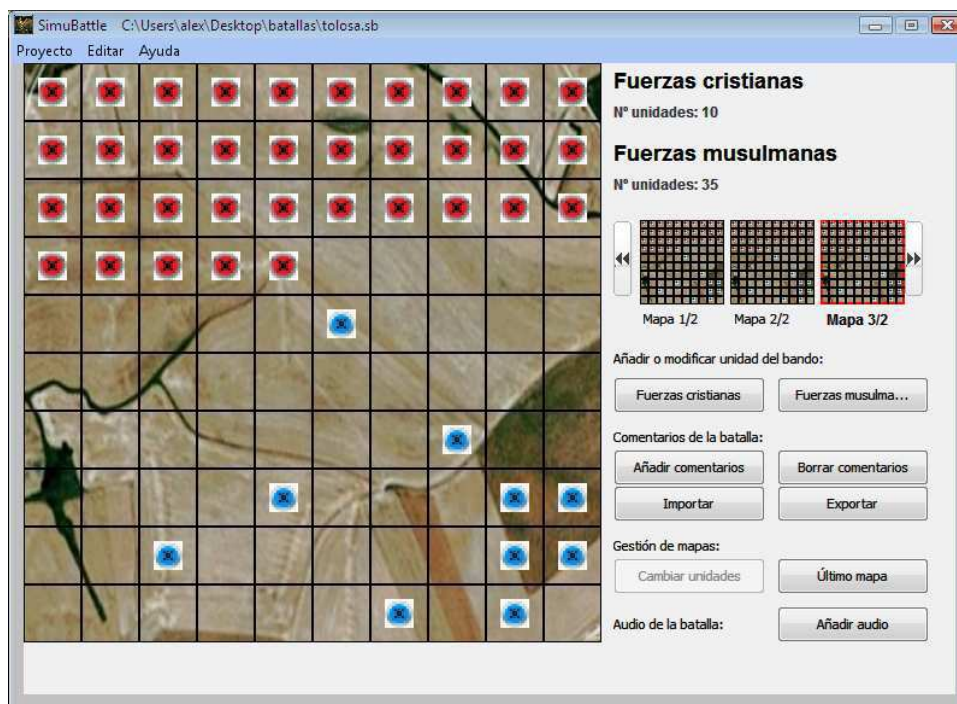


Figura 4.9. Edición de batalla

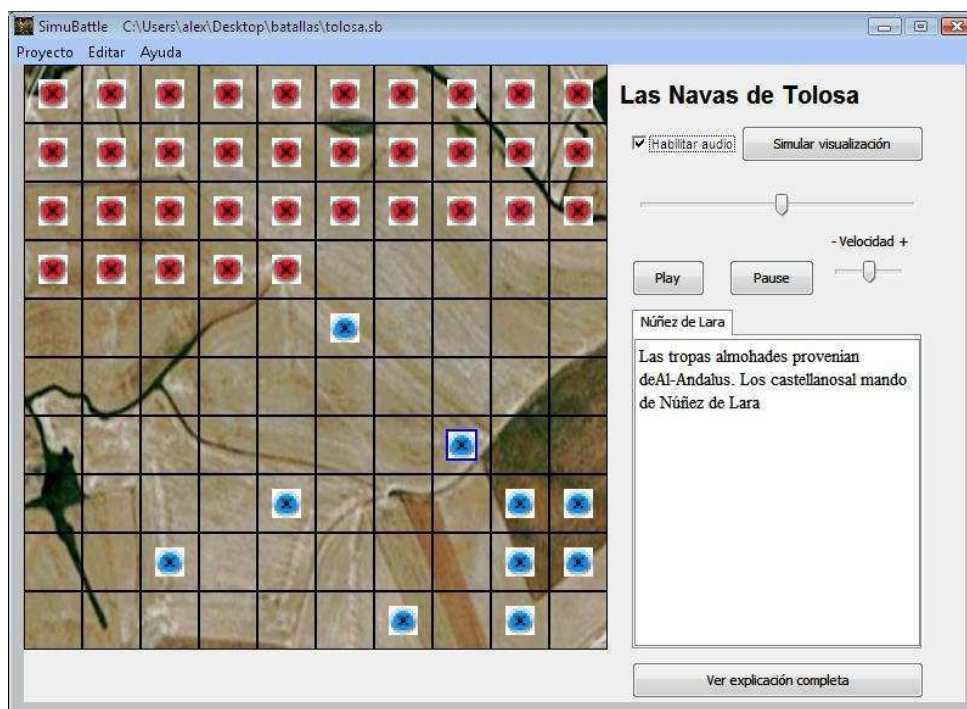


Figura 4.10. Visualización de batalla

Por otro lado, la inclusión de comentarios en las batallas en forma de código HTML requería una visualización completamente independiente. Para ello se requería un formato visual similar al de cualquier procesador de textos que podemos encontrar en el mercado, como los más que conocidos Microsoft Word, o WordPad. Así pues, nos servimos de una aplicación java en código abierto llamada SimplyHTML [47], y que consiste en un procesador de texto enriquecido, creador de documentos HTML en combinación con hojas de estilo

(CSS). Este editor HTML, por tanto, supone la base de nuestro editor de comentarios de batallas. Mediante la sustracción de algunas funcionalidades que no servían para SimuBattle, y, sobre todo, mediante la adición de muchas otras características de nuestro trabajo como el enlace de comentarios a unidades concretas o la inclusión de videos del portal YouTube, obtenemos un creador de comentarios con un formato de editor de textos clásicos, que era nuestro fin. La figura 4.10. permite visualizar la apariencia de este editor de comentarios HTML para batallas incluido en SimuBattle.

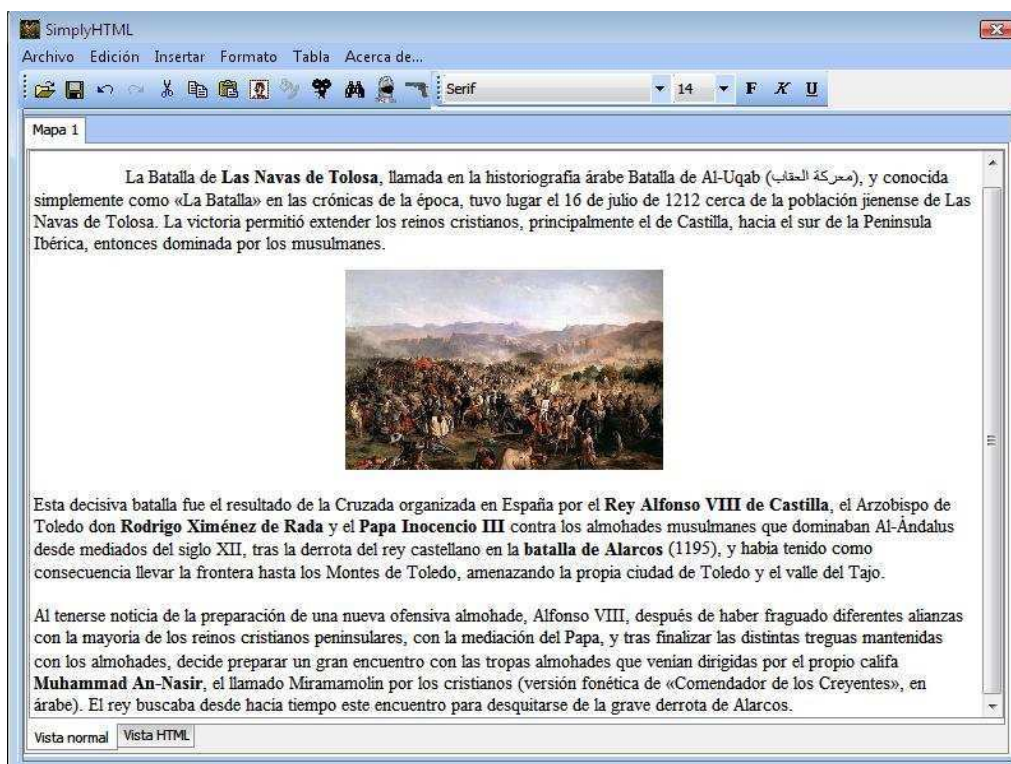


Figura 4.11. Edición de comentarios HTML

D. Conexión de las dos dimensiones de diseño (estructural y de contenido)

Como se ha dicho anteriormente, puede considerarse que la aplicación SimuBattle consta de dos dimensiones de diseño, una concerniente a la creación gráfica de una batalla y otra al contenido “literario” y/o audiovisual de la misma, es decir, los comentarios y explicaciones a la misma. Es por ello necesario que la interfaz gráfica dote de mecanismos a la herramienta para permitir al usuario interactuar entre ambas.

Antes de nada, expliquemos brevemente la estructura de una batalla en SimuBattle para comprender mejor la forma en que se ha trabajado para dar solución gráfica a esta interacción entre las dos dimensiones del diseño. Una batalla está formada por uno o más mapas situados de forma secuencial. Cada mapa contiene una o más unidades, las cuales poseen cada una un turno o movimiento. Esto quiere decir, que a la hora de visualizar una batalla, veremos una sucesión de movimientos de unidades, que en verdad no es más que una sucesión de mapas. Por ello, cada mapa llevará asociado un fichero HTML de comentarios.

Así pues, la idea se fundamenta en que, a la vez que se van creando los distintos mapas de la batalla, en paralelo a su construcción, pueda accederse a la edición de los comentarios asociados a cada uno. Para ello, empleamos un simple botón, el cual, mediante su activación, hace aparecer en pantalla un editor de texto.

Un editor de texto es un entorno con el que cualquier usuario novel de informática está familiarizado, cumpliendo por tanto uno de los principales requisitos de nuestra aplicación: la sencillez de la herramienta dado el público al que va destinado, que no es otro que profesores de historia y alumnos. La elección de una interfaz similar a la de un editor de texto era, por lo tanto, una buena opción para cubrir el diseño de la dimensión de contenido textual. Por otro lado, este editor de texto debía incorporar algunas funcionalidades adicionales a las de la simple edición de texto necesarias para la tarea específica que se quería abordar.

En este momento, surgían dos posibles opciones desde el punto de vista de la implementación: desarrollar por completo y desde cero un editor específico completamente adaptado a las necesidades de la aplicación SimuBattle, o emplear un editor ya existente e integrarlo de forma total en nuestra herramienta, agregándole las funcionalidades específicas de las que careciera.

Tras evaluar exhaustivamente ambas posibilidades, considerando principalmente un análisis comparativo entre el esfuerzo y el tiempo de implementación frente al resultado final obtenido, se llegó a la conclusión de que la opción más viable era la de utilizar un editor ya existente que cubriera la mayor parte de la funcionalidad e introducir en él, durante la fase de integración, aquellas tareas específicas necesarias. A esta conclusión se llegó, no sólo sopesando el aspecto de la interfaz del editor, que también, sino que además primó el hecho de que esta opción permitiría dedicar mayor tiempo y esfuerzo en la creación de la estructura básica de SimuBattle así como el resto de funcionalidades adicionales, aportando así una mayor riqueza a la solución global.

El editor seleccionado para ser integrado en la herramienta con este propósito, tal y como ya se ha explicado con anterioridad, es SimplyHTML. Se optó por este editor en código abierto porque está escrito en Java, y se distribuye junto con su código fuente completo, así como con una documentación muy detallada sobre su uso, funcionalidad e implementación, lo que facilita en gran medida el trabajo de adaptación e integración con nuestro desarrollo.

Anteriormente ya se detalló que SimplyHTML es un procesador de texto enriquecido mediante HTML y hojas de estilo. El hecho de que pueda ser utilizado como una aplicación independiente o como un componente java más, así como la utilización de tecnologías independientes de la plataforma como lo son Java, HTML y CSS que hacen posible que sea utilizado en casi cualquier máquina, fueron razones que pesaron a la hora de seleccionar este editor.

Por todos estos motivos, desde un principio SimplyHTML se perfiló como el mejor candidato para cubrir los objetivos perseguidos por SimuBattle. A continuación se detalla el proceso de integración de esta herramienta hasta llegar a ser el editor de comentarios de nuestra herramienta, poniendo

especial interés en la descripción de las funcionalidades denominadas como específicas.

Entre estas funcionalidades destaca la asociación de comentarios concretos a una unidad. ¿En qué momento surge esta necesidad? Para contestar a esta pregunta recordemos nuevamente la estructura de nuestra “batalla”, que no es otra que la sucesión de mapas con un fichero HTML asociado a cada uno. En el momento de visualizar una batalla, se nos planteaba la posibilidad de presentarla como un mapa tras otro, pero esto convertía la visualización en algo caótico, excesivamente rápido y se perdían muchos detalles. Por ello se optó porque, a pesar de que la estructura siguiera siendo la misma (sucesión de mapas), a la hora de visualizar la batalla, la transición entre mapa y mapa se produjera de una manera escalonada y ordenada. Esto es, cada espacio de tiempo de la visualización se corresponde con el movimiento de una unidad. Esto podía permitir una explicación de la batalla más detallada, con comentarios asociados ya no sólo a un mapa, sino a una unidad.

Pero, ¿cómo lograr esto sin alterar la estructura ya existente de un fichero HTML por mapa? La opción inicial de cambiar la estructura y hacer que existiera un fichero de comentarios por cada movimiento fue rápidamente descartada, por lo inabarcable que resultaría para el creador de la batalla crear todos estos ficheros HTML. Por ello se optó por una solución intermedia y altamente satisfactoria, que permite un mayor detalle en los comentarios sin que esto resulte un trabajo excesivamente engorroso para el usuario. Cada mapa seguiría teniendo asociado un fichero HTML, pero en la edición de este código HTML, se permitiría generar una nomenclatura específica que, llegado el momento de la visualización, mostrara sólo los comentarios propios de un movimiento, y no los de todo el mapa. Para ello, el usuario puede escribir los comentarios totales al mapa en que se encuentra, pero durante la visualización, en una pequeña pantalla, sólo son mostrados aquellos asociados al movimiento concreto que se está produciendo. Estos comentarios asociados al movimiento se producen mediante el subrayado de la parte que quiere visualizarse, y la inclusión de una “marca” HTML con el identificador de la unidad que efectúa el movimiento.

En la figura 4.11. podemos apreciar un ejemplo altamente aclaratorio. En el momento en que se produzca el movimiento de la unidad “Núñez de Lara”, en la pantalla de comentarios de la visualización aparecerá el texto subrayado, es decir: “Las tropas almohades provenían de Al-Andalus. Los castellanos, al mando de Núñez de Lara”.

Además, aparecerá la opción de ver todo el texto del mapa en una pantalla mayor, así como verlo en el navegador habitual empleado por el usuario, para una correcta y completa visualización de los comentarios. Como se ha visto, la selección del texto se realiza mediante un simple subrayado. ¿Pero cómo incluir la “marca” HTML característica de la unidad? Muy simple, mediante la nueva opción creada en SimplyHTML ex profeso para SimuBattle. Esta opción denominada “Asociar unidad”, está visible en la barra de herramientas, y al pinchar en ella, se abre una imagen en pequeño del mapa con todas las unidades, y pinchando sobre la deseada, se genera la “marca” asociativa de la unidad. La marca implica que, todo el texto subrayado anterior a la misma y posterior a otra, será el mostrado junto al movimiento de la unidad.

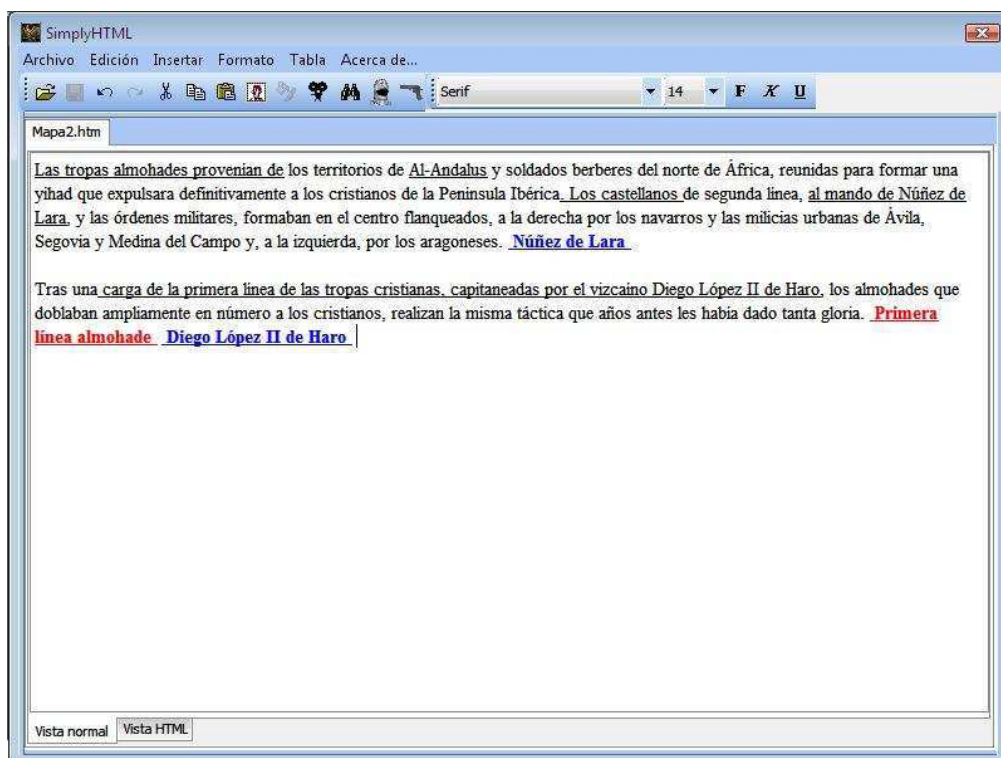


Figura 4.12. Ej. de comentarios a unidad

Para la realización de esta “marca”, necesitábamos emplear código HTML. Para ello se optó por una medida sencilla: emplear una etiqueta de enlace <a> cuyo HREF estuviera vacío, y cuyo ID fuera el identificador interno asociado a cada unidad y con un estilo de color azul, mientras que por pantalla se muestra el nombre dado por el usuario a dicha unidad para una mayor comprensión por su parte.

Esto abría la posibilidad, además, de realizar otro marcado, no sólo para la unidad que realiza un movimiento, sino para la unidad que en ese movimiento recibe un impacto por parte de la primera unidad. Esta implementación fue sencilla habiendo creado ya la anterior. Esta opción se denomina “Asociar enemigo”, y también está presente en la barra de herramientas de SimplyHTML. El funcionamiento es el mismo, y la marca en este caso es similar, pero de color rojo.

Veamos un ejemplo de este mecanismo. En la Figura 4.11. podemos observar que, cuando la unidad denominada “Diego López II de Haro” realice su movimiento, éste será representado como un ataque a la unidad llamada “Primera línea almohade”, y el texto que aparecerá asociado a este movimiento será: “carga de la primera línea de las tropas cristianas, capitaneadas por el vizcaíno Diego López II de Haro”.

Cabe destacar que, durante el visionado de los comentarios por parte del usuario final, el subrayado del texto de cada unidad, así como las marcas asociadas a las unidades, son filtradas y eliminadas, siendo únicamente visibles durante la creación de la batalla y nunca durante su visualización, quedando así un texto limpio y claro.

El editor permite no sólo la inclusión de texto, sino también de imágenes. Es precisamente ese requisito el que nos empujó al empleo de un editor HTML y no de texto plano simplemente. Las bondades educativas de todo lo relacionado con lo multimedia, fueron expuestas en apartados

anteriores, por ello no es necesario volver a recalcarlas, pero para completar el círculo y enriquecer aún más la aplicación, se optó por añadir otra funcionalidad: la posibilidad de incluir, embebidos, videos del conocido portal YouTube. Mediante la opción “Insertar video” de la barra de herramientas, un cuadro de diálogo pide unos valores básicos como son la anchura y altura del video, así como el color del marco que lo contiene. Para facilitar más aún su utilización a cualquier tipo de usuario novel, el único parámetro que se pide introducir asociado al video, es la URL de la página de YouTube.

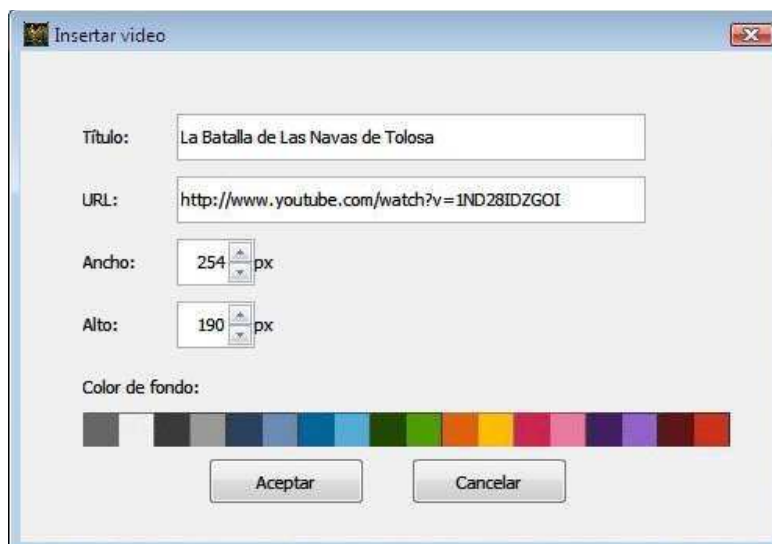


Figura 4.13. Inserción de video

Toda esta información que el usuario creador de la batalla puede introducir, podría perderse durante la visualización de la misma, pues el usuario final podría verse en la tesitura de no poder prestar atención por un lado a los movimientos de las unidades y por otro a los comentarios de la batalla. Por ello se empleó la posibilidad de asociar comentarios a cada movimiento, para acotar los textos mostrados por pantalla, pero eso sí, ofreciendo la opción de ver los comentarios completos del mapa en una pantalla mayor o incluso en el navegador habitual del usuario, pudiendo disfrutar así de todas las posibilidades multimedia ofrecidas por el editor HTML.

Por otro lado, cabe destacar que, puesto que la inclusión de comentarios va asociada a un único mapa, al abrir el editor SimplyHTML se cambió su funcionalidad para que siempre por defecto apareciera el texto asociado al mapa. De igual forma, es posible abrir otros ficheros HTML a voluntad del usuario, pero nunca es posible cerrar el fichero del mapa asociado hasta que no se sale del editor.

E. Una interfaz única para todas las funcionalidades

Ya pasamos fugazmente sobre este requisito, pero cabe recalcar de forma independiente su casi vital importancia en la aplicación SimuBattle. Desde un primer momento se quiso aunar las cuatro funcionalidades básicas (creación de una batalla, parametrización, visualización y simulación) en una única interfaz gráfica, en una sola aplicación. Esto permite no sólo una continuidad en el grafismo que facilita el uso de SimuBattle de cara a los

usuarios noveles, sino que además logra una integración de todas las partes, creando un todo único que redundará en la libertad del usuario final para navegar de un lado a otro de la aplicación y por tanto en la posterior realimentación de conocimientos.

Esta realimentación es conseguida gracias a la posibilidad de que un usuario pueda generar una batalla desde cero, crear a su antojo los distintos mapas y el desarrollo de la misma. En la misma interfaz logra editarla, visualizarla, parametrizarla y simularla. Y la clave reside en que es posible guardar en disco una simulación, como si de una batalla editable se tratara. De esta forma cualquier usuario podría retomar el trabajo de otra persona, y trabajar sobre dicha batalla creada a partir de la simulación y agregarle comentarios, variar el posicionamiento de las unidades, parametrizarla de manera diferente y volver a simularla, aportando su punto de vista particular de la historia. Podría decirse que se generan nuevas batallas a partir de las simulaciones de una inicial, con el potencial didáctico que ello implica.

4.5. Motor de juego

Aunque por motor de juego se entiende, de manera purista, la serie de rutinas de programación que permiten el funcionamiento y dinámica de un videojuego, hemos optado por dar este nombre a este bloque del programa por varios motivos. El primero es que su principal subestructura es el algoritmo de simulación, cuyo funcionamiento a modo de motor de reglas se asemeja en gran medida al motor de juego de un videojuego al uso. Una pretensión clave de SimuBattle es el acercamiento de la historia a los más jóvenes, ofreciendo un juego interactivo didáctico, por ello, aunque este bloque, de manera más genérica, podría haberse denominado “lógica de negocio”, dada la importancia capital de este concepto, preferimos englobarlo bajo la denominación de “motor de juego”.

Por tanto, este bloque engloba a todo el conjunto de acciones que se realizan a nivel interno en la aplicación. En concreto, se buscaba satisfacer los siguientes objetivos:

- Realización del algoritmo de simulación. De una manera encapsulada, crear un motor de reglas tal que, recibiendo los parámetros de la situación concreta de una batalla, devuelva un movimiento coherente de la unidad que posee el turno, para su posterior visualización por parte de la interfaz gráfica.
- Ejecución de las tareas ordenadas por el usuario como respuesta a los eventos generados en la interfaz.
- Tareas de entrada y salida de datos físicos, así como la persistencia y mantenimiento de los mismos.
- Funcionamiento interno del editor de textos HTML.

Dada la importancia del algoritmo de simulación, o motor de juego como tal, su funcionamiento concreto se detallará en un apartado propio, el 4.6., titulado “Algoritmo de simulación”. Es ahí donde se detalla el sistema de reglas en el que se basa, las distintas opciones que se plantearon, qué decisiones se tomaron y qué las motivaron... Si bien se puede ya adelantar algunas características básicas, como la existencia de tres tipos distintos de celdas (terrestre, aérea y naval), así como tres distintos de unidades (unidad básica, aérea y cañón). Cada tipo de unidad tiene unas características y cumple unas reglas propias en cuanto a movimiento y ataque. Con estas divisiones se intentó lograr, de una manera lo más simple posible, abarcar el mayor número de opciones y capacidades. Otras características interesantes, como la inteligencia de cada unidad, así como el grado de cooperación y solidaridad entre las mismas, son detalladas también en el apartado 4.6.

Un punto, también muy importante, y que puede llegar a pasar más desapercibido, es todo lo concerniente al plano de contenido y la persistencia, así como las posibles respuestas a los eventos generados por la interfaz. Estos últimos, se encuentran físicamente más cerca de la interfaz gráfica, pero su lógica reside plenamente en el motor de juego o inteligencia del programa. Una vez establecido el modo en el que todas estas funciones han de llevarse a cabo, el resto de tareas auxiliares no requieren de demasiado esfuerzo de diseño.

Debido a que el editor de texto SimplyHTML que se utiliza para la edición de comentarios está basado en HTML, fácilmente se llegó a la decisión de que un buen modo de mantener la consistencia en la información perteneciente a este plano era almacenarla precisamente así, empleando ficheros HTML. Por otro lado, el almacenamiento de la información en este formato no sólo era conveniente para la compatibilidad con el editor empleado, sino que además permitía tanto su exportación como importación, permitiendo así utilizar dichos ficheros de comentarios en otros editores o visores.

Hasta el momento, durante toda la fase de diseño, se ha pensado en el hecho de que la herramienta debía estar ideada y orientada a usuarios a los que no se les suponía un conocimiento técnico avanzado en informática. Sin embargo, no hay que descartar taxativamente la posibilidad de que algunos de los usuarios finales de la herramienta sí posean esos conocimientos. Para ellos se dirige esta funcionalidad, basándose en el almacenamiento de la información en formato HTML, el editor de texto posee una pestaña de edición en la que se puede ver el código HTML correspondiente al contenido introducido en el área de texto. Es así como, los creadores de la batalla que editen comentarios a la misma que poseen el conocimiento necesario, y que así lo deseen, pueden modificar este código HTML a su antojo.

Esta funcionalidad, que a primera vista puede parecer demasiado específica, se torna especialmente útil si tenemos en cuenta el hecho de que, por razones que se detallarán más adelante, los comentarios relativos a cada mapa de la batalla se almacenan en forma de un fichero HTML por cada uno. Así, logramos las siguientes ventajas:

- El código HTML que representa los comentarios de un mapa es legible por el usuario humano y fácilmente interpretable.
- Este formato es independiente de la aplicación, lo que supone una gran ventaja en caso de que se quiera rehacer, modificar o incluso crear desde cero desde otra aplicación externa. Esto nos abre la posibilidad a importar y exportar ficheros de comentarios de la batalla de manera independiente.

En cuanto a la persistencia, rápidamente uno se da cuenta de lo particular de la información que ha de almacenarse en nuestra aplicación. Nos encontramos con que lo que guardamos es información relativa a mapas, unidades, celdas, objetos de simulación... Toda esta información carece de sentido fuera de SimuBattle, no aporta nada su exportación a otro tipo de formatos. Es por ello que desechamos la idea de guardar la información mediante un archivo XML y la construcción de una DTD (Document Type Definition) que definiese la estructura de la batalla. Bien es cierto que esta solución hubiera sido la más elegante, así como la más eficaz para poder exportar los datos y acceder a ellos desde una aplicación externa a SimuBattle. Pero tal y como hemos planteado, esta utilidad no iba a ser excesivamente demandada dada la particularidad de nuestros datos, sobre todo cuando comparamos el tiempo y esfuerzo destinado a esta solución, cuando Java nos ofrece una solución más rápida y que se adaptaba mejor a nuestras necesidades: la serialización de objetos.

Java facilita el almacenamiento y transmisión de un objeto mediante un mecanismo conocido con el nombre de serialización. La serialización de un objeto consiste en generar una secuencia de bytes lista para su posterior

almacenamiento o transmisión. Posteriormente, es posible recuperar el objeto original mediante un proceso inverso a la serialización, pudiendo así reconstruirlo de nuevo.

Su empleo es muy sencillo, pues para que un objeto Java sea serializable, basta con que implemente la interfaz “Serializable”. Esta interfaz no tiene métodos, y por tanto es muy sencillo su implementación, pues no hay que definir nuevos métodos, simplemente declarar que la clase a la que pertenece nuestro objeto implementa la interfaz “Serializable”.

Mediante esta sencilla solución, cubrimos de un plumazo la engorrosa necesidad de guardar la información propia generada por nuestra aplicación. Ahora bien, aún resta por ver cómo almacenar el resto de información audiovisual: ficheros HTML, imágenes y ficheros de audio. Puesto que esta información sí es completa en sí misma y exportable a numerosos programas, se optó por mantenerlos tal cual, y durante la ejecución de la aplicación, éstos se van almacenando en una carpeta temporal para uso interno de SimuBattle. Finalmente, para guardar en un único fichero legible por nuestra aplicación, tanto la información particular de nuestros objetos serializables como los archivos HTML, imágenes y ficheros de audio, se optó por algo tan simple y común como comprimirla en un único archivo en formato zip, cambiándole la extensión posteriormente a sb (SimuBattle).

El uso de este formato de compresión, soportado de forma cómoda por Java, no restaba compatibilidad a la solución, ya que zip es un formato estándar. De esta forma, todos los archivos son recuperables fácilmente por cualquier otra aplicación externa.

De esta forma, cuando nuestra herramienta abre un archivo nuevo desde disco, lo primero que hace es cambiar la extensión a zip y descomprimirlo llevando los archivos que contiene a un directorio temporal creado por la herramienta, y cargando los objetos serializables en memoria para su uso. De igual manera, cuando se comienza la creación de una batalla desde cero y se le van añadiendo comentarios HTML, imágenes y archivos de audio asociados, éstos se van almacenando en el directorio temporal. Al seleccionar la opción de guardar, estos ficheros son comprimidos con la misma estructura junto a los objetos serializables de la batalla, en la ruta y con el nombre proporcionados por el usuario, guardándose nuevamente con extensión sb. Los archivos temporales se borran con la apertura de una nueva batalla, y se hace una limpieza total cuando se cierra la aplicación, de modo que el sistema de archivos del sistema operativo queda limpio tras el uso de SimuBattle.

Llegados a este punto, podría darse por cerrada la descripción y justificación del que ha sido finalmente el diseño de SimuBattle. Sin embargo, dada su importancia, ya justificada con anterioridad, se ha estimado oportuno dedicar un apartado especial al diseño del algoritmo de simulación.

4.6 Algoritmo de simulación

Del estudio detallado del libro “*AI Game Programming Wisdom*” [49], obtuvimos una serie de conocimientos básicos sobre inteligencia artificial, así como unas ideas que sirvieron de base para nuestro posterior algoritmo de simulación de una batalla:

- Una técnica que todos desean alcanzar es un algoritmo tal que simule una máquina de aprendizaje (“*machine learning*”). Consistiría en un motor tal que fuera aprendiendo de las actuaciones de su rival humano. Finalmente, esta opción, aunque altamente tentadora, no es nada recomendable, pues de igual manera que puede aprender del usuario, también puede adquirir vicios y errores humanos.
- Cada “juego” ha de tener su propia inteligencia artificial propia y diferenciada, según las necesidades. Puede entremezclarse y basarse en inteligencias de otras aplicaciones, pero nunca calcarlas.
- El algoritmo ha de poder simular comportamientos humanos predecibles, pero también impredecibles. Lo ideal sería lograr una inteligencia artificial que fuera capaz de dar sorpresas, e incluso cometer “estupideces”.
- Las técnicas más simples suelen ser las más efectivas, eficientes y eficaces, como por ejemplo máquinas de estados finitas, árboles de decisión o sistemas de reglas.
- Ha de evitarse que la máquina pueda llegar a “hacer trampas” por tener más información que el propio usuario.
- Un sistema centralizado emplea menos recursos que un conjunto de eventos o disparadores (“*triggers*”), pero resulta más complicado dotar de inteligencia independiente a las distintas unidades.
- La respuesta de las unidades puede decidirse finalmente mediante un rango de prioridades o curvas de respuesta en función de distintos parámetros característicos.
- Los movimientos de las distintas unidades pueden basarse en diferentes algoritmos de búsqueda de caminos como “*A* Pathfinding*” o el Algoritmo de Dijkstra. Estos métodos pueden a su vez ser utilizados para el descubrimiento de las llamadas “zonas oscuras” del mapa. Pueden optimizarse mediante la inclusión de bloqueos, limitaciones de tiempos, comportamiento ante búsquedas fallidas, o incluso dotando a cada unidad de un número de “sensores” igual al número de puntos cardinales y uno central, cuya combinación permite decidir cómo será su comportamiento.
- Conviene que se estudie aquellas celdas donde una unidad puede estar a salvo y desde dónde puede atacar. La visibilidad es clave.
- Los movimientos en grupo son claves y pueden adoptarse desde distintos puntos de vista: reglas propias, maximización de puntos de contacto con el enemigo (atacante), minimización de puntos de contacto

(defensivo), comunicación entre miembros del mismo bando, formaciones tácticas complejas...

Todo este conjunto de indicaciones, y las posteriores pruebas efectuadas, impregnaron y salpicaron todas las decisiones tomadas durante el proceso de creación del motor de juego, llevando a conformar un algoritmo de simulación cuyas características básicas son las siguientes:

- El sistema empleado será un sencillo sistema de reglas.
- Cada unidad actuará de manera independiente, pero su comportamiento vendrá determinado por cierta predisposición a permanecer lo más unida posible a otra unidad de su mismo bando. Esto permite que las unidades, aunque independientes, mantengan cierta conciencia de grupo.
- Las unidades tienden a buscar aquella celda cuyo parámetro “accesibilidad” sea mayor en una escala de 0 a 10, pues en aquellas celdas más accesibles, les será más fácil atacar y defender, puesto que son consideradas posiciones estratégicas.
- Las unidades tienen memoria de las celdas visitadas donde no hay enemigo para no volver a pisarlas temporalmente.
- Las unidades tienen memoria de las celdas donde han sufrido ataques, para evitar, en la medida de lo posible permanecer en ellas, peligrando su integridad de manera innecesaria.
- Las unidades se comunican entre ellas, reclamando ayuda al compañero más cercano en caso de peligrar su vida, e informando de su posición cuando entran en batalla con un enemigo.
- Las unidades cuentan con un parámetro llamado “inteligencia”, que sirve para calcular lo más acertadamente posible los parámetros y características de sus enemigos.
- Integrando la visión de cada unidad, junto a los parámetros de los enemigos obtenidos a través de su inteligencia, comparándolos con los suyos propios, y sopesando las distintas posibilidades de movimiento o ataque, la unidad toma una decisión en cuanto a su comportamiento.
- Los cañones tienen un radio de acción, dentro del cual, y dado que causan un número variable de daños colaterales, tienden a disparar sobre aquella unidad donde creen que causarán más daños.
- Las unidades terrestres y aéreas tienden a enfrentarse entre sí antes que unas con otras.
- Las unidades navales sólo pueden avanzar sobre aquellas celdas navales, e igual ocurre con las terrestres.
- Una celda será inaccesible cuando no esté activado su parámetro “transitabilidad”.

- Según los logros o fracasos de cada unidad y los de su propio bando, sus características pueden mejorar o empeorar durante el transcurso de la simulación de la batalla.

Estas reglas aquí enunciadas son las básicas que conforman el algoritmo de simulación. Como se puede apreciar, con unas pocas reglas puede conformarse un comportamiento de las unidades muy aceptable. Todas estas decisiones no fueron tomadas desde un principio, sino que fueron surgiendo según se avanzaba en el diseño e implementación del motor de juego. Aún así esto no es más que un esbozo del funcionamiento completo del algoritmo. Para comprender mejor su total comportamiento, veamos un paso a paso para los tres posibles tipos de unidades: terrestre, aéreo y cañón. Pero antes veamos las características de una celda y de una unidad.

Una celda tiene como características si es naval o no, si es transitable o no, y un número entre 0 y 10 que define su “accesibilidad”. Como se ha visto antes, la accesibilidad influye en el comportamiento de la unidad que la ocupa, siendo éste más efectivo cuanto más accesible sea la celda.

En cuanto a la unidad, además de un número identificativo interno único que la define de manera unívoca, el tipo de unidad que es (terrestre, aérea, cañón), si tiene un carácter naval o no en el caso de que el tipo de unidad sea terrestre (el comportamiento de las unidades navales es igual al de las terrestres, sólo que transitan por celdas navales), si es solidario con sus compañeros, contiene una serie de contadores estadísticos que se explicarán más detalladamente a continuación. Pero principalmente contiene las siguientes características:

- Vida: Valor entre 0 y 10 que indica la vida que le resta a la unidad. En cuanto alcanza el mínimo, la unidad es considerada muerta y desaparece del mapa.
- Moral: Valor entre 0 y 10 que marca la moral con la que la unidad encara la batalla. Cuanto mayor es este valor, más efectivos son sus ataques y defensas.
- Ataque: Valor entre 0 y 10 que marca la capacidad de causar daños a sus oponentes.
- Defensa: Valor entre 0 y 10 que marca la capacidad de defenderse de ataques enemigos.
- Visión: Valor en número de celdas que marca la distancia que es capaz de alcanzar en un disparo y, también, la visión de las celdas adyacentes.
- Inteligencia: Valor entre 0 y 10 y que, cuanto mayor es, mejor permite, sin riesgo a error, averiguar y calcular los parámetros de sus enemigos.
- Alcance mínimo y máximo: En el caso de los cañones, contado en número de celdas, marca el radio de alcance de sus posibles ataques, y por tanto de su visión.
- Radio: En el caso de los cañones, marca el número de celdas adyacentes que pueden sufrir daños colaterales con un disparo suyo. El

mínimo es una celda (la celda que se ataca) y el máximo son nueve (la celda que se ataca y sus ocho adyacentes).

Con estas definiciones ya estamos en disposición de conocer más a fondo el funcionamiento del algoritmo de simulación para los distintos tipos de unidades.

Unidad terrestre

La unidad terrestre, en el momento de efectuar su turno de juego, selecciona qué paso dar de la siguiente manera:

- 1) En primer lugar mira las celdas que la rodean, en todos los puntos cardinales posibles (NW, N, NE, W, E, SW, S y SE) y calcula el coste de avanzar hacia alguna de ellas, e incluso el coste de permanecer en la celda actual.
- 2) Este coste es calculado de la siguiente manera, si la celda no es accesible o es naval y la unidad no lo es (o viceversa), o si la celda está ocupada por otra unidad, el coste devuelto será “infinito”. En caso de que esto no suceda, calcula la transitabilidad, y considera el coste de ir a esa celda, con la fórmula: $\text{coste} = 10 - \text{transitabilidad} / 2$. De esta manera se premia el ir hacia celdas más transitables.
- 3) A continuación comprueba si esa celda está entre las que la unidad recuerda como ya visitadas. Si así es, el coste se incrementará en $\text{transitabilidad} / 3$. Así logramos incentivar el que las unidades visiten celdas no visitadas con anterioridad.
- 4) Ahora comprueba si se encuentra en una zona “amiga”, es decir, si esa celda es adyacente a una en la que se encuentre un compañero de su bando. Si así es, y nuestra unidad protagonista tiene el parámetro “solidaridad”, el coste se reducirá a la mitad. Si es una zona “amiga”, pero la unidad no es solidaria, el coste sólo se reducirá dividiéndose por la raíz cuadrada de 2. De esta manera se incentiva que las unidades tiendan a permanecer unidas en grupo, y más cuanto más solidarias son.
- 5) Si la celda que se está estudiando es una celda en la que nuestra unidad tendrá al alcance un enemigo, su coste se reducirá en el valor del parámetro “ataque” multiplicado por la inversa de la raíz cuadrada de la distancia mínima hasta el enemigo. Si el avanzar hacia esa casilla, también implica que se estará al alcance de algún enemigo, el coste se incrementará multiplicándose por 1.1 unidades. De esta manera estamos haciendo que, cuanto más atacante sea nuestra unidad, más se aproximará al enemigo, aunque con cierta precaución. Esta precaución es la que se logra con el incremento por 1.1 unidades, que también se efectúa en el caso de que la celda a la que se accede no tiene un enemigo al alcance, pero si puedes ser blanco de él.
- 6) Si la unidad tiene en ese momento la misión de ayudar a un compañero, calcula si el enemigo de dicho compañero para el que le ha pedido ayuda para vencerlo está a una distancia menor a su visión, es decir, si en dicha celda ese enemigo estará a su alcance, si así es, rehace los cálculos, tomando el coste anterior al punto 5 y restándole el ataque multiplicado por la inversa de la raíz cuadrada de la distancia al

enemigo. Si la distancia a ese enemigo es menor desde la casilla que estamos estudiando que desde la que nuestra unidad se encuentra actualmente, su coste se reduce dividiéndose por dos veces la diferencia de esa distancia más uno. En caso contrario, el coste se multiplica por dos veces esa distancia más uno. Este paso sirve para que la unidad haga predominar la ayuda a su compañera si tiene esa visión, y para premiar el acercamiento al enemigo pues cuanto más cerca esté de él más efectivo será su ataque.

- 7) Si en cambio, la unidad no está ayudando a una compañera, se comprueba su memoria de celdas donde sus compañeros han entrado en batalla con el enemigo. Para cada una de esas celdas, se comprueba si desde la celda en estudio se puede atacar a alguno de esos enemigos, en cuyo caso calcula el coste anterior al punto 5 y le resta el ataque por la inversa de la raíz cuadrada de la distancia mínima al enemigo. A continuación comprueba si la distancia desde esa celda es menor a la actual, en cuyo caso divide el coste por 1.1 veces la diferencia de la distancia más 1, en caso de que la distancia no sea menor multiplica el coste por 1.1 veces la diferencia de la distancia más 1. Este proceso se repite para todas las celdas de ataque que tiene en memoria, terminando por tomar el coste menor a todas ellas. Este proceso sirve para premiar el ataque al enemigo y para intentar acercarse lo más posible a él.
- 8) Con esto ya tenemos calculados los costes de acceder a las celdas adyacentes o de permanecer en la actual. Ahora se calcula el coste de disparar. En caso de que haya un enemigo al alcance horizontalmente y según los cálculos efectuados por la inteligencia de la unidad, ese enemigo no puede alcanzarnos, su coste es menos infinito. En caso de que sí pueda alcanzar a la unidad el enemigo, el coste adquiere el valor de -10. De esta manera premiamos el ataque de las unidades, principalmente si este ataque puede efectuarse sin temor a ser represaliado.
- 9) Ya estamos en disposición de calcular el coste mínimo. Si este coste mínimo es disparar, se calcula el posible ataque. Este proceso consiste en que la unidad, mediante su inteligencia, intenta calcular los parámetros del enemigo (vida, ataque y defensa). En caso de que la unidad sea superior al enemigo en alguno de estos parámetros, procederá al ataque, procediendo en primer lugar, si está ayudando a una compañera, a atacar al enemigo de dicha compañera por delante del resto de posibles enemigos.
- 10) En el ataque influyen distintos factores. El ataque tendrá un valor de ataque por peso de la distancia, a los que se le suma la moral y la transitabilidad de la celda, todo ello dividido por dos y posteriormente multiplicado por un número aleatorio, siendo el peso de la distancia, la inversa de la raíz cuadrada de la distancia al enemigo. Se calcula un umbral que es la defensa del enemigo, más su moral, más la transitabilidad de su celda, dividido todo ello por dos y multiplicado por un número aleatorio. En caso de que el valor del ataque sea superior al del umbral, el disparo será certero y se restará una unidad de vida al enemigo, en caso contrario no se producirá daño alguno. Se ve, por tanto, como la batalla tiene un punto de aleatoriedad, aunque siempre modulado de manera importante por el resto de parámetros.

- 11) En caso de que no se opte por disparar aunque el coste mínimo sea ese, porque los parámetros del enemigo son superiores a los de la unidad, se comprueba si puede atacarse verticalmente a algún enemigo aéreo, en cuyo caso se procede de igual manera que en un ataque horizontal. Como se puede ver, se incentiva el ataque en el mismo plano sobre el ataque entre distintas alturas.
- 12) Finalmente, si el coste mínimo no es un disparo, se procede a realizar el movimiento correspondiente a ese coste mínimo.
- 13) Ya sí, por último, cabe destacar que se lleva una estadística por cada unidad del número de veces que efectúa ataques certeros, ataques fallidos, defensas realizadas correctamente, defensas fallidas así como número de bajas producidas al enemigo. Estas estadísticas sirven para incrementar, una vez finalizados todos los turnos de un mapa, o decrementar valores como la moral, el ataque, la defensa o la inteligencia. De igual forma ocurre de manera global con un bando en cuanto a sus estadísticas conjuntas. Así logramos que las unidades varíen sus comportamientos en el transcurso de la batalla, mejorando cuando actúa correctamente, y empeorando cuando esto no sucede.
- 14) Cabe destacar que, cuando una unidad es atacada y su vida baja de valor 3, pide ayuda a la compañera más cercana para que sea socorrida, ayuda que puede ser prestada o no según la situación y las características de sus compañeras.

Unidad aérea

En el caso de las unidades aéreas, su comportamiento es exactamente igual al de las terrestres, predominando su ataque horizontal a otras unidades aéreas antes que verticalmente. Hay que destacar, eso sí, dos salvedades a modo de restricciones que tienen este tipo de unidades:

- No pueden permanecer en la misma celda más de un turno, salvo que sea para disparar, en cuyo caso, una vez que ha efectuado el disparo, su siguiente turno debe desplazarse a una celda adyacente. Esto cobra sentido por el hecho de que una unidad aérea no puede permanecer suspendida en el aire, ha de estar desplazándose continuamente.
- De igual forma, no pueden disparar en dos turnos consecutivos (por el hecho de no permanecer en la misma celda), ha de permanecer dos turnos sin poder disparar. Esto se ha hecho así, para que puedan acometer vuelos en los que efectúen un disparo vertical, se alejen una celda y puedan volver sobre sus pasos y disparar en el siguiente turno nuevamente.

Cañón

Por su parte, el cañón tiene la misma particularidad que la unidad aérea, ha de permanecer dos turnos sin poder disparar. Esto es por el hecho de que un cañón tarda tiempo en volver a cargarse, y sobre todo para no dotar al cañón de una fuerza excesivamente superior a la del resto de unidades, que ya de por sí tiene, tal y como se explicará ahora. Por motivos similares, un cañón no puede desplazarse de su emplazamiento inicial.

Un cañón, por tanto, sólo puede disparar horizontal o verticalmente. Aún así, su visión es diferente a la del resto de unidades. En lugar de alcanzar un radio de celdas adyacentes, su visión se efectúa entre una distancia mínima y una máxima, lo que genera que, como mínimo, no pueda disparar en sus celdas adyacentes. ¿Esto por qué? Muy sencillo, al disparar un cañón, éste carga contra un enemigo, pero puede producir efectos colaterales a las unidades adyacentes a la atacada, dependiendo del parámetro “radio”. Por ejemplo, si el radio de daños colaterales de un cañón es 1 (el mínimo), sólo afectará a la unidad atacada. Sin embargo, si su radio es 5, afectará a la unidad atacada y a 4 celdas adyacentes tomadas aleatoriamente. Se puede ver fácilmente que el mayor valor que puede tomar el parámetro “radio” es 9. Son estos daños colaterales los que impiden que un cañón pueda disparar a una celda adyacente a ella, pues ella misma podría sufrir los daños. Esto además da una ventaja al resto de unidades, pues para poder atacar libremente a un cañón, pueden aproximarse lo más posible a ella, donde estarán a salvo de sus disparos.

Cabe destacar que a la hora de elegir qué unidad atacar, los cañones tienden a atacar ya no sólo a la unidad que consideran más débil, sino también a aquella que se encuentre rodeada de más unidades para poder generar mayores daños colaterales.

También es importante reseñar que el disparo de un cañón no puede causar daños colaterales a unidades de su mismo bando.

Capítulo 5: Implementación

5.1. Implementación.....	101
5.2. La interfaz gráfica	102
5.3. Motor de juego	108
5.4. Pruebas.....	113

5.1. Implementación

En los siguientes subapartados se analizará la implementación de cada uno de los bloques descritos en el diseño. Expondremos la estructura de clases generada, señalando las bibliotecas externas empleadas cuando corresponda, y los puntos más significativos de la dinámica de la aplicación.

El objetivo fundamental consistirá en explicar la arquitectura de la aplicación realizada, su estructuración en módulos o paquetes, aquellas estructuras de datos que sean fundamentales así como los desarrollos algorítmicos menos triviales. Con toda esta información se pretende mostrar al lector una especie de plano o guía que le permita obtener una visión general del programa, así como explicaciones a un nivel de detalle más bajo de aquellos puntos del programa que no resulten tan obvios a primera vista.

5.2. La interfaz gráfica

En el siguiente diagrama UML se muestran las clases más significativas en la estructura software del módulo de interfaz gráfica. Para una referencia completa, el lector puede consultar el Pliego II de este documento: Planos.

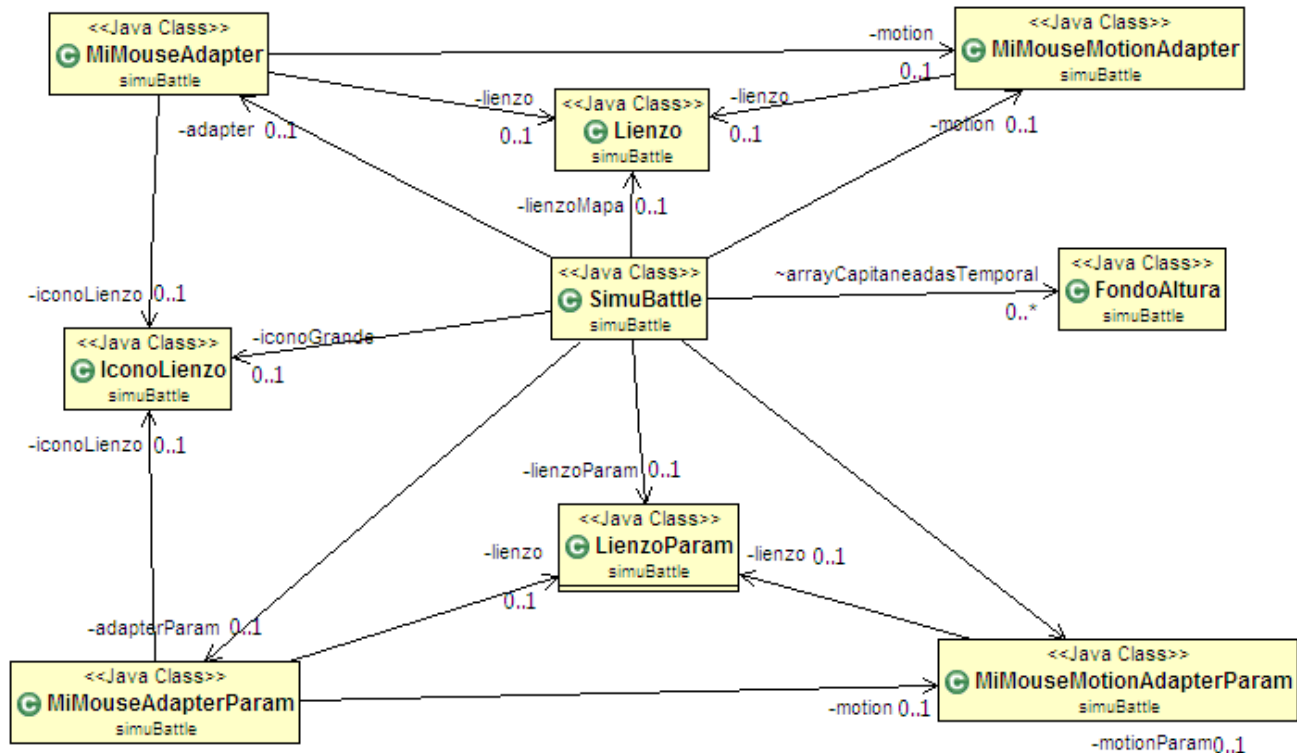


Figura 5.1. Diagrama UML interfaz gráfica

Toda la funcionalidad desarrollada para la herramienta se encuentra dentro del paquete **simuBattle**. La clase central del paquete (y de la aplicación completa) es *SimuBattle*. Esta clase contiene el método *main* y se encarga de presentar la interfaz gráfica en pantalla así como de gestionar los flujos de interacción con el usuario proporcionando:

- Los controles que el usuario puede accionar (botones, menús, ventanas emergentes *popup*...).
- El soporte necesario para otros objetos que necesitan residir en un contenedor gráfico.
- Los mecanismos de realimentación hacia el usuario (cuadros de diálogo, mensajes de error, etc.).
- Todas las acciones que el usuario es capaz de realizar y sus correspondientes condiciones de habilitación y deshabilitación para poder actualizar su estado de forma dinámica durante el uso de la herramienta.

En resumen, se puede afirmar que *SimuBattle* implementa lo que con anterioridad se ha pasado a denominar “interfaz gráfica” o “interfaz de usuario” (Figura 4.1.). En la Figura 5.2. se muestra una vista de la interfaz de usuario que proporciona esta clase.

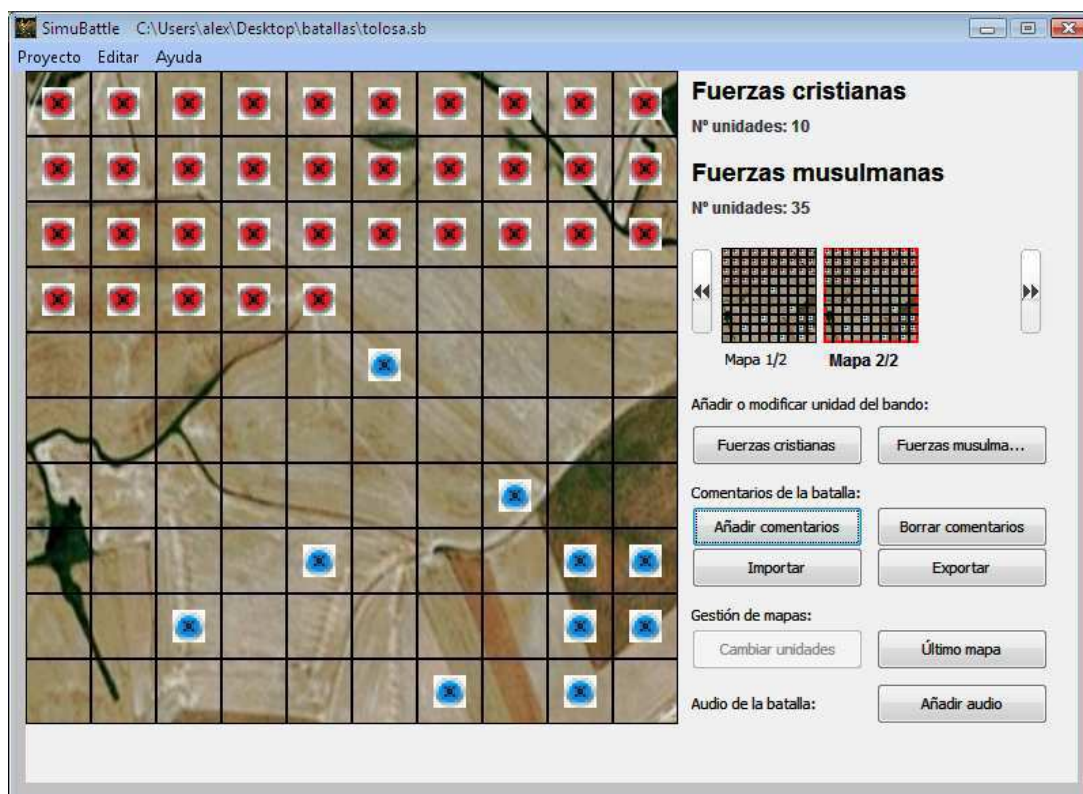


Figura 5.2. Interfaz de usuario SimuBattle

Las operaciones principales o primarias que el usuario puede realizar en cada momento de la ejecución del programa, se encuentran disponibles en el menú superior. Adicionalmente, las operaciones secundarias y específicas de cada apartado de la aplicación, se presentan en la parte derecha de la pantalla, siempre al lado del mapa de la batalla, cambiando según las posibilidades de cada momento.

En la parte izquierda de la aplicación se puede observar el mapa de la batalla. Este mapa varía según la actividad que se esté realizando en cada momento (edición, visualización, parametrización o simulación). Este mapa no es una imagen estática, sino que es interactiva y, dependiendo de la labor realizada, es posible realizar diferentes acciones pinchando sobre sus celdas y/o unidades.

Cuando se quiere acceder al diseño y edición de comentarios de la batalla, pulsando los botones correspondientes, se abre el editor de comentarios, cuya interfaz ya se mostraba en las figuras 4.11. y 4.12., y cuya definición está contenida en la clase *FrmMain*.

Ésta es la interfaz del editor de comentarios que, como ya se ha comentado con anterioridad, no se ha desarrollado desde cero para los propósitos de este

proyecto, sino que consiste en la adaptación e integración de un editor de texto ya existente con el código propio de la herramienta. Este editor se llama SimplyHTML [47] y está compuesto por los siguientes paquetes:

- com.lightdev.app.shtm
- com.sun.demo
- de.calcom.cclib.text

La clase *FrmMain* está contenida en el primero de los paquetes, y es llamada desde SimuBattle a través de la clase *App*, contenida en el mismo paquete, de manera estática. Esto es así porque el editor de comentarios funciona en paralelo a SimuBattle, casi podría afirmarse que de manera independiente, pues los comentarios son guardados físicamente en archivos HTML, no manteniendo referencia a ellos en memoria por parte de SimuBattle. La clase *FrmMain* es la encargada de construir la interfaz gráfica, así como de recibir y gestionar los eventos generados por el usuario en su interacción con la misma. Para conocer más detalles sobre la aplicación en concreto, el lector puede dirigirse a la documentación oficial en [47]. En lo relativo a las adaptaciones necesarias llevadas a cabo en la aplicación, éstas han sido ya descritas con anterioridad y en ningún caso han implicado una modificación en la estructura de clases del editor original.

Por ello, a partir de este momento, el editor de comentarios será tratado como un todo, como una especie de caja negra, motivo por el cual no se va a hacer una descripción de su estructura interna. En cambio, sí se expondrán, siempre y cuando sea considerado oportuno, aquellos aspectos relativos a la implementación del código de las adaptaciones que puedan resultar interesantes para el lector y que puedan aportar riqueza a este documento y arrojar luz acerca de la estructura interna de SimuBattle.

Como se puede apreciar en la Figura 5.2., y al igual que ocurre con la mayoría de interfaces de editores de texto, todas las operaciones que puede realizar el usuario se encuentran disponibles en la barra de menús, y además, algunas de ellas también desde la barra de herramientas y desde teclas de acceso rápido predefinidas. Las funcionalidades específicas añadidas a esta interfaz son la inclusión de vídeos del portal YouTube y la asociación de comentarios a unidades específicas. El área de trabajo ocupa el área central de la ventana. En ella se presenta al usuario el documento con el que se está trabajando y se permite su edición.

Con anterioridad, ya se citó brevemente que la manera en que la aplicación almacena los comentarios de las batallas, es físicamente en ficheros HTML, en una carpeta temporal que crea la propia aplicación mientras está en funcionamiento. De esta manera se logran los siguientes objetivos:

- Práctica independencia del editor de comentarios con respecto a la aplicación SimuBattle.
- Se pueden crear diferentes archivos HTML filtrados por características, siguiendo una estructura de ficheros interna, para poder mostrarlos por pantalla. Por poner un ejemplo, aunque el usuario trabaje sobre un fichero HTML en el editor de comentarios, éste no es el mismo que se muestra durante la visualización de la batalla (se filtra el texto para limpiarlo de comentarios asociados a las unidades). De igual forma, así

también pueden visualizarse los comentarios en el navegador por defecto que el usuario tiene definido en su ordenador.

- El almacenamiento posterior en un archivo de tipo SimuBattle, es mucho más sencillo, pues se comprimen los ficheros HTML en un archivo zip genérico, junto a la estructura de objetos del proyecto, y se renombra con extensión sb.

Llegados a este punto, no merece la pena emprender una exposición más detallada del proceso de implementación ya que, al margen de lo comentado sobre el uso del editor de texto, se trata de una tediosa sucesión de declaraciones de paneles, menús, botones, escuchadores y códigos de acción, etc.

Aún así, el lector interesado en una descripción exhaustiva de la funcionalidad de la interfaz de usuario, a más alto nivel, puede consultar el “Manual de Usuario” que se encuentra en el Pliego III de este proyecto.

Sin embargo, como consecuencia de ejercer como clase principal de la aplicación, *SimuBattle* se responsabiliza, como respuesta ante la gestión de los eventos generados por las acciones del usuario, del control del flujo sobre el resto de clases. Este punto resulta de especial relevancia para la comprensión del conjunto. Cabe resaltar también que, cuando el editor de comentarios se encuentra en uso, la clase *FrmMain* toma el relevo y es ella quien gestiona los eventos generados por su propia interfaz hasta que el control vuelve a *SimuBattle*.

La estructura gráfica de importancia capital en SimuBattle es, sin lugar a dudas, el mapa de la batalla. Para representarlo se optó por la creación de la clase *Lienzo*, que podemos observar en el diagrama UML de la Figura 5.1. *Lienzo* extiende de la clase *Canvas*, que se define como una zona de dibujo o lienzo (de ahí el nombre de la clase en SimuBattle), pues es una zona rectangular vacía de la pantalla sobre la cual una aplicación puede pintar, imitando el lienzo sobre el que un artista plasma su arte, o desde la cual una aplicación puede recuperar eventos producidos por acciones del usuario. Es un componente básico que captura eventos de exposición, de ratón y demás eventos relacionados. La clase base *Canvas* no responde a estos eventos, pero se pueden extender a esa clase base creando subclases en las que se controlan los mismos.

Las características aquí expuestas de la clase *Canvas*, encajan a la perfección con las necesidades de nuestro mapa gráfico, de ahí su implementación a través de la clase *Lienzo*. Nos permite representar fácilmente nuestra imagen del terreno de la batalla, así como la división en celdas al poder pintar sobre dicho panel. Por otro lado, podemos generar y capturar distintos eventos de ratón que se produzcan sobre el mismo, logrando así un mapa interactivo sobre el que trabajar fácilmente, pudiendo seleccionar celdas y unidades concretas.

Estos eventos son gestionados a través de las clases *MiMouseAdapter* y *MiMouseMotionAdapter*, quedando por tanto la clase *Lienzo* destinada a la plasmación por pantalla de los elementos gráficos del mapa. *MiMouseAdapter* implementa métodos de gestión de eventos de ratón tales como *mouseClicked*, *mouseEntered* y *mouseExited*, necesarios para la interactividad del mapa y la edición correcta de un mapa de una batalla. Por su parte,

MiMouseMotionAdapter está destinado a los eventos de movimiento de ratón sobre el mapa, implementando para ello el método *mouseMoved*.

Cabe destacar, que dadas las características propias de cada una de las cuatro funcionalidades básicas de SimuBattle (edición, visualización, parametrización y simulación), las clases *Lienzo*, *MiMouseAdapter* y *MiMouseMotionAdapter* son empleadas para el pintado de los Mapas de edición, visualización y simulación. Por su parte, para el pintado del mapa de parametrización, se emplean las clases *LienzoParam*, *MiMouseAdapterParam* y *MiMouseMotionAdapterParam*. Estas tres clases tienen una estructura completamente similar a las anteriormente explicadas, con la salvedad de que su funcionalidad está destinada a las necesidades propias, y diferentes, del mapa de parametrización, donde, por ejemplo, es posible seleccionar una unidad o una celda, mientras que en los otros mapas sólo es posible seleccionar celdas.

La utilización de *Canvas* presenta un inconveniente notable, consistente en la forma de pintado de sus componentes. Todo componente *Canvas* (y *Lienzo* y *LienzoParam* lo son), requieren sobrescribir el método *repaint*, al que se llama cuando es necesario que un componente se repinte. Otros componentes normales como *JTable*, *JButton* o *JLabel* no requieren su llamada, pues cuando se cambia algo de dichos componentes, automáticamente se repintan, cosa que no sucede con *Canvas*. Esta llamada a *repaint*, lo que efectúa es un aviso a la máquina virtual java de que dicho componente gráfico necesita ser repintado. El método en sí mismo no borra ni dibuja nada, la máquina virtual encola dicha petición en su cola de eventos de pantalla, junto a eventos de teclado y ratón. Llegado el momento oportuno, se comienza el redibujado en un hilo awt distinto. Cuando la máquina virtual decide repintar el componente, lo que hace es llamar a su método *update*, que, por defecto, borra el componente por completo, rellenándolo con un rectángulo del tamaño del componente y del color de fondo. Inmediatamente después, el método *update* llama al método *paint*, que es el que realmente dibuja el componente.

Esta forma de proceder ocasiona un efecto visual: en cada repintado se aprecia un parpadeo evidente del componente, al borrarse y volver a pintarse desde cero. La clase *Lienzo* que pinta nuestro mapa de la batalla, necesita repintarse constantemente con cada movimiento del ratón sobre el mismo o con cada movimiento de una unidad durante la visualización de la batalla, por lo que se requiere una solución para evitar este efecto óptico poco deseable.

La solución pasa por emplear la técnica de Doble Buffer, consistente en que se pinte toda la estructura en un objeto diferente de tipo *Image*. Una vez terminado se “pega” esa imagen directamente sobre el componente *Canvas*. Al no borrar el componente y dibujar la imagen directamente encima, se evita el parpadeo. Recibe el nombre de doble buffer porque un buffer son los píxeles en pantalla y el otro es el objeto *Image*. Lo que se hace es dibujar sobre uno de ellos (el de tipo *Image*), evitando hacerlo directamente sobre la pantalla. Y sólo pegar directamente el resultado en la pantalla una vez completado en el primer buffer.

La estructura escogida para pintar las distintas unidades, es la clase *IconoLienzo*, que hereda también de *Canvas*, y que mantiene un esqueleto muy sencillo, con campos como la ruta de la imagen y las coordenadas x e y donde se sitúa la unidad.

Una estructura diferente, a la hora de representar los mapas, es llevada a cabo en las ventanas de elección de unidades al seleccionar una unidad a capitanear o a atacar durante la simulación, y de igual forma al asociar comentarios a una unidad en el editor de comentarios. En estos casos, el mapa presentado en la pantalla de menor tamaño, sigue una estructura de etiquetas *JLabel*. En este caso, el mapa y las distintas unidades son etiquetas superpuestas unas sobre otras, pintadas en distintos niveles de altura o capas, gracias a la clase *JLayeredPane*, que es un componente Swing que proporciona una tercera dimensión para posicionar componentes en profundidad. Para facilitar esta representación, es por lo que se creó la clase *FondoAltura*, que encapsula la estructura de capas y así mostrar las distintas unidades.

Cabe destacar, también, la existencia de la clase *AppCloser*, contenida dentro de *SimuBattle* y que sirve para gestionar los eventos de cierre de la ventana principal del programa.

Se ha optado por incluir en este bloque las bibliotecas encargadas de reproducir sonidos mp3, que aunque no se ciñan estrictamente a la definición de interfaz gráfica, sí que cumplen el cometido de interfaz de usuario. Para dar solución a la reproducción de ficheros mp3 en Java, se ha optado por usar la implementación *Tritonus* [50], que se distribuye con licencia pública y que utiliza *Java Sound API*. También empleamos la biblioteca de reproducción de mp3 *MP3SPI* [51], y para simplificar todo el proceso, la biblioteca *BasicPlayer* que proporciona una API de alto nivel para no tener que programar las funciones básicas [52]. Así pues, con estas bibliotecas, ya podemos disponer de métodos que encapsulan toda la funcionalidad de reproducción de sonidos, sin tener que modificarlos.

5.3. Motor de juego

El siguiente diagrama de clases contiene una representación resumida de la implementación final de la lógica de negocio. Para una referencia completa, el lector puede consultar el Pliego II: Planos.

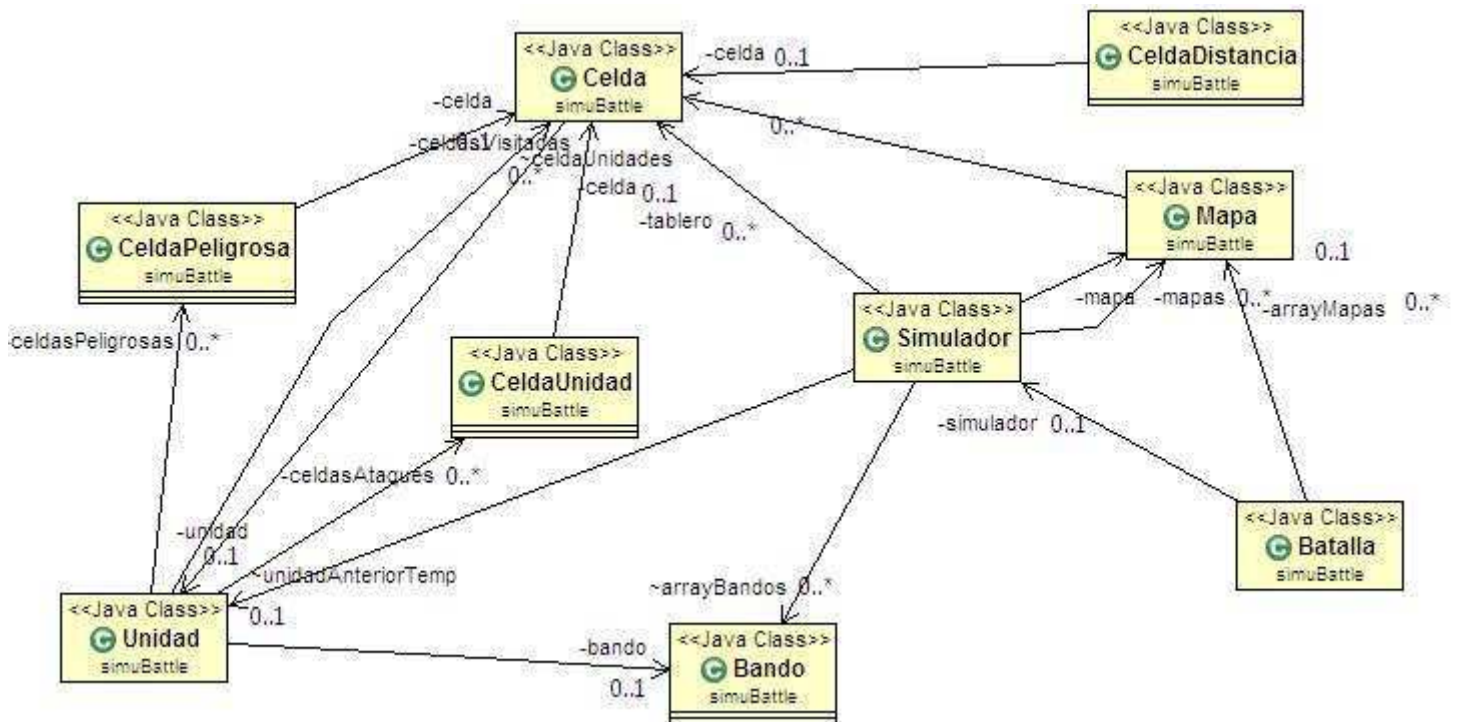


Figura 5.3 Diagrama UML motor de juego

En el apartado 4.5. se describieron las funciones que debía realizar este bloque, entre las que destaca por encima de todas el algoritmo de simulación. Este algoritmo ya fue detallado anteriormente en el apartado 4.6. Es importante aclarar antes de empezar este apartado, que el objetivo de la descripción aquí expuesta no es un listado exhaustivo de los métodos y variables de estas clases, sino una descripción aclaratoria de las mismas y sus puntos esenciales. Del mismo modo, el diagrama mostrado en la Figura 5.4. es también un diagrama resumido en el que sólo están reflejadas las clases esenciales.

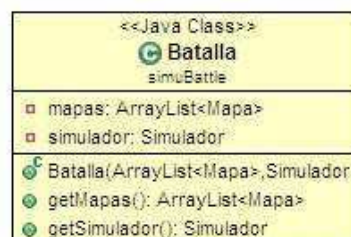


Figura 5.4. Clase Batalla

La clase principal de la estructura de nuestro modelo de datos, es, sin lugar a dudas, la clase *Batalla*. A pesar de su importancia, su funcionamiento es muy simple, pues simplemente encapsula, tal y como vemos en la Figura 5.5., las estructuras más complejas y que dotan de auténtica funcionalidad a la aplicación.

Como puede observarse, la clase *Batalla* simplemente engloba un objeto *Simulador* y un conjunto de objetos de la clase *Mapa*. Estas dos clases son las auténticamente importantes, pero la funcionalidad real de la clase *Batalla* reside precisamente en eso, en servir de aglutinador del resto de las estructuras, facilitando su manejo a la hora de leer y guardar sus datos. Una batalla, según la estructura aquí realizada, está formada por una sucesión de mapas y por una simulación.

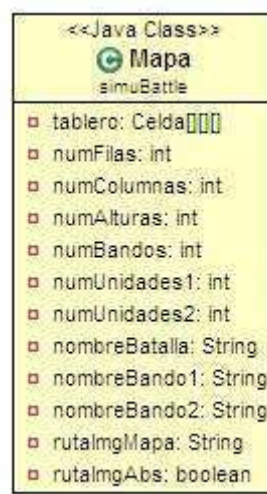


Figura 5.5. Clase Mapa

La clase *Mapa*, por su parte, contiene la información necesaria para definir completamente su estructura: número de bandos, de unidades de cada bando, los nombres, la imagen del mapa o las dimensiones (x, y, z) del mapa-tablero. Mención especial merece, precisamente, esta estructura, pues el mapa contiene una matriz de tres dimensiones de objetos *Celda*, y es la que recrea la estructura del tablero.



Figura 5.6. Clase Celda

Por su parte, un objeto *Celda* está constituido por sus características básicas, como las coordenadas que ocupa en el mapa-tablero, y los parámetros que la definen: accesibilidad, transitabilidad y si es o no naval. Asimismo, una *Celda* puede contener, o no, una *Unidad* y sólo una como máximo.

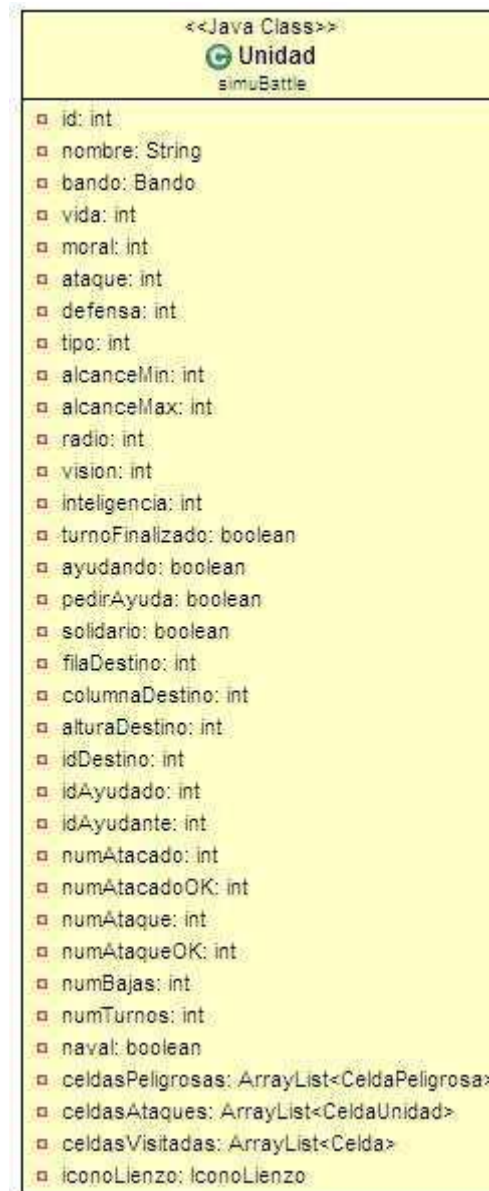


Figura 5.7. Clase Unidad

Un objeto *Unidad* contiene elementos identificadores, como un ID y un nombre, al igual que sus características parametrizables y a las que ya hemos hecho referencia en varias ocasiones (vida, moral, ataque, defensa...), variables destinadas a la gestión de ayuda a unidades amigas (ayudando, pedirAyuda, idDestino...), así como listas de celdas que sirven a la unidad para mantener una memoria de las celdas ya visitadas, celdas donde ha sufrido ataques o celdas donde alguna compañera está enfrentándose a un rival. Las estructuras *CeldaUnidad* y *CeldaPeligrosa* no son más que estructuras

formadas por un objeto *Celda* e identificadores de la unidad que ocupan dicha celda, como el ID en el caso de *CeldaUnidad*, o las coordenadas de dónde se encuentra la unidad que ha atacado en dicha Celda (*CeldaPeligrosa*). Finalmente, cada unidad pertenece a un *Bando*.



Figura 5.8. Clase Bando

Finalmente, el objeto *Bando* representa el conjunto de unidades de batalla que se enfrentan unidas contra el enemigo, y sus campos son simplemente un identificador y un conjunto de variables que sirven de estadísticas para el algoritmo de simulación.

Como se ha podido comprobar, hemos realizado un repaso a la casi totalidad de clases que conforman el motor de juego, la estructura de la lógica de negocio de SimuBattle. Su composición es muy sencilla, pero permite construir de una manera sólida el esqueleto de cualquier batalla.

Aún así, resta por explicar, aunque sea someramente, la estructura de la clase *Simulador*. Contiene numerosos métodos que conforman el algoritmo de simulación y que no pasaremos a detallar por su extensión, si bien se puede conocer más de cerca el funcionamiento real del algoritmo en el apartado 4.6.

En cuanto a los campos que componen la clase *Simulador*, se encuentran aquellos que determinan el mapa de la batalla, como son el número de bandos o las dimensiones del mapa. De igual forma también variables que son utilizadas por el algoritmo para llevar cuenta de ciertas estadísticas necesarias para su correcto funcionamiento, como el número de turnos sin haberse efectuado disparos, la última unidad en realizar un movimiento, o las celdas que contienen unidades y aún no han efectuado movimiento. Finalmente destacar que el objeto *Simulador* contiene una lista de objetos *Mapa*, que representa los sucesivos mapas que se van generando durante la simulación. De esta forma, es posible volver a un mapa anterior y continuar la simulación desde ese punto. En un principio, el *Mapa* inicial es exactamente igual al mapa editado con los datos introducidos durante la parametrización. En el transcurso de la simulación, si se decide cambiar la parametrización para ese mapa en concreto, los datos de las distintas unidades y celdas adoptarán los nuevos valores de la parametrización en ese mapa concreto y posteriores, no en los anteriores ya almacenados en la lista, tal y como por otra parte resulta lógico. De igual forma, la clase *Simulador* guarda en una de sus variables el *Mapa* actual en el que se encuentra la simulación.

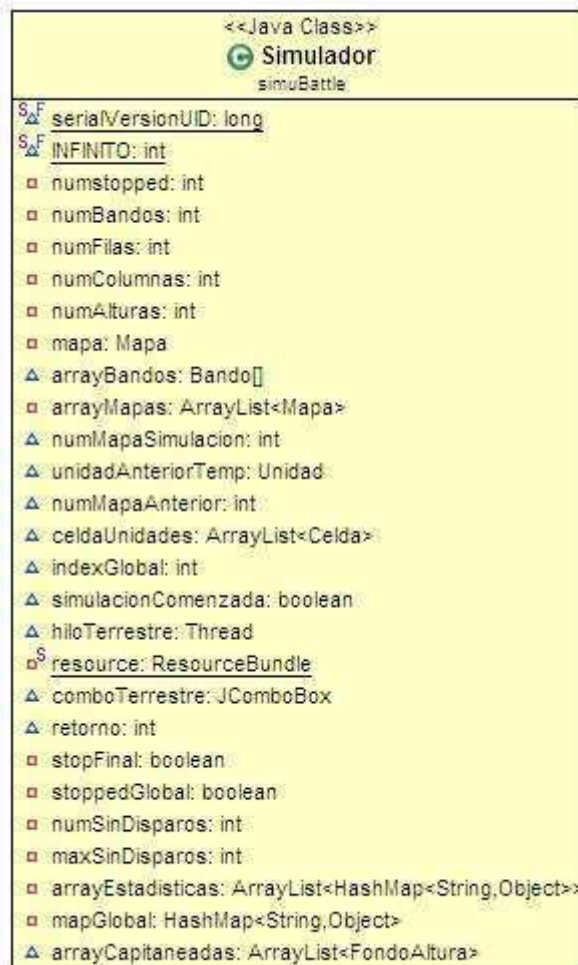


Figura 5.9. Clase Simulador

5.4. Pruebas

En los apartados anteriores se ha descrito la implementación de la que ha sido, por ahora, la versión definitiva de SimuBattle. Sin embargo, anteriormente se habían generado versiones provisionales que fueron sometidas a distintas pruebas.

Como resultado de las pruebas llevadas a cabo, fueron realizadas diferentes mejoras en la aplicación. A pesar de que la mayoría de estos cambios se han ido señalando de manera sutil a lo largo de este documento, este apartado pretende servir de recopilatorio de todos ellos. Así pues, se va a proceder a describir con brevedad en qué consistieron los dos procesos de pruebas que se efectuaron sobre la aplicación, así como los resultados obtenidos a partir de los mismos.

Los próximos párrafos no contienen la descripción de un exhaustivo plan de pruebas. Las pruebas funcionales y de integración básicas ya se habían realizado durante la implementación de la aplicación. Lo que el lector encontrará a continuación es una especie de batería de pruebas de estrés sobre la aplicación, realizada en dos fases diferentes y de modos muy distintos.

La primera fase de pruebas, consistió en presentársela a dos voluntarios, para que la utilizaran a su voluntad, siendo ésta la primera vez que lo hacían. El objetivo fundamental de esta primera fase no era otro que recabar las opiniones de los usuarios, prestando especial atención en aquellos puntos que creían mejorables, atendiendo sus sugerencias, e intentando minimizar, en la medida de lo posible, aquellos aspectos que pudieran resultar más complicados o menos intuitivos para el usuario. Se trata pues de comprobar la funcionalidad y usabilidad de SimuBattle.

Algunas de las sugerencias aportadas por los usuarios, y que finalmente fueron implementadas en SimuBattle, son:

- Asociar comentarios resumidos a la batalla, y que pudieran ser visualizados en el panel derecho, en el apartado de Visualización, ofreciendo a su vez la posibilidad de leer los comentarios completos en una pantalla mayor, optando por la de un navegador dado su potencial visual.
- A colación de la sugerencia anterior, surgió la posibilidad de asociar comentarios a movimientos específicos de unidades, y no sólo a mapas completos. Por lo que además, la visualización pasó a convertirse en una sucesión de movimientos de unidades, y no sólo de mapas globales. Esto conllevó una revisión completa de la filosofía llevada a cabo para la visualización, surgiendo la necesidad de realizar un algoritmo que especificara el orden en el que las unidades debían moverse entre un mapa y otro.
- De igual forma, se optó por asociar no sólo comentarios a movimientos de unidades, sino también el marcado de las unidades atacadas en ese movimiento.
- La inclusión de sonido durante la batalla, fue propuesta con el objetivo de dotar de una mayor capacidad multimedia a la batalla.

- La posibilidad de guardar una simulación como una batalla editable, también surgió en esta fase de pruebas, al igual que la posibilidad de parametrizar una simulación en un momento cualquiera de la misma.
- Se añadieron los botones que, mediante un código de colores, permiten de un rápido vistazo, conocer los valores de los parámetros más importantes de las unidades durante la simulación: vida, inteligencia, ataque, defensa y moral.
- Se introdujo también la posibilidad de dotar de nombre a las unidades, para mejorar el control del usuario sobre las mismas, así como facilitar la visualización y simulación de la batalla.

Durante una segunda fase, el objetivo fundamental consistió en intentar llevar al programa al límite de su funcionalidad para descubrir posibles errores o “bugs” que pudiera contener. Y, aunque esta fase se dio mayor importancia a la robustez de la aplicación, también se recabaron opiniones acerca de la aplicación y de las mejoras llevadas a cabo en la primera fase.

El desarrollo de esta segunda fase requiere el planteamiento de situaciones de considerable complejidad, y durante el desarrollo del mismo, la herramienta siguió un proceso de depuración que eliminó los pequeños errores de funcionalidad que aparecían en situaciones extremas de uso.

En cuanto a las opiniones recavadas, al margen de sugerencias de mayor calado que las efectuadas en la primera fase, y que fueron llevadas a cabo, fueron bastante favorables. Los usuarios lograban sentirse libres con la aplicación a la hora de generar sus batallas y de explicarlas a su antojo, pudiéndose apoyar de todo tipo de material multimedia. Además, calificaron de muy importante la posibilidad de que el alumno se acerque a esta aplicación desde un punto de vista lúdico, pero sin alejarse, en ningún momento, de su vertiente educativa.

Lamentablemente no disponemos de datos más específicos y estructurados sobre estas opiniones, puesto que para eso habría que haber dedicado un tiempo importante a preparar cuestionarios específicos y su posterior análisis, lo que estaba fuera del propósito de las pruebas. Sin embargo, sí se ha querido reflejar en este documento, al menos, la sensación que la herramienta provocó en sus primeros usuarios.

Capítulo 6:

Conclusiones y trabajos futuros

6.1. Conclusiones	117
6.2. Trabajos futuros.....	118

6.1. Conclusiones

Este proyecto ha desarrollado una herramienta para servir de apoyo a la enseñanza de la historia en los primeros niveles educativos. Partiendo de la premisa de intentar aunar en una misma aplicación aprendizaje y entretenimiento para los alumnos, se ha creado la herramienta SimuBattle, cuyo fin básico es permitir a profesores y alumnos generar y recrear batallas históricas, con el aliciente de poder simular distintos finales alternativos (o batallas ucrónicas) a partir de una serie de parámetros y características de las mismas.

Se ha realizado un breve estudio de las distintas teorías educativas en las que se circunscribe esta aplicación, con la pretensión de mostrar la idoneidad de su utilización en el aula.

Por otro lado, se ha realizado un recorrido por aquellas aplicaciones en las que se inspira SimuBattle, fuentes de las que inicialmente bebe. Videojuegos que recrean diferentes épocas históricas, y que requieren habilidad para gestionar un ejército o incluso un país. Estos videojuegos, cuyo fin es meramente lúdico, también pueden ser utilizados para incentivar en el alumno ciertas capacidades, pudiendo servir, por tanto, como medios vehiculares para impartir enseñanzas de la materia de historia.

De esta manera, damos el salto hacia la explicación de qué es un “serious game”, calificativo que perfectamente podría recibir SimuBattle, ya que, precisamente, nace con la misión de entretener, pero sobre todo de educar. Así, y teniendo siempre presentes los juegos de guerra (wargames) que han servido de diversión a muchas generaciones, se ha acabado configurando la estructura de esta aplicación.

Cualquier profesor puede disponer de esta herramienta, sencilla en su uso pero poderosa en cuanto a fines didácticos, complementando así sus clases de historia, apoyándose en elementos multimedia que se han demostrado muy efectivos y atrayentes para los alumnos.

Con plena consciencia de la importancia que toma la iniciativa del alumno en el proceso de adquisición de nuevos conocimientos por su parte, esta aplicación le otorga total libertad creadora, sin descuidar en ningún momento la tutela ejercida por el profesor, que no sólo es recomendable, sino esencial para la mejor comprensión de las ideas que este último desea transmitir.

SimuBattle pretende ser una ayuda al profesor en su intento de impartir sus clases de historia de una manera renovada, con un lenguaje, el multimedia, más cercano a los jóvenes, y mucho más eficaz. En definitiva, un pequeño apoyo para que profesores y alumnos interactúen unidos, facilitando a ambos la impartición de la clase de historia.

6.2. Trabajos futuros

Aquí se ha presentado una versión inicial de SimuBattle, que no deja de ser, a fin de cuentas, el punto de partida hacia una aplicación mucho más completa, pues admite continuas mejoras que permitan convertirla en una herramienta cada vez más potente y eficaz, tanto en su vertiente educativa como lúdica.

Existen, por tanto, diferentes aspectos en los que se puede continuar trabajando, y que pueden ser tanto mejoras a la aplicación actual, como modificaciones estructurales que vendrían a completar de manera definitiva a la herramienta. Entre ellas cabe citar las siguientes:

- Mejora del grafismo en la recreación de las batallas. Sería muy interesante dotar de mayor fuerza visual la representación tanto de los mapas como de las unidades. Sería deseable desprenderse de la estructura clásica de tablero y fichas que contiene actualmente, y presentar a las unidades como personajes más reales, dotándolas de movimientos propios. Los mapas podrían presentar un zoom que permitiera, durante la visualización de la batalla, aumentar o centrar la atención en distintos puntos.
- Podrían añadirse nuevos parámetros tanto a las unidades como a las celdas para lograr así una parametrización más detallada y completa. Por ejemplo: meteorología, tipos de terreno, distintos tipos de armamento...
- Inclusión de nuevos tipos de unidades que complementen a los ya existentes: terrestre, aérea y cañón.
- Al hilo del punto anterior, sería interesante poder disponer de unidades específicas con distintos niveles de avance tecnológico según la época a la que pertenezcan.
- Modificación y mejora del algoritmo de simulación, empleando por ejemplo un motor de reglas más potente, que incluso pudiera ser configurable por el propio usuario.
- Incluir la posibilidad de que el usuario pueda “programar” el comportamiento de las unidades, consiguiendo así un motor de reglas flexible y configurable.
- Permitir que las unidades actúen de manera conjunta, conformando así un batallón como entidad propia, y haciendo que las formaciones sean flexibles para desarrollar así diferentes tácticas bélicas de ataque y defensa.
- De igual forma que una batalla está formada por varios mapas, sería muy interesante incluir la posibilidad de que un mismo proyecto albergue varias batallas, representando así toda una guerra en un único proyecto.
- Permitir que existan más de dos bandos en una batalla.

- Realizar mapas que incluyan más niveles en la dimensión z o de altura. Actualmente presenta dos planos de representación, permitiendo por tanto únicamente dos alturas: la terrestre y la aérea. Podría incluirse, como mínimo, una altura inferior a la terrestre, permitiendo así la representación de submarinos y trincheras.
- Incluir pantallas de visualización de los comentarios, embebidas en el propio programa, más cercanas, en cuanto a su aspecto y funcionalidad, a la de un navegador de web.
- Añadir la funcionalidad de arrastrar y soltar (drag and drop) para insertar y modificar unidades en el mapa de la batalla, así como para realizar los movimientos durante la simulación. Incluso sería deseable permitir acciones como copiar y cortar para poder distribuir las unidades en el mapa durante la edición de la batalla de una manera más rápida y cómoda.

Sin embargo, todas estas mejoras no deberían incumplir uno de los requisitos fundamentales de SimuBattle, que no es otro que mantener una interfaz y una funcionalidad lo más sencilla y atrayente posible, de manera que sea fácilmente utilizada por personas pertenecientes al mundo de las Humanidades, a las que no hay por qué presuponer conocimientos técnicos avanzados. En este sentido, podría resultar muy interesante la inclusión de un manual de ayuda extenso y fácilmente accesible desde la propia aplicación.

Independientemente de estas mejoras, SimuBattle podría reconvertirse en una aplicación Web 2.0. Esto permitiría guardar las batallas creadas por los usuarios en cualquier parte del mundo en una sola base de datos, permitiendo una auténtica interactividad global y haciendo de la realimentación entre distintos usuarios su mayor baza. De este modo, podrían, por ejemplo, realizarse batallas y sus simulaciones entre distintos usuarios, capitaneando incluso cada uno un bando distinto. Las tecnologías actuales tienen recursos suficientes para conseguir las capacidades gráficas necesarias, y la aplicación se beneficiaría de las ventajas de correr sobre la red (ubicuidad, accesibilidad a un público mucho más numeroso...).

Bibliografía

- [1] BOU BAUZÁ, Guillem. “*El guión multimedia*” 1997. Editorial Anaya Multimedia-Anaya Interactiva.
- [2] BOU BAUZÁ, Guillem. “*El guión multimedia*” 2003. Editorial Anaya Multimedia-Anaya Interactiva.
- [3] 3D JUEGOS.
<http://www.3djuegos.com>
- [4] RODRÍGUEZ, Fernando. “*Historia del software español de entretenimiento*” 2003
<http://www.msdox.com/reportajes/historiasoftesp210203.php>
- [5] PASCUAL, Carlos. “*Turistas en el campo de batalla*” 2007. El País.
http://www.elpais.com/articulo/viajes/Turistas/campo/batalla/elpcanviaesp/20070324elpviavje_3/Tes/
- [6] BATALLA DE LOS ARAPILES.
<http://www.losarapiles.com/Default.asp>
- [7] SEMPRÚN, ALFREDO. “*Una batalla sobre el tablero*” 2007. La Razón.
http://www.larazon.es/6559/noticia/Vivir_el_d%EDa/Una_batalla_sobre_el_tablero
- [8] DELTA EDICIONES.
<http://www.deltaediciones.com/>
- [9] ASOCIACIÓN NAPOLEÓNICA ESPAÑOLA.
<http://www.asocne.es/>
- [10] ASOCIACIÓN CULTURAL LOS SITIOS DE ZARAGOZA.
<http://www.asociacionlossitios.com/>
- [11] AEFRRH.
<http://www.imaengine.com/fehrh/>
- [12] WW1: EXPERIENCES OF AN ENGLISH SOLDIER.
<http://wwar1.blogspot.com/>
- [13] BLUMSCHEIN, P., Fischer, M. “E-learning en la formación profesional: diseño didáctico de acciones de e-learning” 2007. Editorial Cinterfor/OIT.
- [14] GALVEZ DE LA CUESTA, María del Carmen. “*Aplicaciones de los videojuegos de contenido histórico en el aula*”. Revista Icono 14. Nº 7 2006
<http://www.icono14.net/revista/num7/articulos/carmen%20galvez.pdf>
- [15] SAZ RUBIRA, José Manuel. “*Aplicación educativa de los videojuegos*” Revista Educar en el 2000. Nº 8. Abril 2004
<http://www.tecnoneet.org/docs/colabora/articulo3.pdf>

- [16] ÁGORA DIGITAL.
<http://cuentayrazon.blogcindario.com/>
- [17] INSA GHISAURA, Daniel. “*Multimedia e internet*”. 1998. Ediciones Paraninfo S.A.
- [18] ETXEBERRÍA BALERDI, Félix. “*Videojuegos y educación*”. Revista Teoría de la Educación. Nº2 Universidad de Salamanca
http://www.usal.es/~teoriaeducacion/rev_numero_02/n2_art_etxeberria.htm
- [19] GRUPO F9. “*Utilització Educativa dels jocs d’Ordinador*”. 1997
<http://www.xtec.net/~abernat/castellano/justif.htm>
- [20] GRUPO F9. “*Videojocs a l’Aula*”
<http://www.xtec.net/~abernat/castellano/propuest.htm>
- [21] CUENCA LÓPEZ, José María. “*Los videojuegos en la enseñanza de la historia*”. Ponencia del Seminario Internacional Taula d’Història. Barcelona. Julio 2007.
- [22] DUNNIGAN, James F. “*The Wargames Handbook*”. 1997. Editorial Quill
<http://www.hyw.com/Books/WargamesHandbook/Contents.htm>
- [23] KODO. “*Testing Kodo*”
<http://bdkodo.blogspot.com/2007/08/wargame-y-eso-qu-es-lo-que-es.html>
- [24] POZO, J. I. “*Teorías cognitivas del aprendizaje*”. 1993. Ediciones Morata.
- [25] CASTELLS, Manuel. “*La era de la información. Vol. III: Fin de Milenio*”. 2001. Alianza Editorial.
- [26] VERDECIA CARBALLO, Enrique. “*Edutec. Revista Electrónica de Tecnología Educativa*” Núm. 23 / Julio 07
<http://bdkodo.blogspot.com/2007/08/wargame-y-eso-qu-es-lo-que-es.html>
- [27] COON, Denis. “*Fundamentos de psicología*”. 2001. Editorial Thomson Internacional
- [28] GREGORY, Richard L. “*El compañero Oxford para la mente*”. 1987. Editorial Oxford University Press.
- [29] FONSECA, León. “*Los software educativos. Una alternativa en la actualidad*”. 2005. Editorial Pueblo y Educación.
- [30] TASCÓN TRUJILLO, Claudio. “*La página web CTT*”
<http://www.ctascon.com/>
- [31] Centro Virtual Cervantes.
http://cvc.cervantes.es/ensenanza/biblioteca_ele/diccio_ele/diccionario/cognitivismo.htm

[32] El constructivismo de Papert.

http://swiki.agro.uba.ar/small_land/157

[33] MIT Media Lab: Epistemology and Learning.

<http://el.media.mit.edu/>

[34] SÁNCHEZ GÓMEZ, María. “*Buenas prácticas en la creación de serious games*”. IV Simposio Pluridisciplinar sobre Diseño, Evaluación y Desarrollo de Contenidos Educativos Reutilizables (SPDECE). Bilbao, 2007.

[35] ABT, Clark C. “*Serious Games*”. 1970. University Press of America.

[36] X Media Lab.

<http://xmedialab.typepad.com/xmedialab/>

[37] Álvarez, J., RAMPNOUX O. “*Serious Game: ¿solo una cuestión de postura?*” AISB’07. Newcastle, Abril 2007.

[38] Night Driver. Rexona.

<http://home.siainteractive.com/demojuegos.juegos/RexonaNightDriver/RexonaNightDriver.htm>

[39] Johnnie Walker Interactive Experience

<http://home.siainteractive.com/demojuegos.juegos/JWF1/index2.htm>

[40] Rompecabezas Andalucía. Junta de Andalucía. Consejería de Turismo, Comercio y Deporte.

<http://home.siainteractive.com/demojuegos.juegos/AndaluciaRompe2/juego.php?idioma=1>

[41] Serious Games Interactive.

<http://www.seriousgames.dk/>

[42] Gamelearn.

www.gamelearn.eu

[43] REY, Roger y ROMERO, Fernando. GenMàgic.

<http://genmagic.org/tangram/index.swf>

[44] Xuegus educativos llingua asturiana

http://www.genmagic.net/llingua_asturiana/llingua_asturiana.swf

[45] Funschool. Formula Fusion.

http://funschool.kaboose.com/formula-fusion/games/game_addition_attack.html

[46] COLLER, B. D. y SCOTT, M. J. “*Effectiveness of using a video game to teach a course in mechanical engineering*”. 2009

[47] Homepage de SimplyHTML

<http://www.lightdev.com/page/3.htm>

<http://sourceforge.net/projects/simplyhtml/>

[48] API de Java

<http://java.sun.com/j2se/1.5.0/docs/api>

[49] RABIN, Steve. “*AI Game Programming Wisdom*”. 2002. Editorial Charles River Media.

[50] Tritonus: Open Source Java Sound

<http://tritonius.org/>

[51] MP3 SPI for Java Sound

<http://www.javazoom.net/mp3spi/mp3spi.html>

[52] jlGui – Music Player for the Java Platform

<http://www.javazoom.net/jlgui/sources.html>

[53] CARBALLUDE, Pablo. “*Blog: La playina del Norte*”.

<http://pablocarballude.blogspot.com/>

[54] CEBALLOS, Fco. Javier. “*Java 2. Curso de programación*”. 2000. Edidtorial Ra-Ma.

[55] ChuWiki. Wiki de lenguajes de programación.

http://www.chuidiang.com/chuwiki/index.php?title=P%C3%A1gina_Principal

Pliego II

Planos UML

<<Java Class>>

Simulador

simuBattle

serialVersionUID: long = 71206L

INFINITO: int = 214748364

numstopped: int

numBandos: int

numFilas: int

numColumnas: int

numAlturas: int

numMapaSimulacion: int

numMapaAnterior: int

indexGlobal: int

simulacionComenzada: boolean

hiloTerrestre: Thread

resource: ResourceBundle

comboTerrestre: JComboBox

retorno: int

stopFinal: boolean

stoppedGlobal: boolean

numSinDisparos: int

maxSinDisparos: int

arrayEstadisticas: ArrayList<HashMap<String, Object>> = null

mapGlobal: HashMap<String, Object>

Simulador(Mapa, ResourceBundle)

setResource(ResourceBundle): void

getSimulacionComenzada(): boolean

setSimulacionComenzada(boolean): void

<<Java Class>>

Bando

simuBattle

serialVersionUID: long = 71

idBando: int

numAtacado: int

numAtacadoOK: int

numAtaque: int

numAtaqueOK: int

numBajasPropias: int

numBajasAjenas: int

Bando(int)

Bando(Bando)

compareBando(Bando): boolean

getIdBando(): int

setIdBando(int): void

getNumAtacado(): int

setNumAtacado(int): void

incrementarNumAtacado(): void

getNumAtacadoOK(): int

setNumAtacadoOK(int): void

incrementarNumAtacadoOK(): void

getNumAtaque(): int

setNumAtaque(int): void

incrementarNumAtaque(): void

getNumAtaqueOK(): int

setNumAtaqueOK(int): void

incrementarNumAtaqueOK(): void

getNumBajasPropias(): int

setNumBajasPropias(int): void

incrementarNumBajasPropias(): void

getNumBajasAjenas(): int

setNumBajasAjenas(int): void

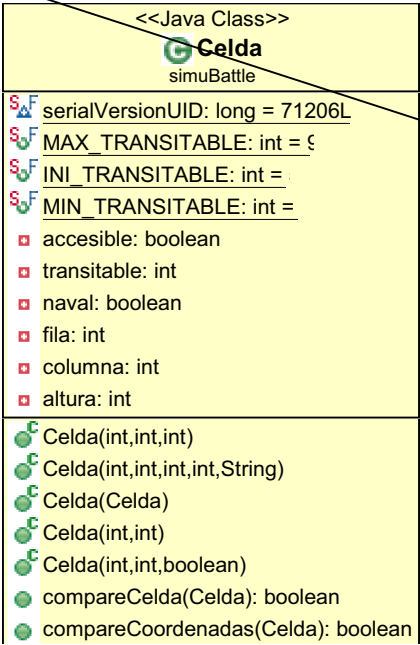
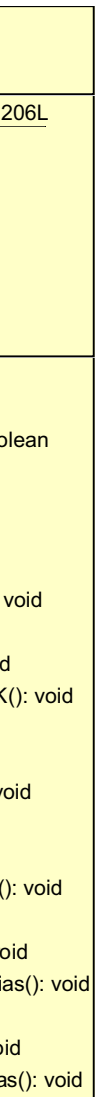
incrementarNumBajasAjenas(): void

~arrayBandos

0..*


-bando

0..1



+celdaUnidades

<<Java Class>>

 **MiMouseAdapterParam**

simuBattle

✖ numFilas: int


✖ numColumnas: int


✖ altura: int


✖ pressedIndividual: boolean

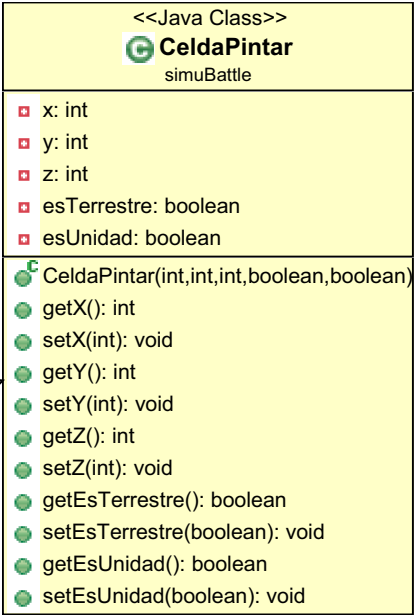
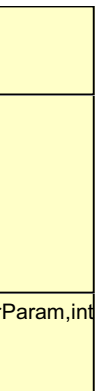
✖ esCelda: boolean

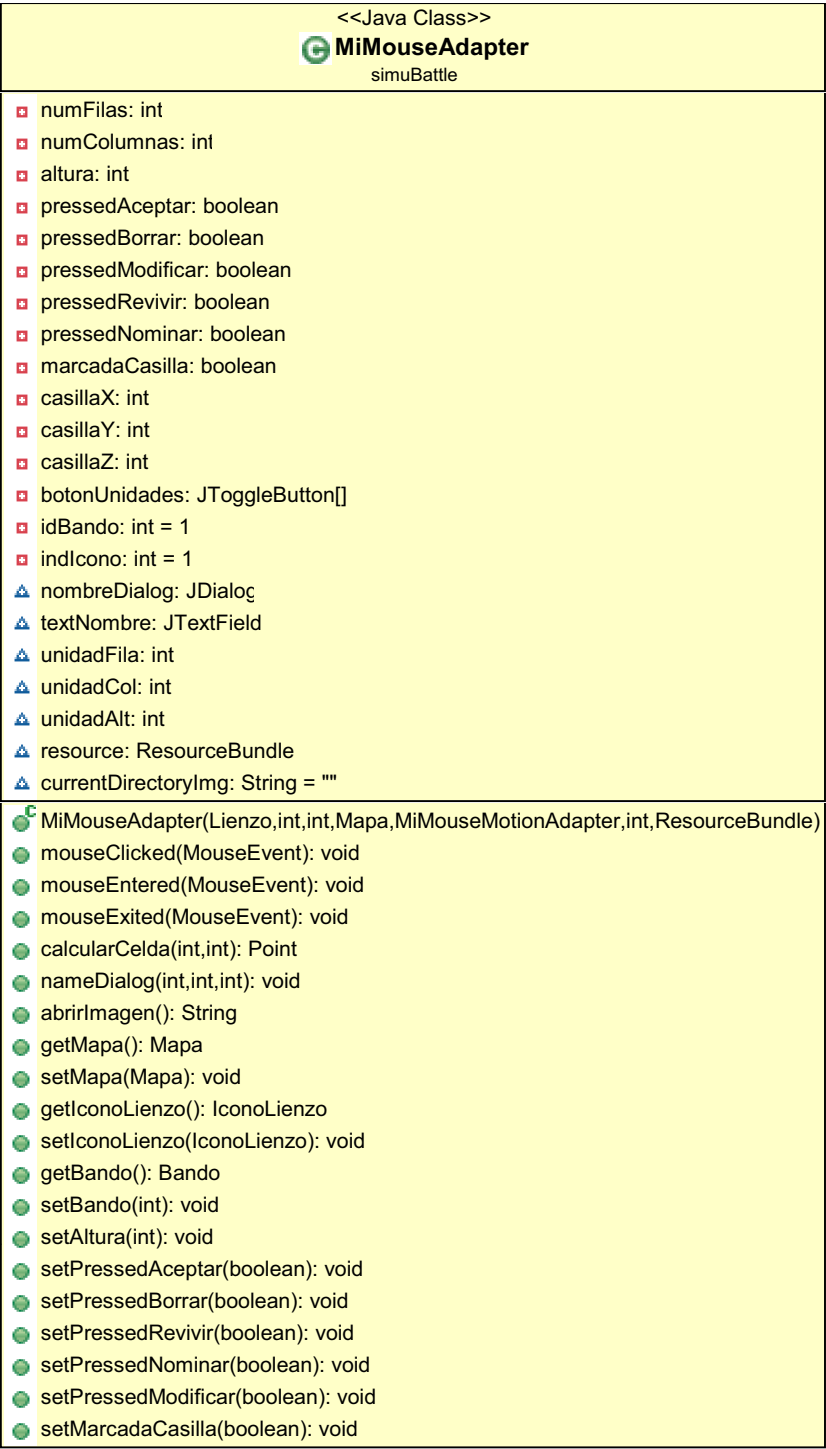
✖ esTerrestre: boolean

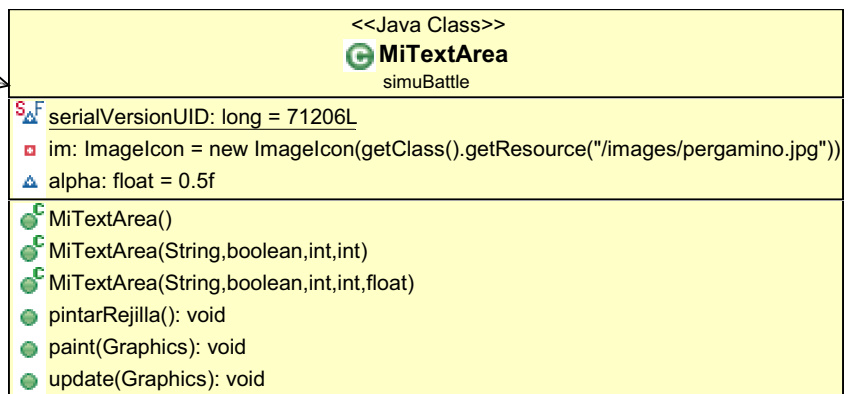
 MiMouseAdapterParam(LienzoParam,int,int,Mapa,MiMouseMotionAdapter

 mouseClicked(MouseEvent): void

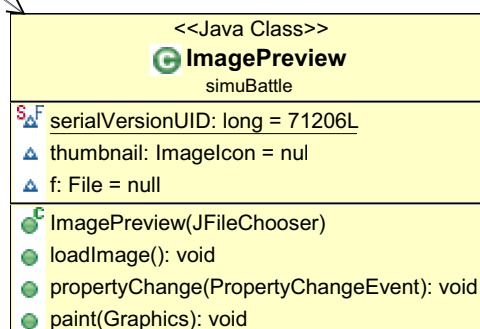
 mouseEntered(MouseEvent): void

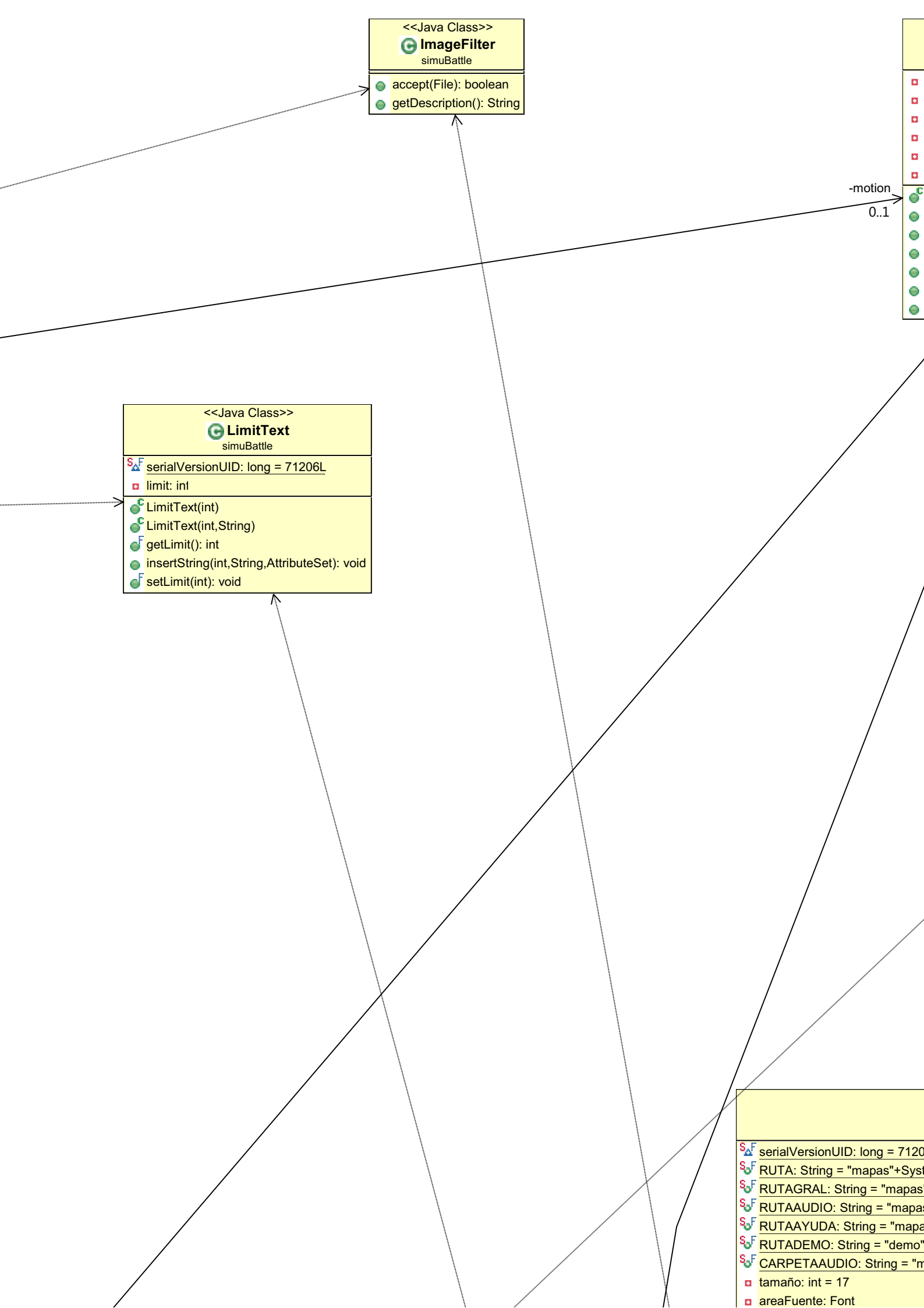


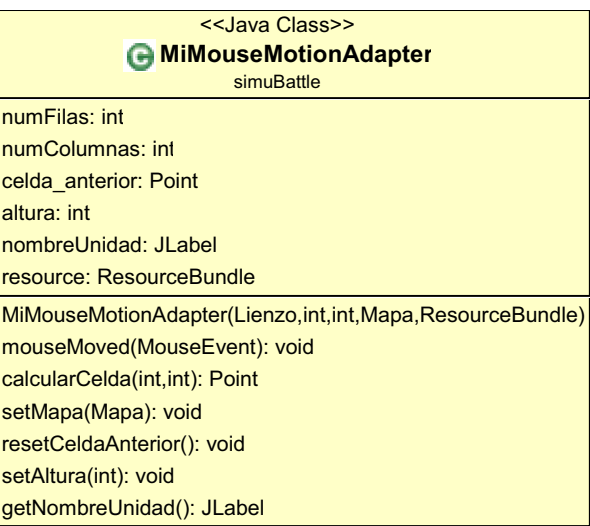




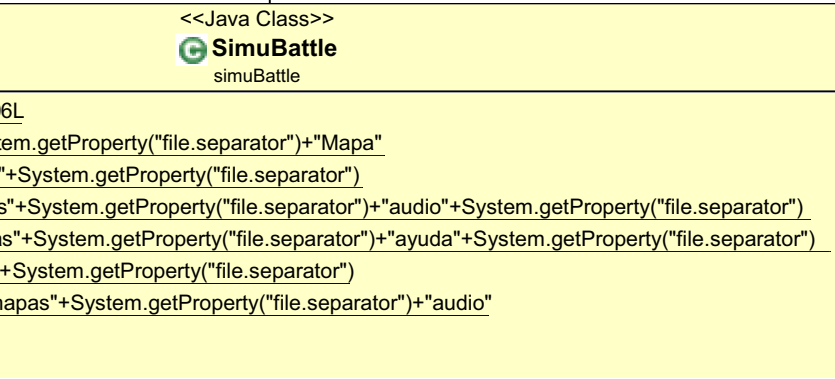
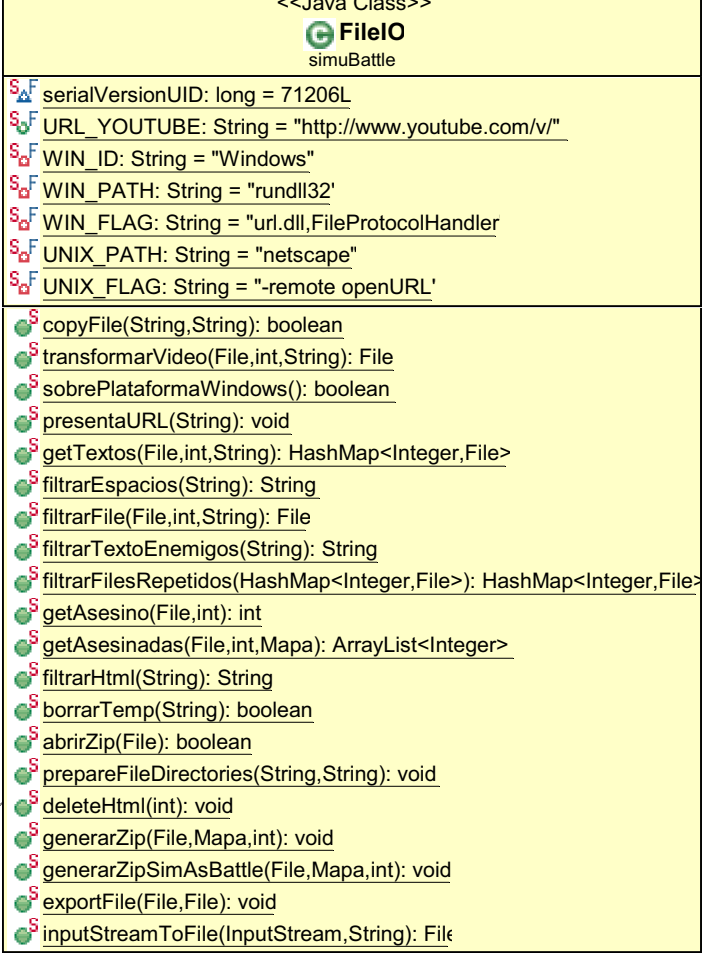
-areaTexto 0..1

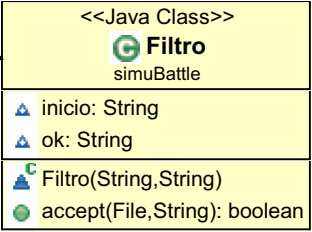




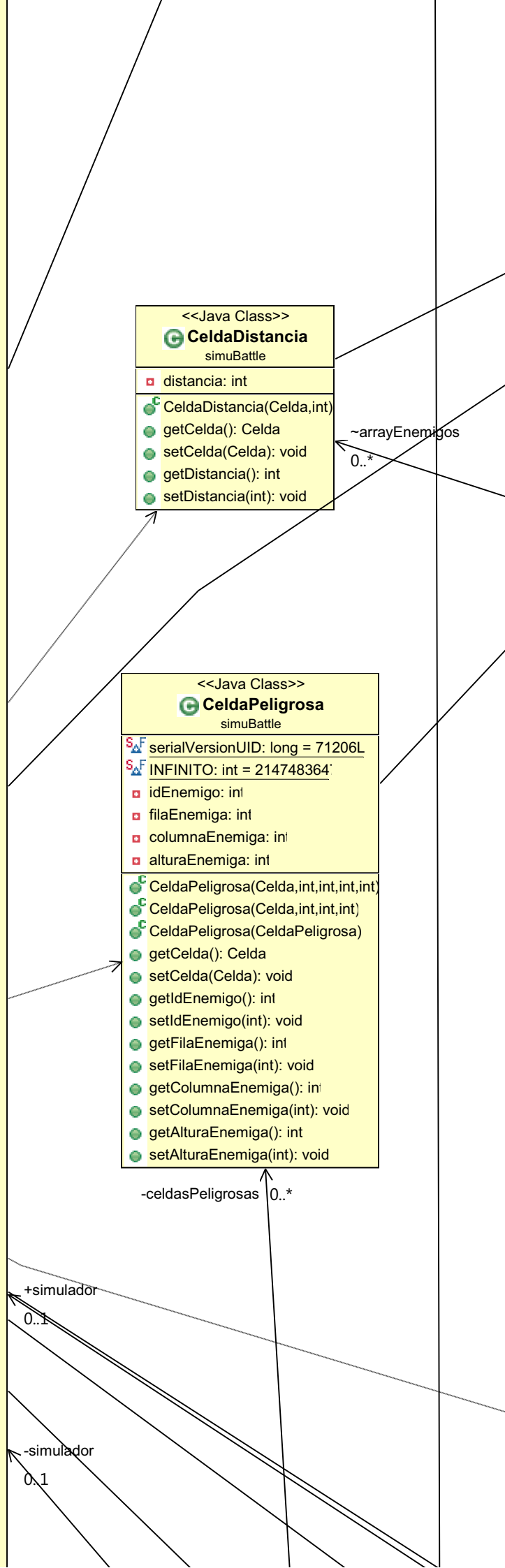


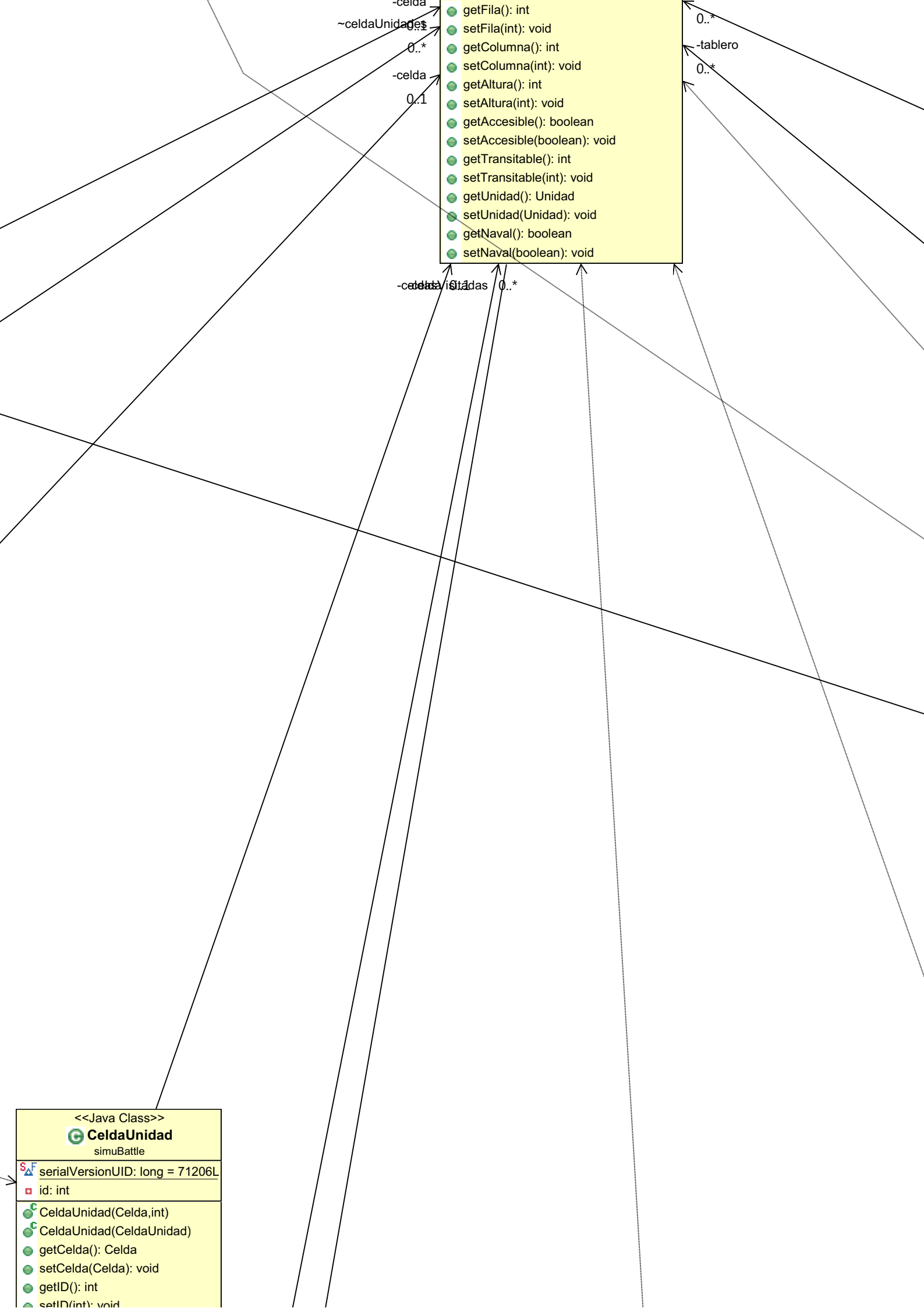
-motion 0..1





● borrarPosterioros(int): void
● volverMapa(int): void
● volverMapa(Mapa,HashMap<String,Object>): void
● getNumMapaSimulacion(): int
● setNumMapaAnterior(int): void
● getNumUnidadesSinDisparo(): int
● getStopFinal(): boolean
● setStopFinal(boolean): void
● getStoppedGlobal(): boolean
● setStoppedGlobal(boolean): void
● getNumStopped(): int
● setNumStopped(int): void
● getNumBandos(): int
● getNumFilas(): int
● getNumColumnas(): int
● getNumAlturas(): int
● getMapa(): Mapa
● setMapa(Mapa): void
● getArrayMapas(): ArrayList<Mapa>
● getArrayEstadisticas(): ArrayList<HashMap<String,Object>>
● getArrayCapitaneadas(): ArrayList<FondoAltura>
● inicializarNumMapaSimulacion(): void
● incrementarNumMapaSimulacion(): void
● addMapEstadisticas(): void
● getEstadisticasActual(): HashMap<String,Object>
● getCeldaUnidades(): ArrayList<Celda>
● setCeldaUnidades(ArrayList<Celda>): void
● getIndexGlobal(): int
● setIndexGlobal(int): void
● getNumSinDisparos(): int
● setNumSinDisparos(int): void
● getMaxSinDisparos(): int
● setMaxSinDisparos(int): void
● imprimirTablero(Mapa): void
● resetFileTemp(Unidad,boolean): void
● saveFileTemp(Unidad): void
● writeFileTemp(String): void
● writeFileTempFirstEstadistica(String): void
● writeFileTempLastEstadistica(String): void
● clearFileMapsYTemp(int): void
● clearFileMaps(int): void
● clearFileTemp(): void
● algoritmo(Unidad,int,int,int): HashMap<String,Object>
● algoritmoHumano(int,Unidad,int,int,int,int,int,int): HashMap<String,Object>
● resetearNumSinDisparos(): void
● aumentarNumSinDisparos(): void
● getArrayCostes(Unidad,int,int,int): float[]
● calcularPuntos(Mapa): void
● refreshMapa(Mapa,Bando[]): void
● actualizarEstadisticas(Mapa,Unidad,int,int,int): void
● actualizarEstadisticasBandos(Mapa,Bando[]): void
● unidadViva(Mapa,int): boolean
● getAyudada(Mapa,int): Unidad
● getCeldaUnidades(Mapa): ArrayList<Celda>
● getCeldaUnidadesRnd(): ArrayList<Celda>
● masDeUnBando(Celda[],int,int,int): boolean
● reiniciarTablero(Celda[],int,int,int): void
● getArrayCostes(Mapa,int,int,int,Unidad): float[]
● decisor(Mapa,int,int,int,Unidad): HashMap<String,Object>
● getObjectivoVertical(Mapa,int,int,int): Celda
● seleccionarAtaque(Mapa,Celda,Celda,int): HashMap<String,Object>
● alcanceEnemigo(Mapa,int,int,int,Unidad): ArrayList
● getCeldasEnemigas(ArrayList<Integer>,Mapa): ArrayList<Celda>
● dispararMisiilPorHumano(Celda,Mapa,Unidad,int,int,int): HashMap<String,Object>
● calcularHeridosColaterales(Mapa,Unidad,int,int,int): int
● dispararMisiil(Mapa,ArrayList<Integer>,Unidad,int,int,int): HashMap<String,Object>
● coordenadaOK(Mapa,int,int,int): boolean
● ayudar(Unidad,Unidad): void
● calcularDistancia(int,int,int,int): int
● getCompaneroCercano(Mapa,int,int,int,int,int,boolean): Celda



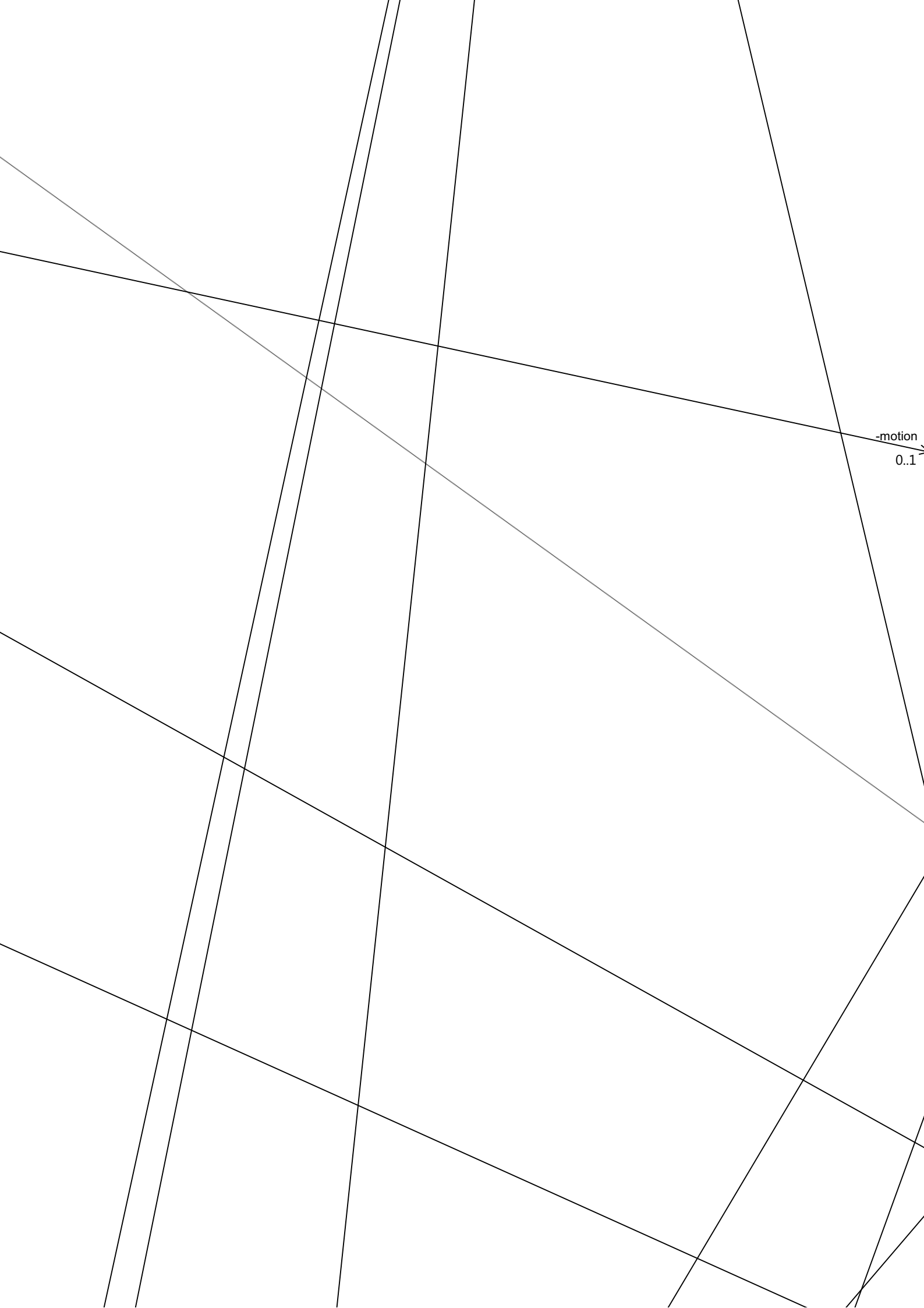


- mouseExited(MouseEvent): void
- calcularCelda(int,int): Point
- getMapa(): Mapa
- setMapa(Mapa): void
- getIconoLienzo(): IconoLienzo
- setIconoLienzo(IconoLienzo): void
- getBando(): Bando
- setBando(int): void
- setAltura(int): void
- setPressedIndividual(boolean): void
- setEsCelda(boolean): void
- setEsTerrestre(boolean): void
- getEsTerrestre(): boolean



-adapterParam

0.1



-motion

0..1



<<Java Class>>



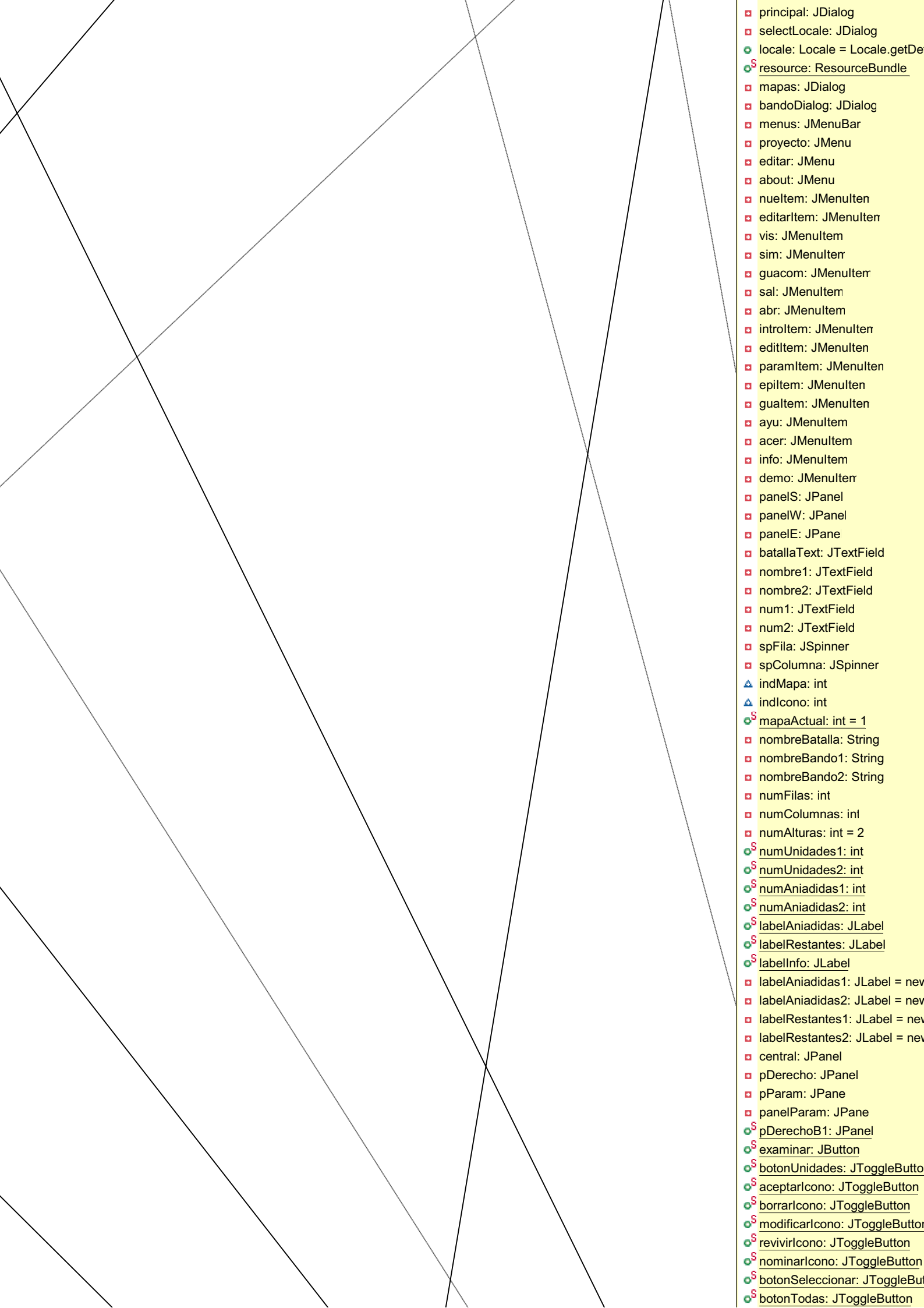
MiMouseMotionAdapterParam

simuBattle

■ numFilas: int
■ numColumnas: int
■ celda_anterior: Point
■ altura: int
■ nombreUnidad: JLabel
■ resource: ResourceBundle

● MiMouseMotionAdapterParam(LienzoParam,int,int,Mapa,ResourceBundle)
● mouseMoved(MouseEvent): void
● calcularCelda(int,int): Point
● setMapa(Mapa): void
● resetCeldaAnterior(): void
● setAltura(int): void
● getNombreUnidad(): JLabel

-motionPararr 0.1



fault()

v JLabel()
v JLabel()
v JLabel()
v JLabel()

n[]

n

ton

<<Java Class>>

LimitDocument

simuBattle

S

F

serialVersionUID: long = 71206L

limit: int

C

LimitDocument(int)

C

LimitDocument(int,String)

F

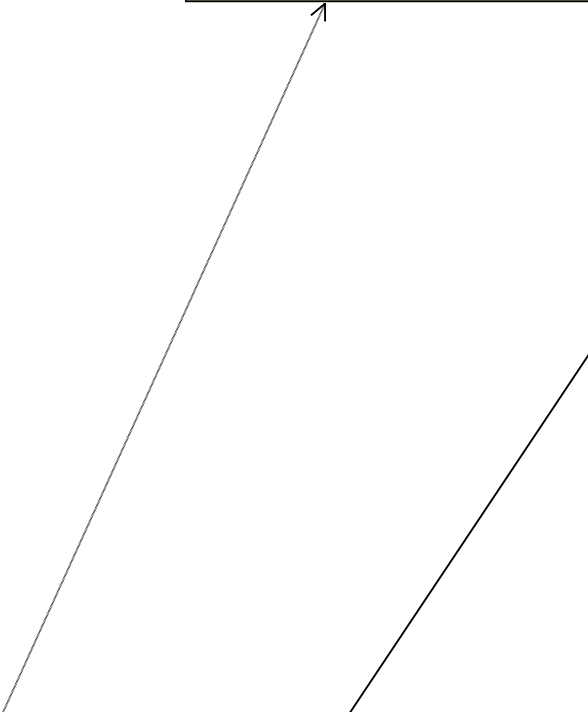
getLimit(): int

C

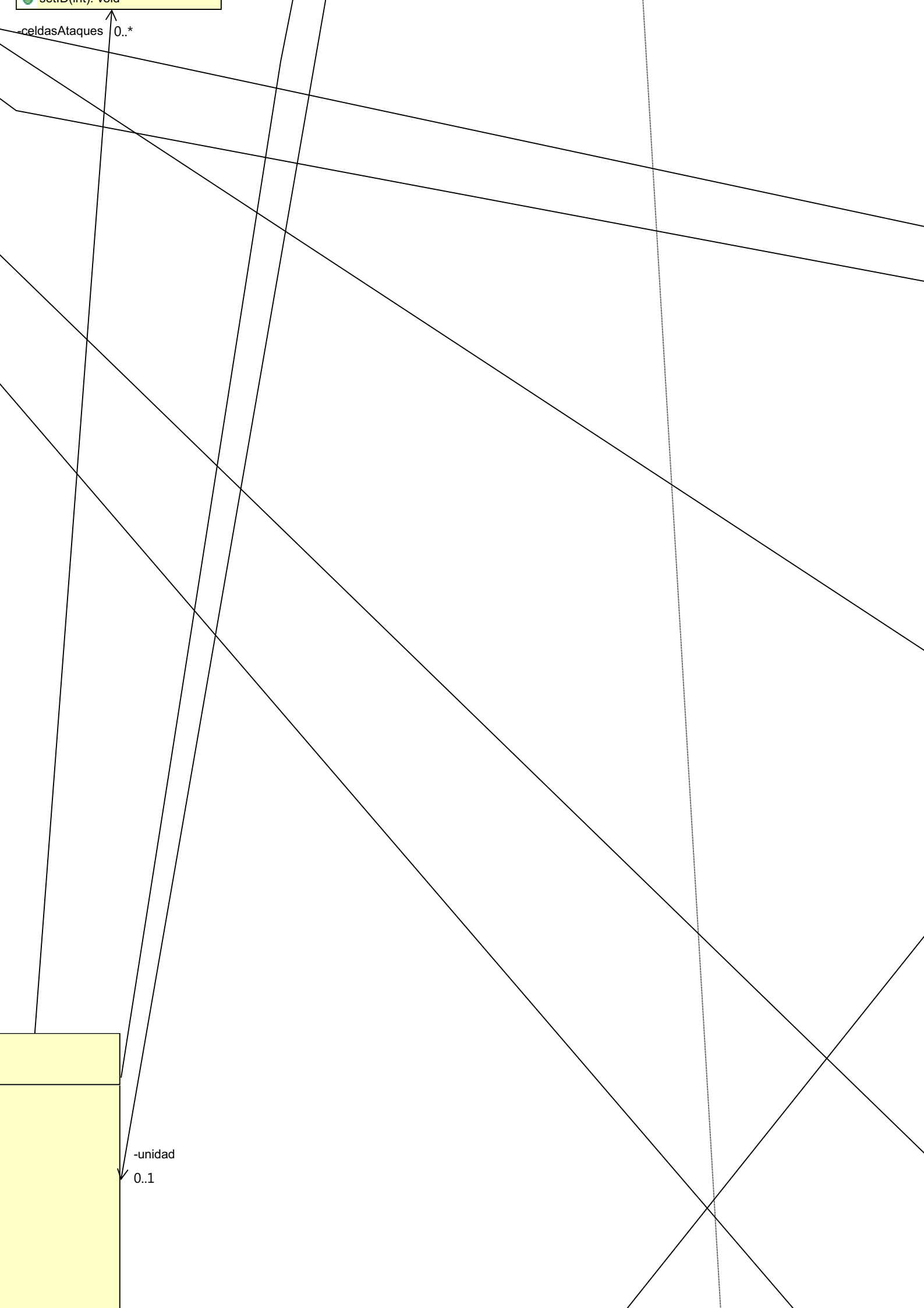
insertString(int,String,AttributeSet): void

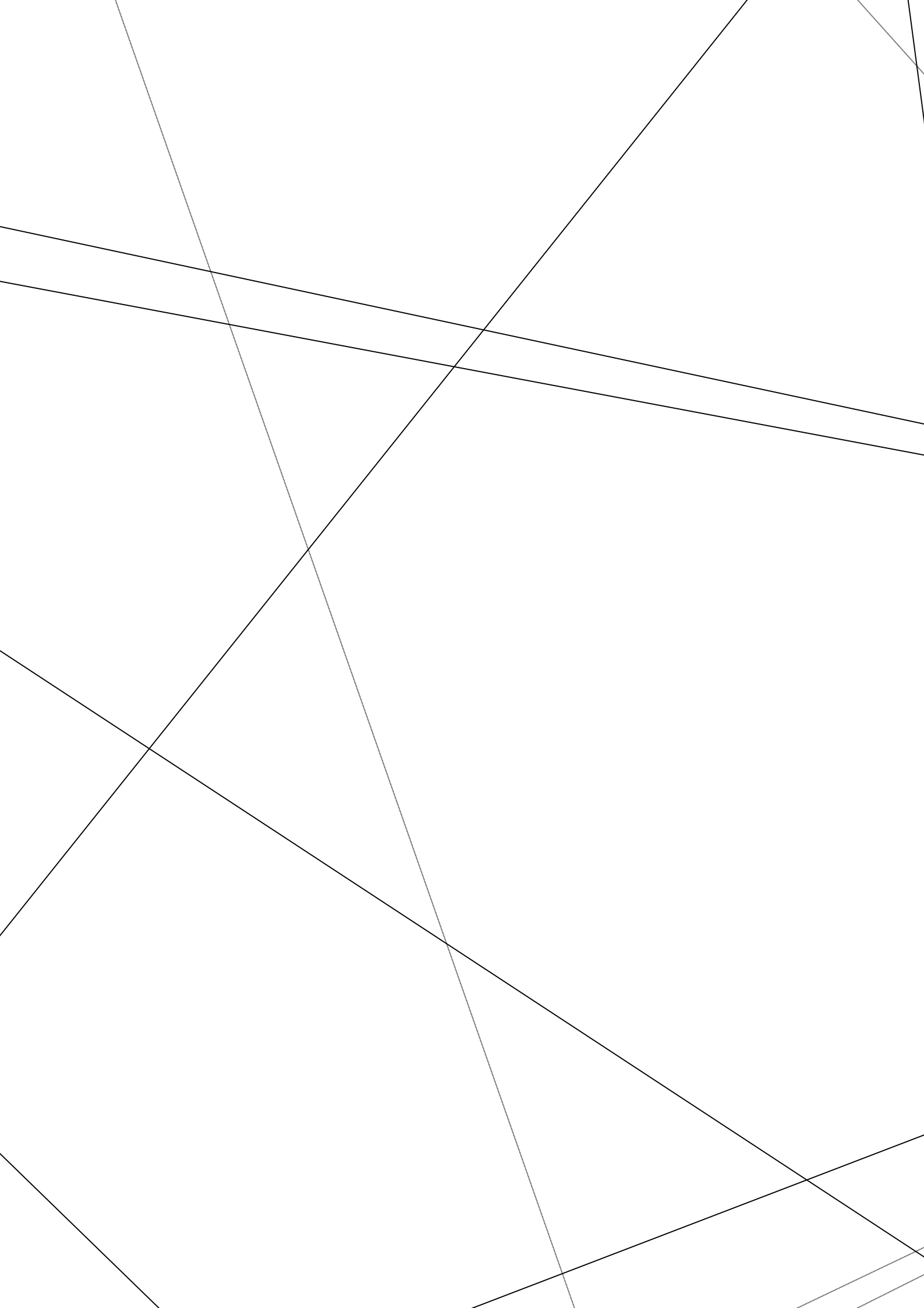
F

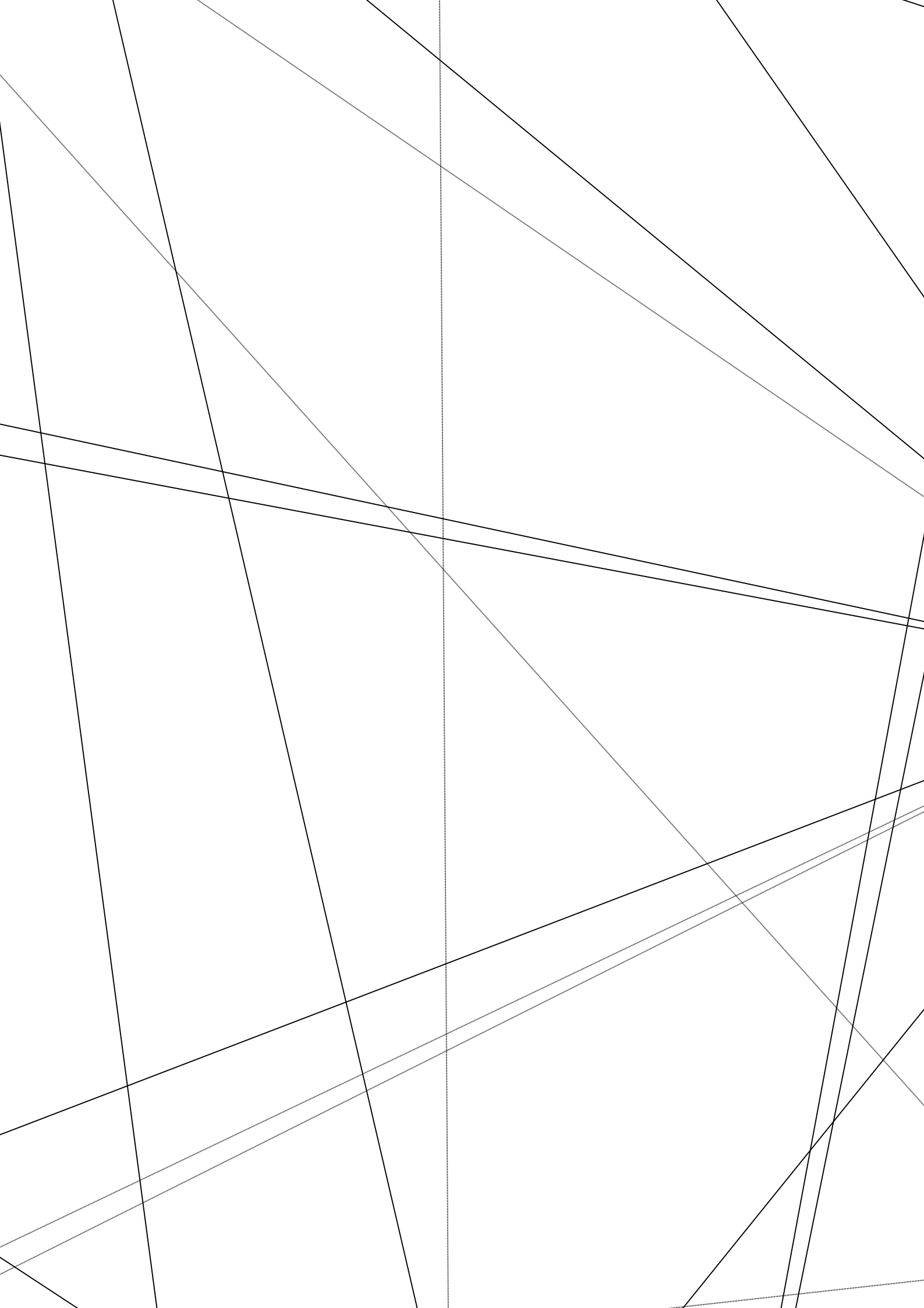
setLimit(int): void

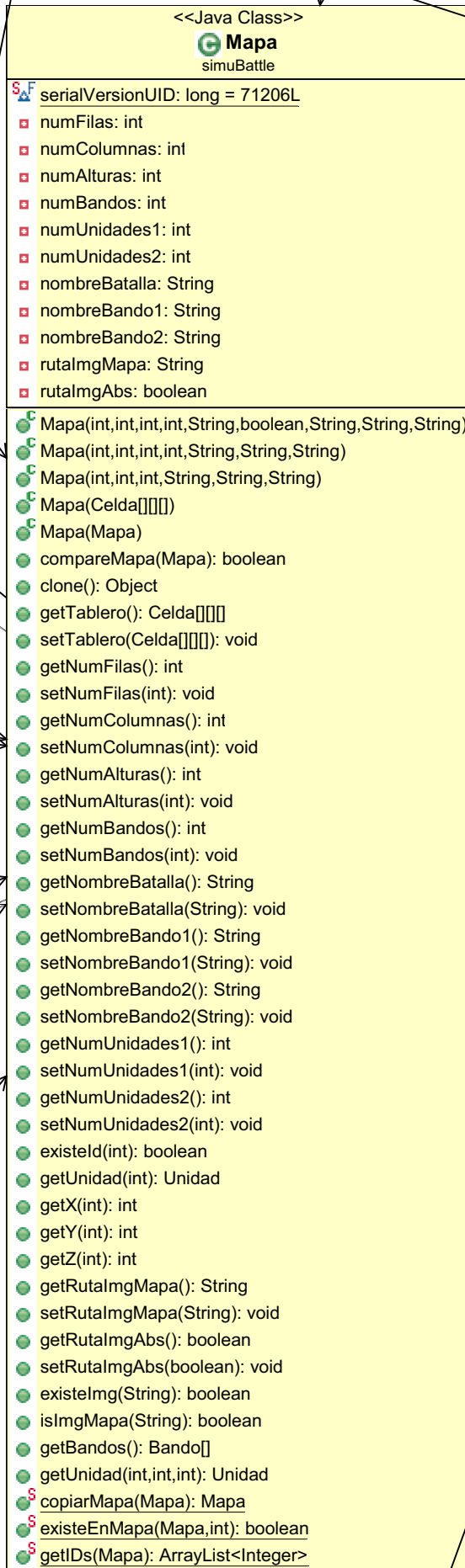












-mapa 0..1

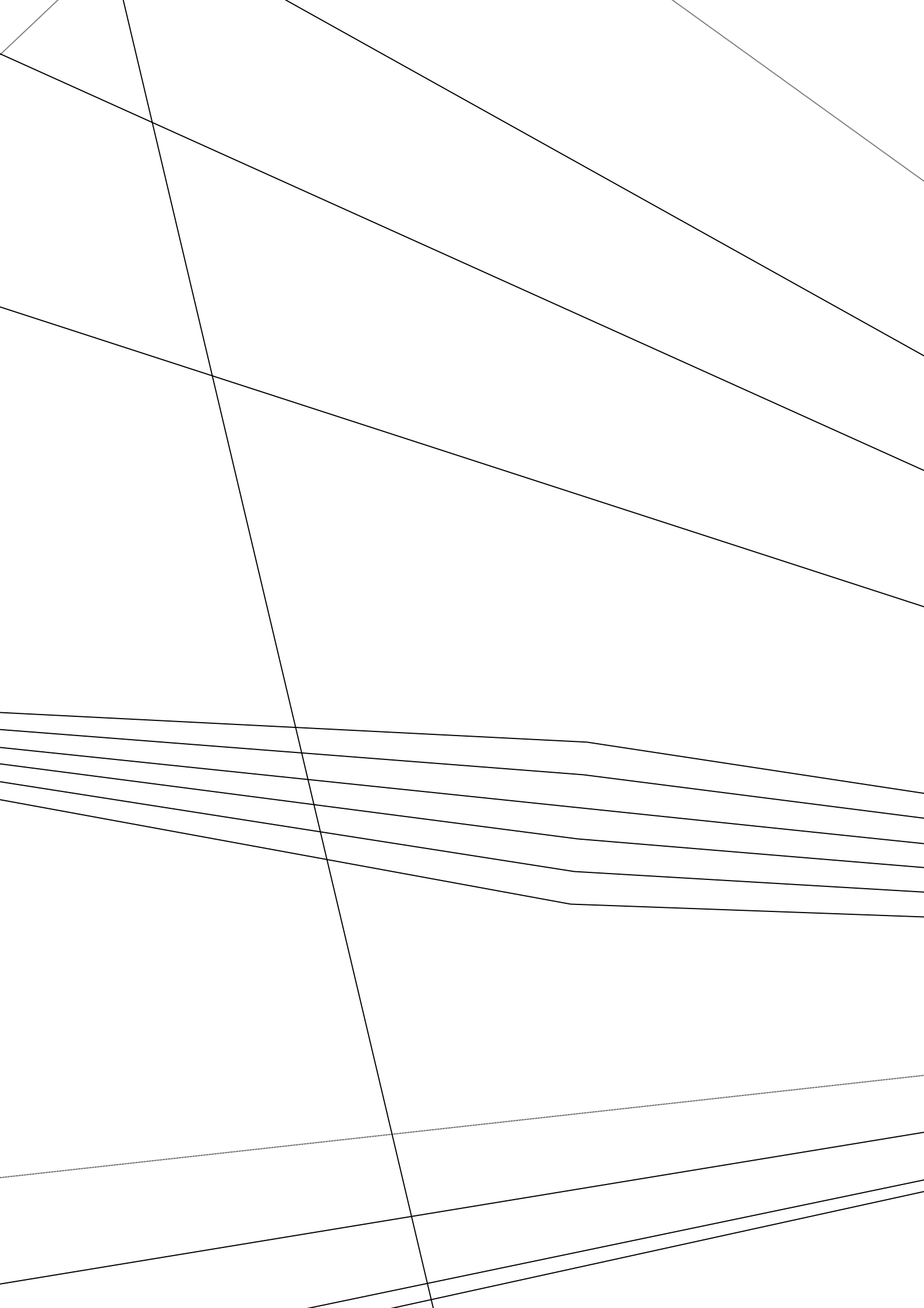
-mapa
0..1

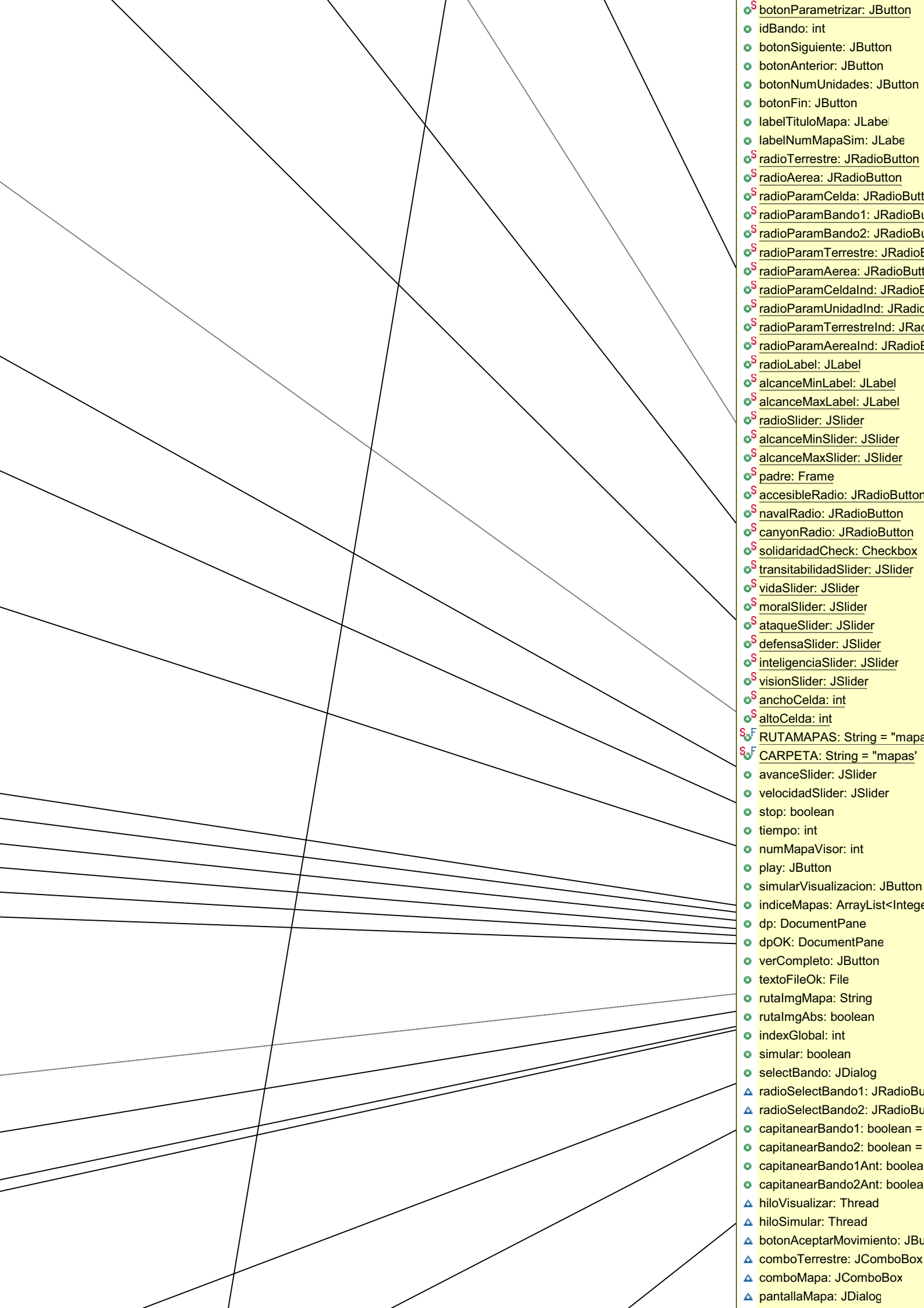
-mapa
0..1

+arrayMapas
+Mapa SimulacionActual
+Mapa SimulacionAux
0..1

~mapa
0..1

~mapa 0..1





or
utton
utton
Button
on
Button
Button
dioButton
Button

is"+System.getProperty("file.separator")+"Mapa"

er>

tton
tton
false
false
n
n
tton

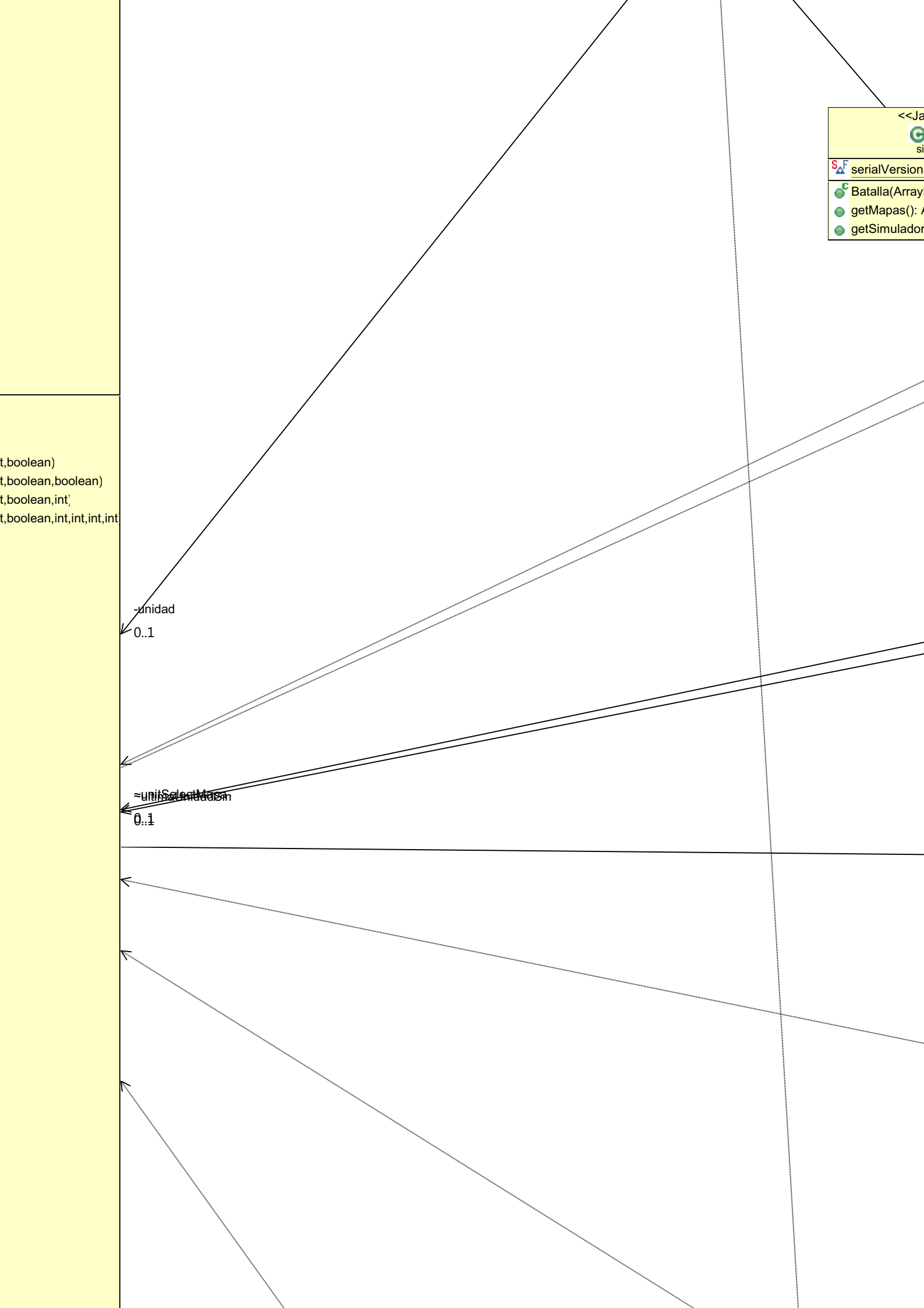


~unidadAnteriorTemp

0..1

alcanceMax: int = 2
radio: int = 1
vision: int
inteligencia: int
turnoFinalizado: boolean
ayudando: boolean
pedirAyuda: boolean
solidario: boolean
filaDestino: int
columnaDestino: int
alturaDestino: int
idDestino: int
idAyudado: int
idAyudante: int
numAtacado: int
numAtacadoOK: int
numAtaque: int
numAtaqueOK: int
numBajas: int
numTurnos: int
naval: boolean

Unidad(int,Bando)
Unidad(int,String)
Unidad(int,Bando,int)
Unidad(int,Bando,int,int,int,int,int,int,int)
Unidad(int,Bando,int,int,int,int,int,int,int)
Unidad(int,Bando,int,int,int,int,int,int,int)
Unidad(int,Bando,int,int,int,int,int,int,int)
Unidad(Unidad)
compareUnidad(Unidad): boolean
getID(): int
setID(int): void
getNombre(): String
setNombre(String): void
getBando(): Bando
setBando(Bando): void
getIDBando(): int
setIDBando(int): void
getFilaDestino(): int
setFilaDestino(int): void
getColumnaDestino(): int
setColumnaDestino(int): void
getAlturaDestino(): int
setAlturaDestino(int): void
getIdDestino(): int
setIdDestino(int): void
getIdAyudado(): int
setIdAyudado(int): void
getIdAyudante(): int
setIdAyudante(int): void
getVida(): int
setVida(int): void
getMoral(): int
setMoral(int): boolean
getAtaque(): int
setAtaque(int): boolean
getDefensa(): int
setDefensa(int): boolean
getTipo(): int
setTipo(int): void
getTurnoFinalizado(): boolean
setTurnoFinalizado(boolean): void
getAyudando(): boolean
setAyudando(boolean): void
getPedirAyuda(): boolean
setPedirAyuda(boolean): void
getVision(): int
setVision(int): void
getInteligencia(): int
setInteligencia(int): void



<<Java Class>>

Batalla

simuBattle

UID: long = 71206L

List<Mapa>,Simulador)


ArrayList<Mapa>

(): Simulador

~arrayCapitaneadas

0..*

<<Java Class>>

 **FondoAltura**

simuBattle

serialVersionUID: long = 71206L

INFINITO: int = 214748364

id: int

layer: int

fondo: JLabel

FondoAltura(int,int,JLabel)

FondoAltura(FondoAltura)

getID(): int

setID(int): void

getLayer(): int

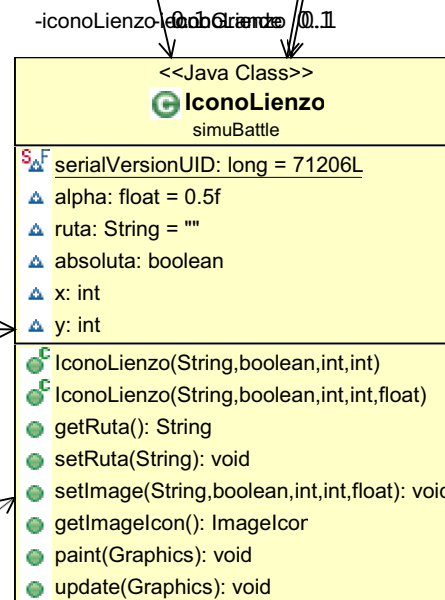
setLayer(int): void

getFondo(): JLabel

setFondo(JLabel): void

~arrayCapitaneadasTemporal

0..*









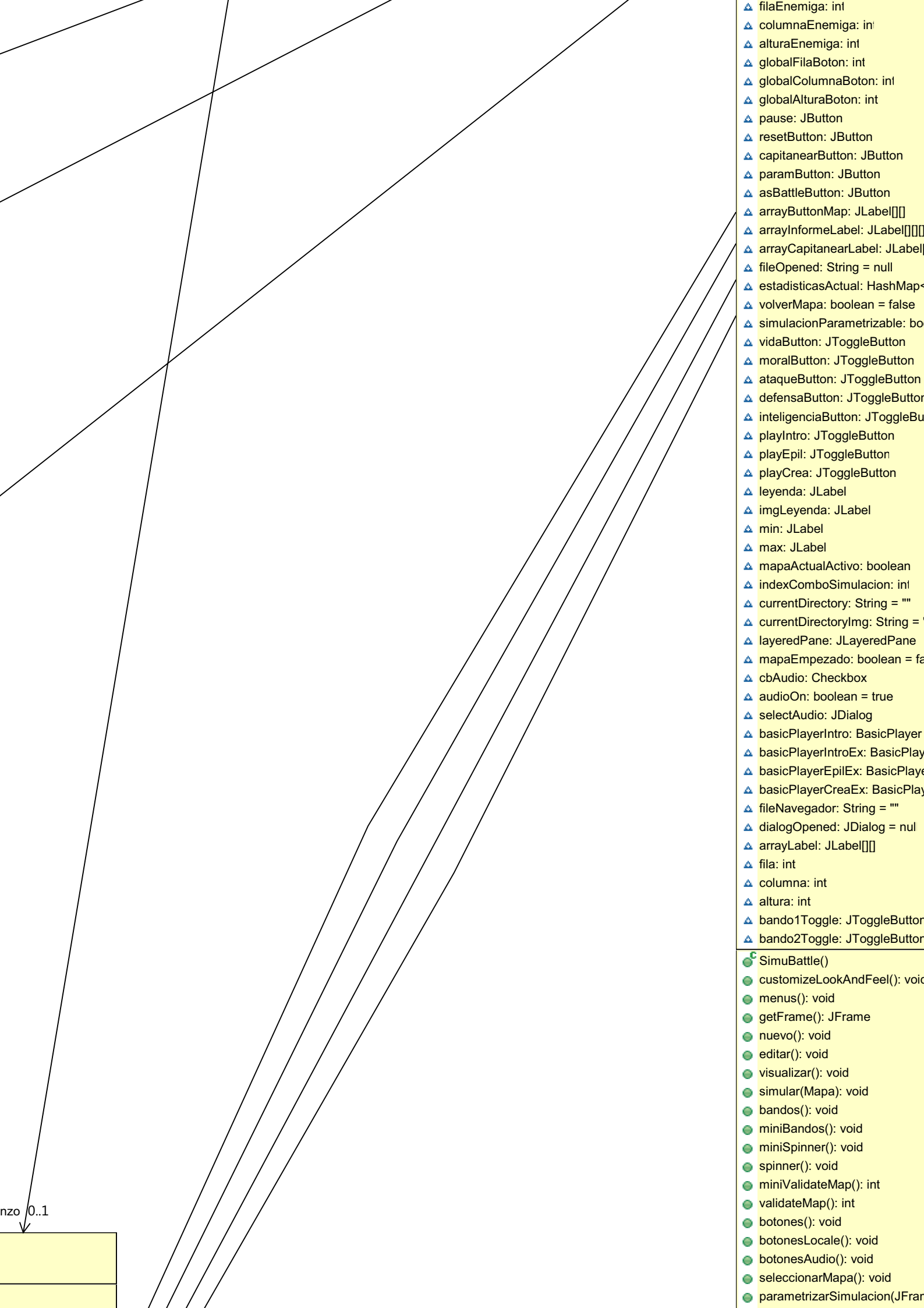
-lienzo 0..1

-lie

<<Java Class>>

 **Lienzo**
simuBattle

 serialVersionUID: long = 71206L



```
<String,Object>
```

```
boolean = false
```

utton

00

else

```

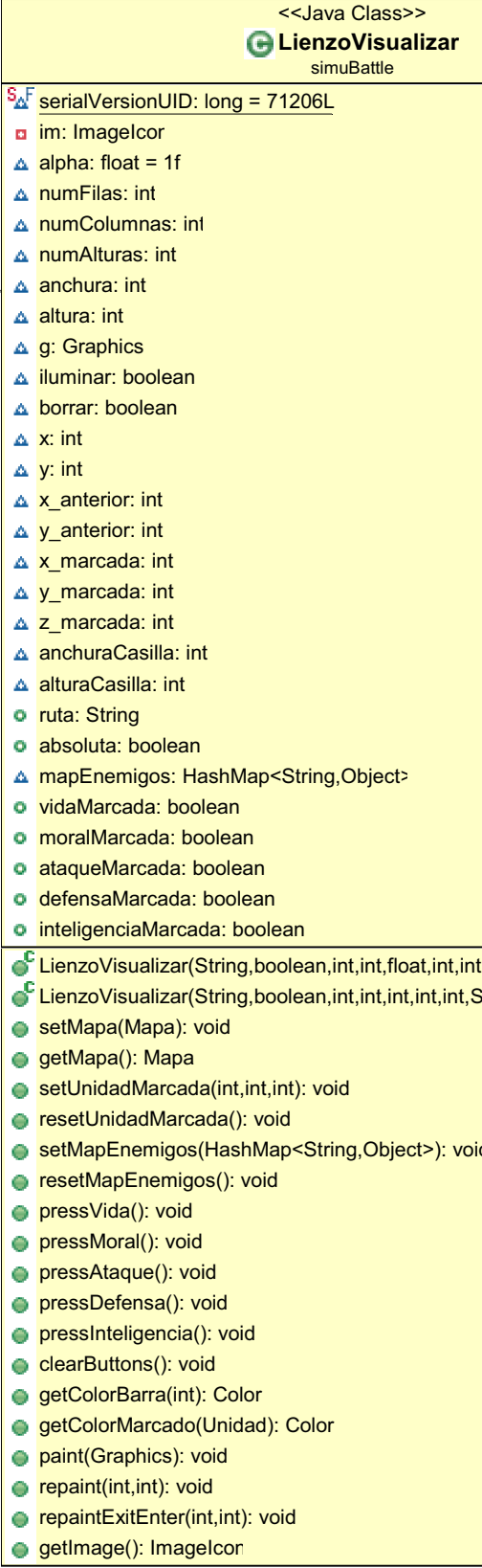
= null
er
er
ver = null

```

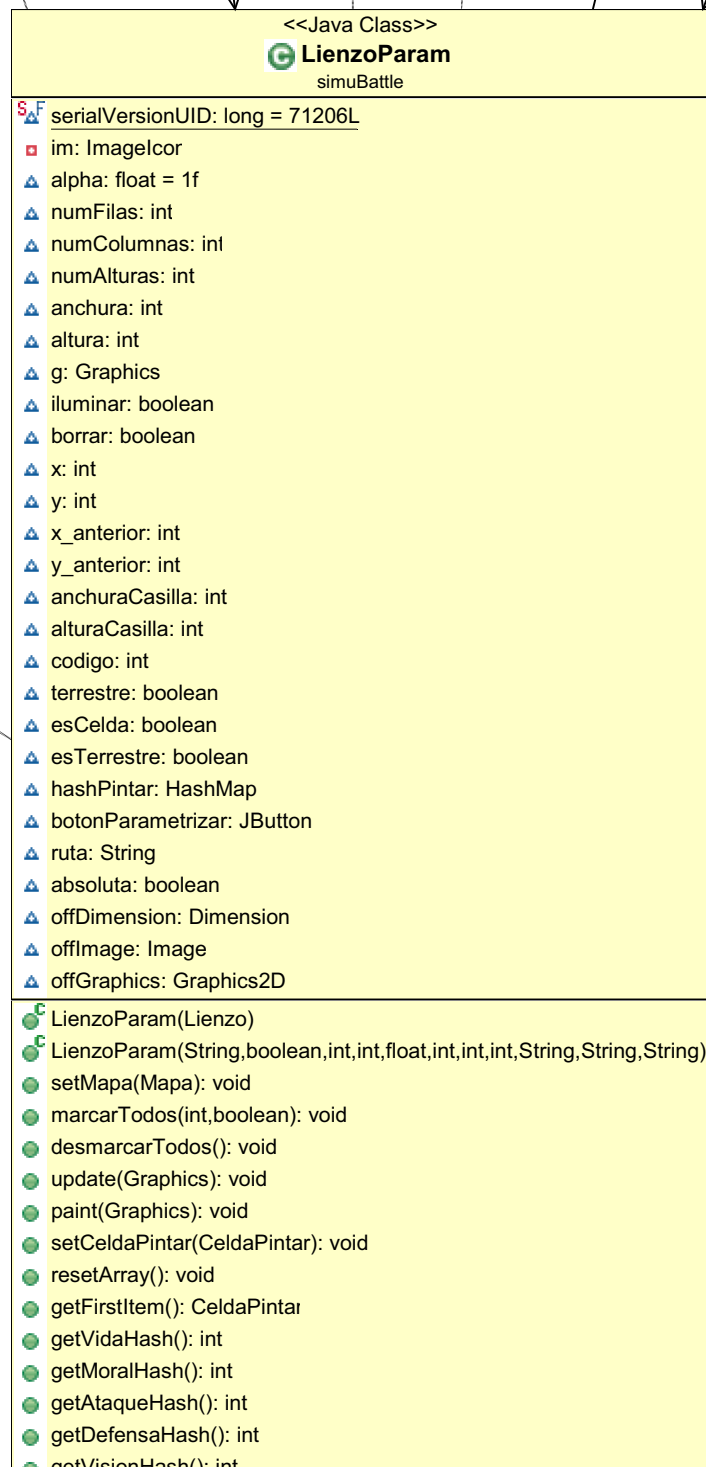
```
ne,int,boolean): void
```


- getSolidario(): boolean
- setSolidario(boolean): void
- getNumAtacado(): int
- setNumAtacado(int): void
- incrementarNumAtacado(): void
- getNumAtacadoOK(): int
- setNumAtacadoOK(int): void
- incrementarNumAtacadoOK(): void
- getNumAtaque(): int
- setNumAtaque(int): void
- incrementarNumAtaque(): void
- getNumAtaqueOK(): int
- setNumAtaqueOK(int): void
- getAlcanceMin(): int
- setAlcanceMin(int): void
- getAlcanceMax(): int
- setAlcanceMax(int): void
- setAlcances(int,int): void
- getRadio(): int
- setRadio(int): void
- incrementarNumAtaqueOK(): void
- getNumBajas(): int
- setNumBajas(int): void
- incrementarNumBajas(): void
- getNumTurnos(): int
- setNumTurnos(int): void
- incrementarNumTurnos(): boolean
- getNaval(): boolean
- setNaval(boolean): void
- getCeldasPeligrosas(): ArrayList<Celda>
- getCeldasAtaques(): ArrayList<Celda>
- getCeldasVisitadas(): ArrayList<Celda>
- getIconoLienzo(): IconoLienzo
- setIconoLienzo(IconoLienzo): void
- getCelda(Mapa): Celda

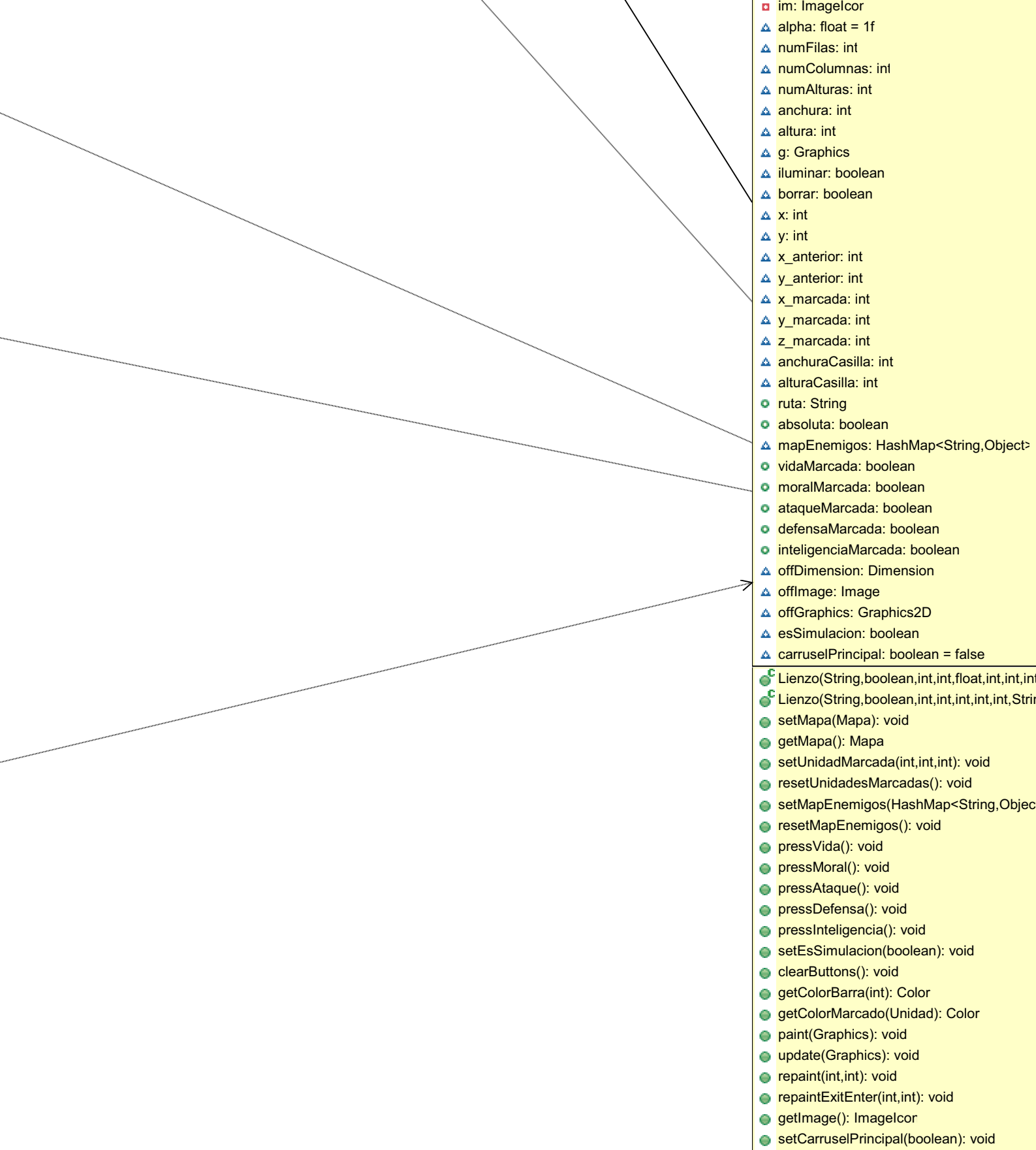
celdaPeligrosa>
 daUnidad>
 lda>

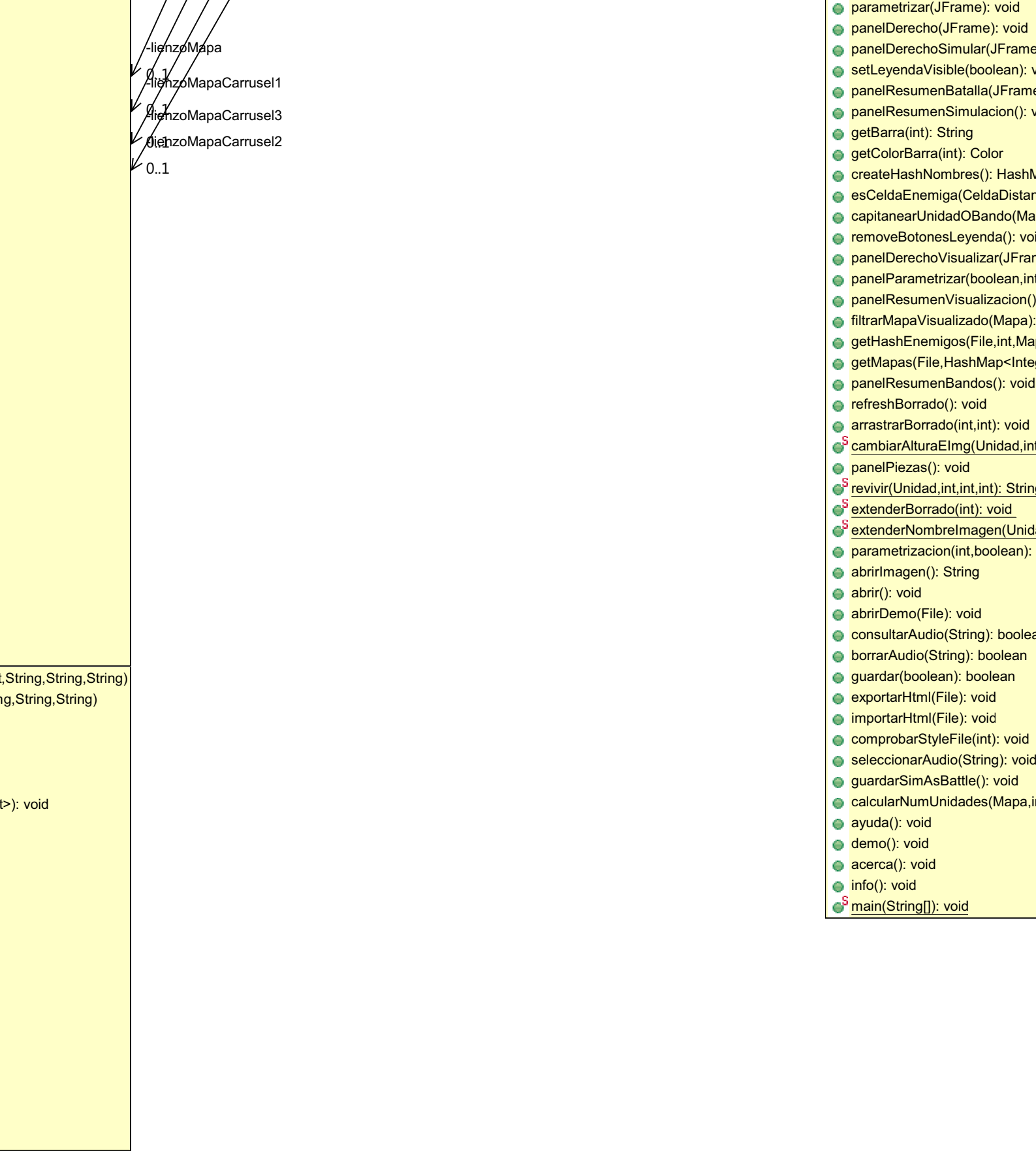












,Mapa): void
void
): void
void

Map<Integer,String>
cia[],int,int,int): boolear
pa): void
d
ne): void
,boolean): void
: void
Mapa
pa): HashMap<String,Object>
ger,File>,int): ArrayList<ArrayList>

): String

g
ad): void
void

in

nt): int

- getVisionHash(): int
- getInteligenciaHash(): int
- getSolidaridadHash(): boolean
- getTipoHash(): int
- getAlcanceMinHash(): int
- getAlcanceMaxHash(): int
- getRadioHash(): int
- getTransitabilidadHash(): int
- getAccesibilidadHash(): boolean
- getNavalHash(): boolean
- repaint(int,int): void
- setBotonParametrizar(JButton): void
- setEsCelda(boolean): void
- setEsTerrestre(boolean): void

Pliego III

Manual de usuario

Contenido

1. Sobre SimuBattle	173
2. Empezando	174
3. Edición de batallas.....	176
3.1. Inserción, modificación y borrado de unidades.....	180
3.2. Gestión de mapas.....	182
3.3. Audio de la batalla	183
3.4. Comentarios de la batalla.....	183
4. Visualización de la batalla.....	190
5. Parametrización.....	192
6. Simulación.....	195

1. Sobre SimuBattle

SimuBattle es una herramienta que permite a profesores y alumnos de historia, disponer de una aplicación con la que poder estudiar con mayor profundidad, pero también desde un punto de vista más lúdico, cualquier batalla histórica y las causas y consecuencias de la misma. Esto se logra mediante la creación y simulación de las batallas. La creación no sólo permite generar los mapas explicativos, sino además comentarios que los acompañan y los dotan de un mayor didactismo. Por su parte, la simulación permite a los alumnos interactuar directamente e incluso participar en la batalla, variando no sólo los parámetros, sino decidiendo los movimientos de una o varias unidades.

SimuBattle permite construir batallas de forma dinámica y cómoda para el usuario, de una manera totalmente gráfica. Por su parte, los comentarios son insertados mediante un editor de texto sencillo. Esto genera una estructura altamente familiar para el usuario, interactiva, lúdica y didáctica.

A semejanza de cualquier otro editor, SimuBattle puede guardar en disco la batalla con la que está operando, así como abrir aquellas previamente guardadas. Una vez haya creado su batalla empleando SimuBattle, usted podrá trabajar libremente con ella, introduciendo comentarios en forma de texto, imágenes, vídeo e incluso audio. La libertad a la hora de generar la batalla es prácticamente total, se puede escoger orografía del terreno, distintos tipos de unidades, número de mapas, movimiento de las unidades... De igual forma es posible simular tantas veces como se quiera una batalla, variando las características de las unidades y del terreno, comprobando así la importancia de ciertos factores en la resolución de la misma. Es posible trabajar con las simulaciones como si se tratara de una batalla generada por el usuario, permitiendo completarla con nuevos comentarios, e incluso modificarla totalmente.

SimuBattle es una aplicación especialmente diseñada para profesores de Historia de los primeros niveles de enseñanza y para sus alumnos, a quienes no se les presupone un gran conocimiento tecnológico. Es por ello que pretende ser una herramienta interactiva y sobre todo divertida que le permita a usted, usuario, comprobar “in situ” cómo se desarrolló cualquier batalla histórica, y participar activamente en ella.

Para ayudarle en esta tarea, a lo largo de este manual encontrará



algunas notas marcadas con el siguiente símbolo. Estas notas le explicarán los conceptos y nociones básicas que no están directamente relacionadas con SimuBattle, sino con las estructuras o técnicas usadas en la creación de batallas y que, probablemente, le ayuden a fijar sus ideas. Si usted está familiarizado con el empleo de mapas históricos de batallas, puede saltar estas notas sin miedo alguno.

Además de éstas, a lo largo del texto del manual encontrará otras notas



marcadas con el símbolo que le aclararán aspectos de tipo práctico (auxiliares) que puede que ya conozca, y que por lo tanto, sólo necesitará consultar si tiene una duda concreta.

2. Empezando

A estas alturas ya ha leído un poco sobre esta herramienta, lo que hace y sobre todo cómo y para qué debe usarse. Es, por lo tanto, un buen momento para echarle un vistazo general. Para empezar con buen pie deberá, por supuesto, iniciar la aplicación.



SimuBattle se proporciona en dos formatos diferentes, cada uno de los cuales tiene su propia forma de inicialización:

- *Ejecutable (.exe). Sólo para sistemas Windows. Para iniciar la aplicación basta con hacer doble clic sobre el archivo .exe.*
- *Archivo jar. Válido tanto en sistemas Windows como Linux. La aplicación puede iniciarse de dos formas:*
 - a) *En línea de comandos mediante la instrucción “java -jar SimuBattle.jar”.*
 - b) *Haciendo doble clic sobre el archivo .jar si dispone de una versión de java mayor o igual a 1.6 (usted puede comprobar este punto tecleando “java -version” en su línea de comandos).*

Al iniciarse la aplicación, y tras seleccionar el idioma de la misma, surgirá una ventana vacía, con un barra de menús donde podrá escogerse tan sólo entre dos opciones: Nuevo y Abrir. Se ha intentado que el menú sea lo más sencillo posible, además de que el grueso de las opciones se le ofrecen al usuario en la pantalla principal de la aplicación. En la Figura 1 se muestra un ejemplo del formato de vista de SimuBattle que servirá para comprender mejor la manera gráfica en que se estructura.

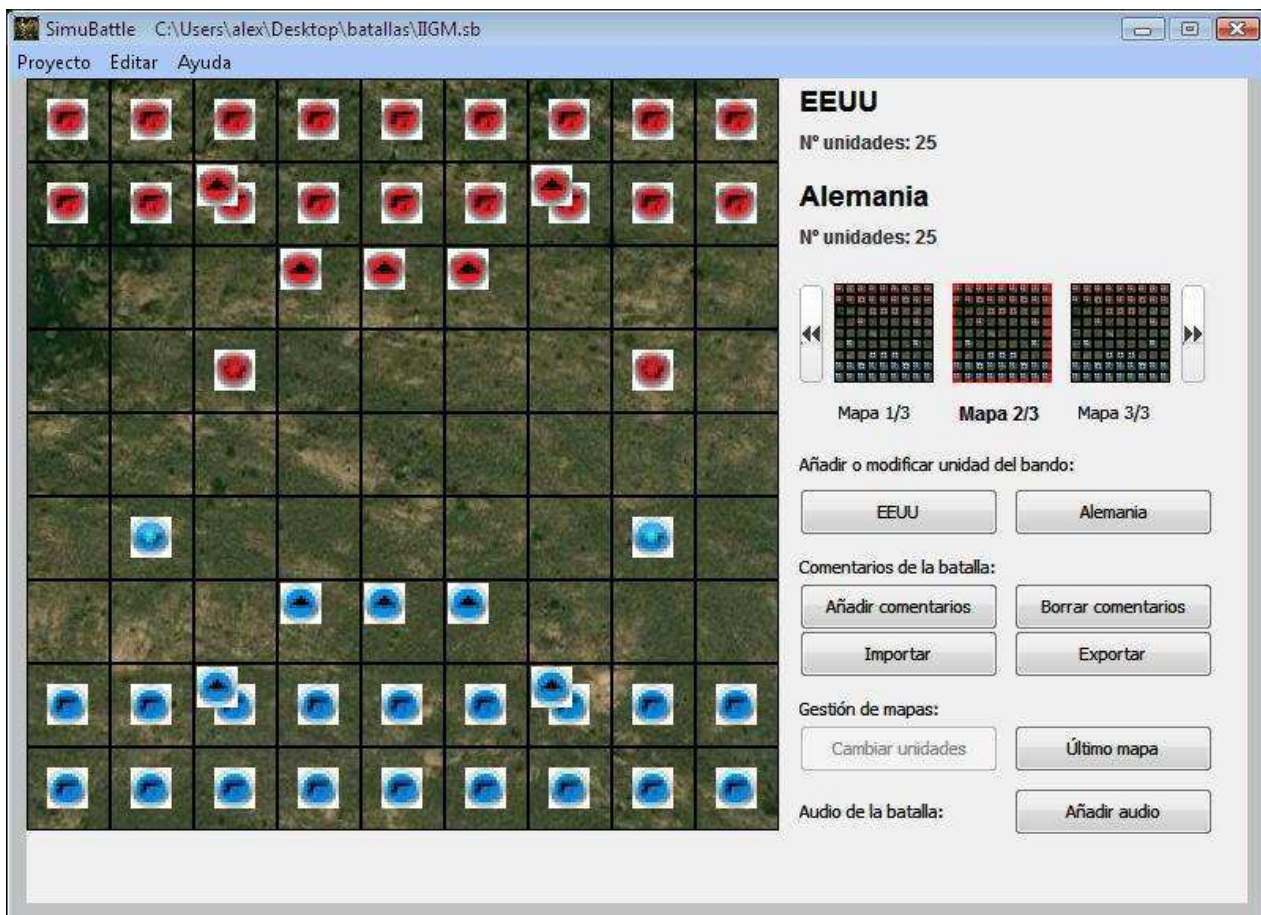


Figura 1. Ejemplo de pantalla de SimuBattle

La estructura de las distintas pantallas sigue un formato similar al aquí mostrado. Al igual que la gran mayoría de las aplicaciones actuales, dispone de un menú que contiene las acciones principales que se pueden realizar con la aplicación. Como ya se ha explicado, se ha intentado que éste fuera lo más sencillo posible, mostrando en cada pantalla las distintas opciones en el propio área de trabajo, disponiéndolo así a modo página Web.

En la parte izquierda se despliega el mapa de la batalla con el que se está trabajando. Siguiendo una disposición de cuadrícula, a modo de tablero de ajedrez, sobre él se disponen las distintas unidades y efectúan sus movimientos.

Ya en la parte de la derecha, se despliegan ante el usuario, en forma de botones, todas las opciones de que dispone en cada uno de los grandes bloques en que se divide la aplicación: Creación, Visualización, Parametrización y Simulación.



Aviso: Los diversos botones y opciones del menú desplegable permanecen deshabilitados en caso de que no sean aplicables en ese momento en que se encuentra la batalla. Así, se evita que el usuario pueda generar situaciones incongruentes o que pueda llevar a equívocos.

Esta es por tanto la estructura visual que nos acompañará durante la utilización de SimuBattle. Una estructura claramente definida y continuista a lo largo de las distintas secciones de que se compone esta aplicación. Recordemos por tanto:

- Un menú con las opciones básicas y del que se puede acceder de una sección a otra: Simulación, Visualización, Parametrización y Simulación.
- La batalla visual en forma de mapa y sus correspondientes unidades, situado siempre en el lado izquierdo y ocupando la mayor parte de la pantalla, dándole así todo el protagonismo.
- Las distintas opciones en la parte derecha de la pantalla en forma de botones.

Hasta este momento se ha realizado un breve recorrido a vista de pájaro por la aplicación. Este recorrido le habrá resultado de ayuda para entrar en contacto con la herramienta; aún así, sería buena idea que antes de comenzar la lectura de los siguientes apartados, se atreviese un poco a trabajar con SimuBattle. Las diversas opciones que se presentan son muy intuitivas, y siempre se aprende más “lanzándose a la piscina” y atreviéndose con el uso de una aplicación.

Aún así, los puntos restantes en este apartado contienen detalles de la operación con SimuBattle, y puesto que estará deseando entrar en dinámica, pasemos a ellos sin más dilación.

3. Edición de batallas

Por supuesto, la creación de una batalla es la primera y más fundamental de las operaciones a realizar con SimuBattle, ya que el objetivo de esta aplicación no es otro que el que usted pueda realizar sus propias creaciones históricas.

Comenzaremos por la creación de una batalla (o proyecto) nueva. Para ello, diríjase al menú Proyecto -> Nuevo. En la pantalla aparecerá un resumen de la batalla con la que se comienza a trabajar. En este caso, al ser una batalla nueva, ésta carece de nombre, bandos, así como mapa de la misma. La aplicación le preguntará por el siguiente paso a dar, mostrándose en la parte derecha 4 opciones.

En la Figura 2 podemos ver un ejemplo de esta pantalla a modo de resumen, para una batalla creada con anterioridad, donde se presentan todos los datos y opciones posibles.



Figura 2. Pantalla resumen de batalla



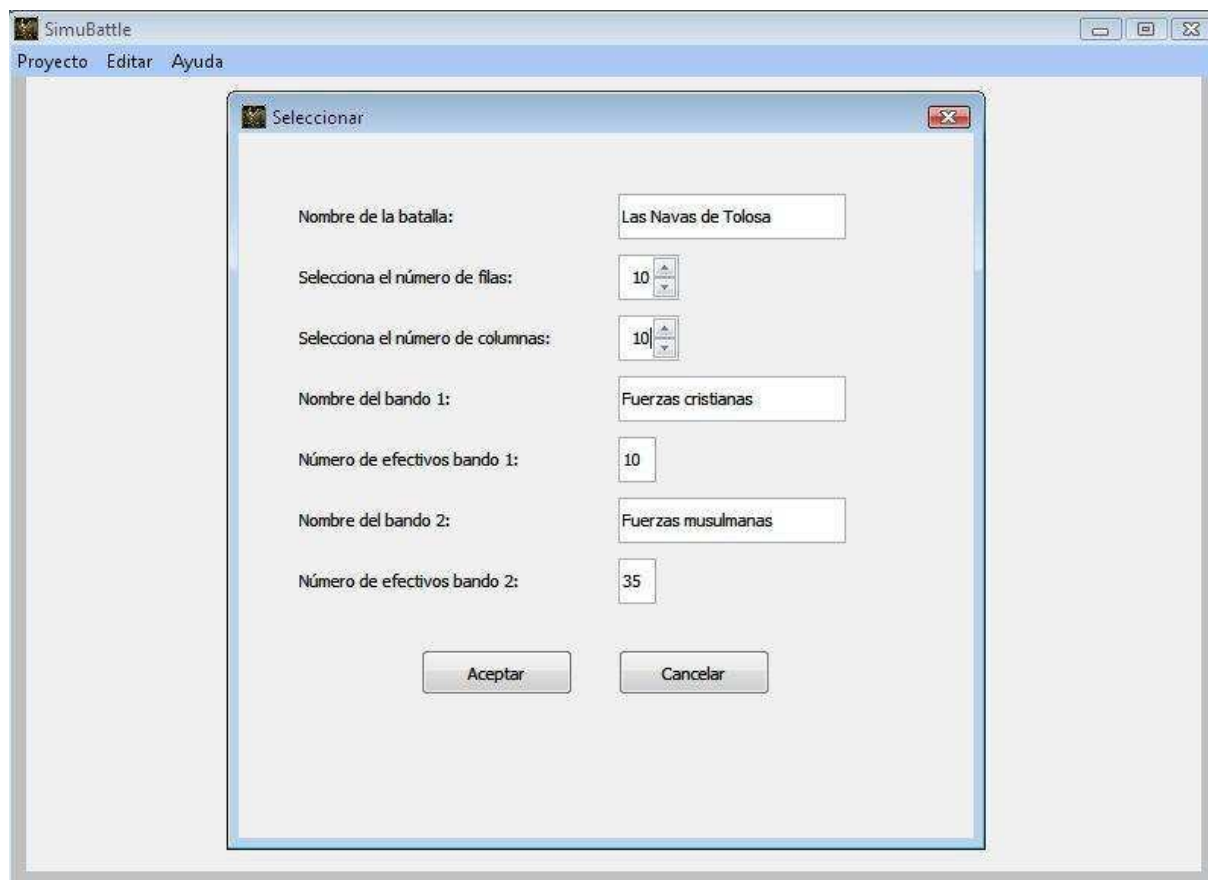
Una batalla histórica se estructura temporalmente en 3 fases: introducción, desarrollo y epílogo. Tanto la introducción como el epílogo, en SimuBattle, es información multimedia (texto, imágenes, vídeo, enlaces...), que sirve para presentar la situación histórica del momento que precedió a la batalla (introducción), y las consecuencias geopolíticas subsiguientes que tuvo dicha batalla (epílogo). Por su parte el desarrollo de la batalla es la sucesión de mapas en los que se muestran los distintos movimientos de tropas, desde su inicio hasta la capitulación.

Estas 4 opciones son:

- Editar introducción de la batalla histórica
- Crear mapas de la batalla histórica
- Parametrizar la batalla histórica
- Editar epílogo de la batalla histórica

La opción de parametrizar la batalla permanece deshabilitada, pues no es posible realizarla hasta que no está creado el primer mapa de que se compone la batalla. Tanto la introducción como el epílogo serán explicadas más adelante, junto a la introducción de comentarios de la batalla, pues su funcionamiento es exactamente el mismo. Nos resta por tanto la creación de mapas, que es la función principal. Una batalla se compone como mínimo de un mapa en el que han de estar presentes, a su vez, como mínimo una unidad por cada bando.

Tras pinchar en el botón “Crear mapas de la batalla histórica”, la aplicación abre una ventana donde se pide al usuario la introducción de las características básicas de la batalla (Figura 3).



The screenshot shows the SimuBattle application window with a menu bar containing 'Proyecto', 'Editar', and 'Ayuda'. A 'Seleccionar' dialog box is open, allowing the user to configure battle parameters. The dialog contains the following fields and values:

Field	Value
Nombre de la batalla:	Las Navas de Tolosa
Selecciona el número de filas:	10
Selecciona el número de columnas:	10
Nombre del bando 1:	Fuerzas cristianas
Número de efectivos bando 1:	10
Nombre del bando 2:	Fuerzas musulmanas
Número de efectivos bando 2:	35

At the bottom of the dialog are two buttons: 'Aceptar' and 'Cancelar'.

Figura 3. Pantalla de características de la batalla

Tanto el nombre de la batalla, como el de los bandos contendientes, son caracteres alfanuméricos a introducir por el usuario por teclado. Mientras, el número de unidades iniciales, con las que comienzan la batalla cada bando, ha de ser un número mayor que cero. Recordemos que con anterioridad citamos que el mapa se dividirá en celdas o escaques a modo de tablero, esta cantidad será la indicada por el número de filas y columnas.



La suma del número de unidades de ambos bandos, ha de ser como máximo un 66% del número de celdas. Esta restricción se produce para poder desarrollar una batalla con el suficiente espacio posible, acorde al número total de unidades.

Tras introducir correctamente estos valores, la aplicación nos preguntará por la imagen del mapa que deseamos mostrar. Por defecto la aplicación nos ofrece una variada gama de escenarios, pero usted puede escoger la imagen de su ordenador que desee, mediante la opción “Examinar...” (Figura 4).

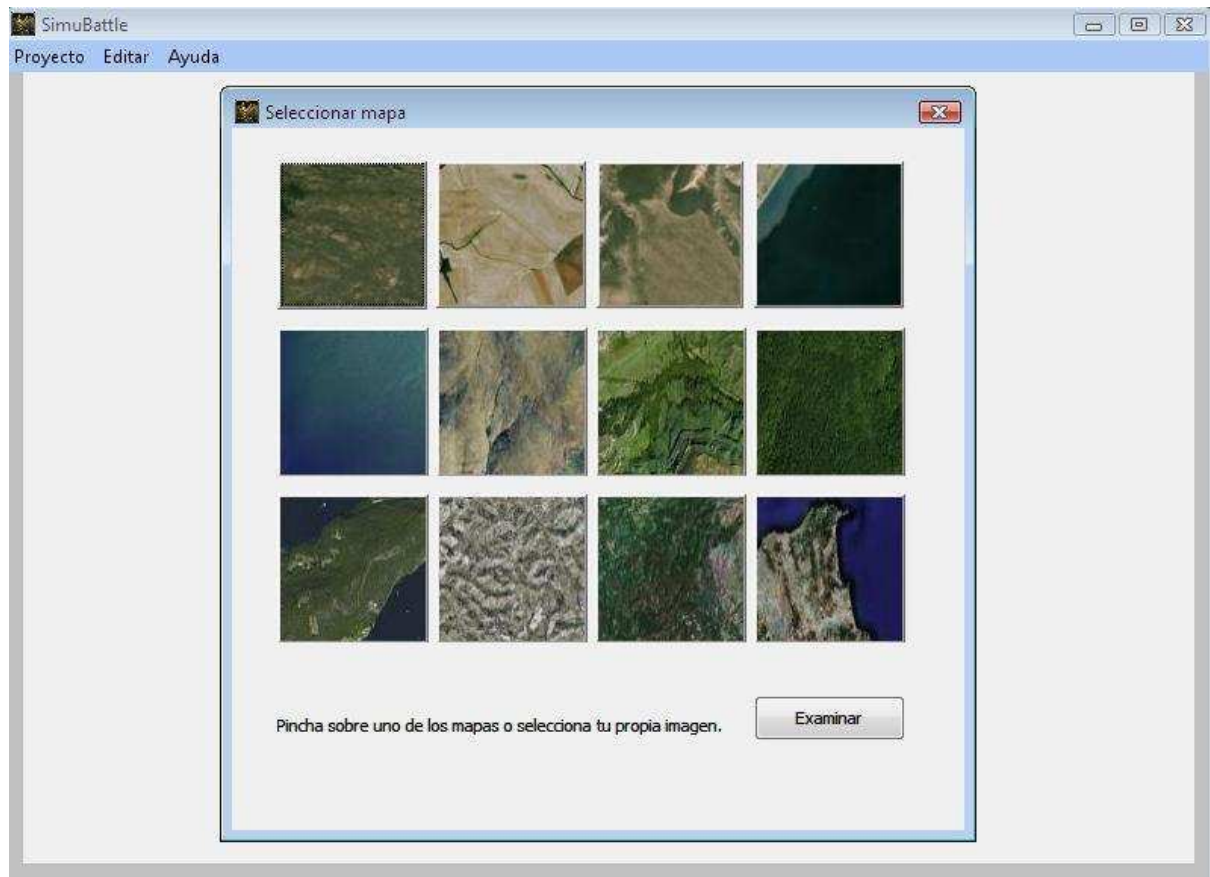


Figura 4. Pantalla de selección de fondo del mapa

Ya tenemos visible el mapa-escenario donde se desarrollará la batalla. Esta es la pantalla de edición de batalla. Como se puede apreciar, el mapa se encuentra ocupando la mayor parte de la misma, en el lado izquierdo, mientras que a la derecha se nos presentan una serie de opciones que iremos descubriendo poco a poco. En la pantalla puede verse el nombre de cada bando, junto al número de unidades de cada bando que ya han sido insertadas en el mapa, y las que restan por ser insertadas (Figura 5).

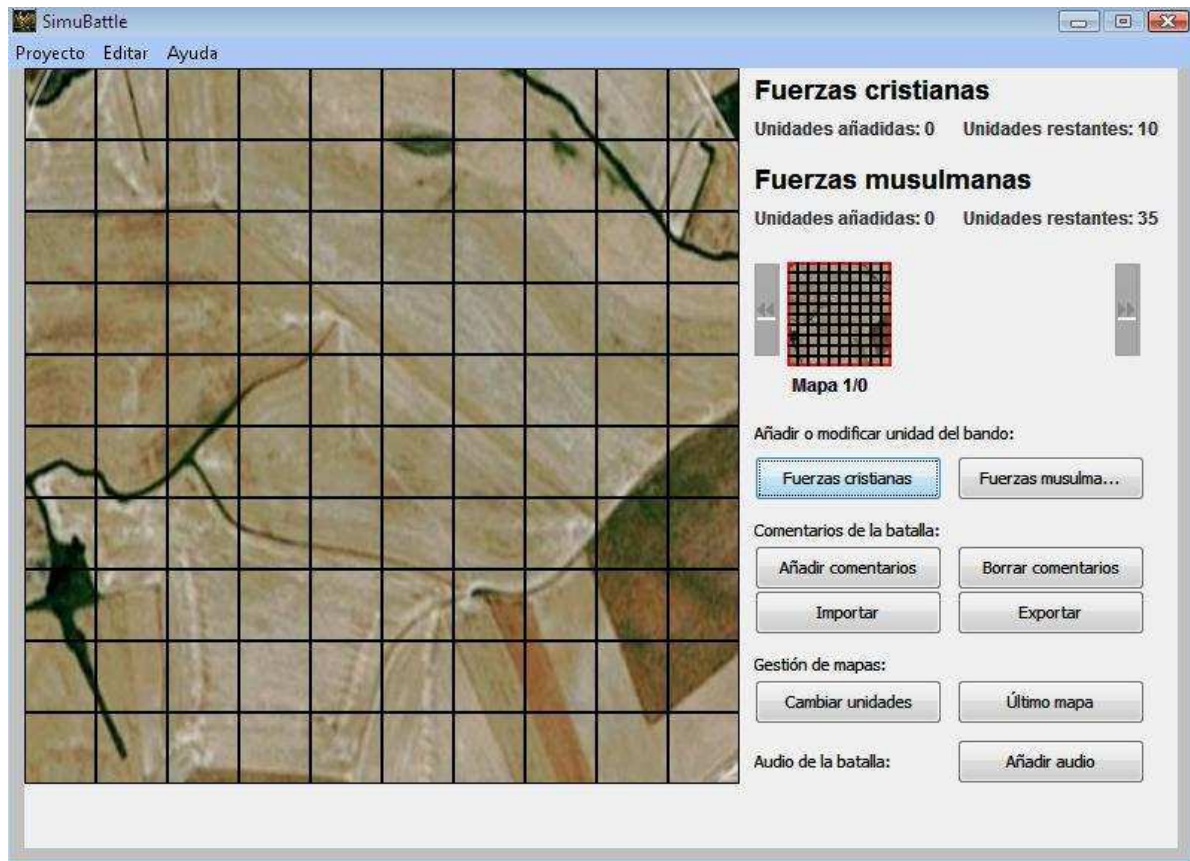


Figura 5. Pantalla principal de edición de batalla

3.1. Inserción, modificación y borrado de unidades

Comenzaremos por situar las unidades de cada bando en el mapa de la batalla. Pinchando sobre los botones con los nombres de cada bando, pasamos a la pantalla de edición de cada bando. Esta pantalla presenta una serie de botones específicos para la inserción, modificación y borrado de unidades. Para insertar una nueva unidad en el mapa, hemos de seleccionar la altura de la unidad: aérea o terrestre. Con esto posicionaremos la unidad en el plano inferior del mapa o en uno superior que ocupan las unidades aéreas. Pulsando sobre el botón “Insertar”, el mapa pasará a estar activo, y al pasar el ratón sobre el mismo, la celda activa en cada momento, donde podrá situarse la unidad, aparecerá marcada en color rojo. Pinchando sobre la celda deseada, la unidad se situará en la misma. Tras realizar esto, se nos pedirá insertar un nombre a la unidad, que por defecto tiene un identificador asignado, y se nos presenta la oportunidad de cambiar la imagen que representa a la unidad, y que por defecto se presenta una para las unidades aéreas y otra para las terrestres. Disponemos de 10 imágenes más que el programa nos ofrece, pero también es posible seleccionar una de nuestro propio ordenador mediante la opción “Examinar...”. Tras aceptar, vemos cómo la unidad ya ha sido insertada en el mapa, en la posición deseada, y que el número de unidades añadidas y restantes de este bando se actualiza (Figura 6).

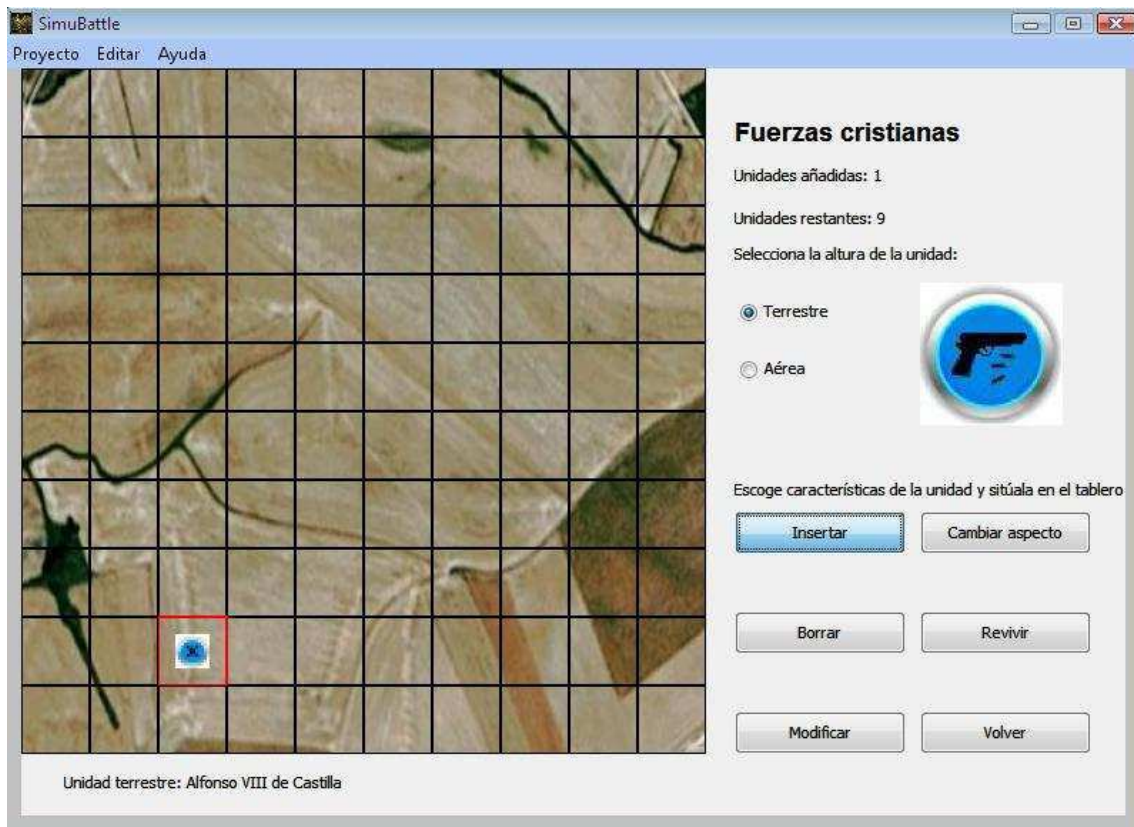


Figura 6. Inserción de una unidad en el mapa

Actuando de una manera similar, podemos eliminar una unidad ya insertada en el mapa. Pinchando sobre la opción “Borrar” y a continuación seleccionando la celda que ocupa dicha unidad, será eliminada de dicho mapa. Para modificar una unidad, seleccionamos primero la opción “Modificar” y a continuación la celda donde se encuentra la unidad, tras ser seleccionada podemos modificar el tipo de unidad así como su posición en el mapa. La opción “Cambiar aspecto”, nos permite cambiar el nombre de una unidad, así como la imagen que la representa, seleccionando dicho botón y la unidad deseada, de igual manera a la por ahora explicada.



Dos unidades terrestres, o dos aéreas, no pueden ocupar la misma celda. Recordemos que una batalla está compuesta de una sucesión de mapas. En un momento dado, las unidades que fallecen han de ser borradas de su correspondiente mapa. En este primer mapa estamos obligados a situar en el mapa todas las unidades de que disponen ambos bandos, pues es esta su situación inicial. En mapas sucesivos podremos ir borrando unidades, aquellas que se consideran fallecidas.

Si una unidad ha sido borrada de un mapa, ésta ya no podrá aparecer en los sucesivos. Si quisiéramos recuperar dicha unidad, pues consideramos que su eliminación ha sido un error, podemos retroceder hasta el mapa anterior a su fallecimiento, donde la unidad “aún vivía”, y, mediante la opción “revivir”, hacer que dicha unidad vuelva a aparecer en los mapas sucesivos.



Al revivir una unidad, ésta mantendrá la posición, en los mapas sucesivos que ya han sido creados, que ocupa en el momento de ser revivida. Podría darse el caso en que en un mapa posterior, esa posición está siendo ocupada por otra unidad, en cuyo caso, la unidad revivida, se situará en la posición más cercana posible a dicha celda.

Una vez posicionadas todas nuestras unidades, volvemos a la pantalla principal de edición de batalla mediante la opción “Volver”.

3.2. Gestión de mapas

Para pasar de un mapa a uno posterior, encontramos un “carrusel” donde se nos presenta el mapa actual, así como el anterior y el posterior si existieran. Para pasar de uno a otro basta con pulsar las flechas a derecha e izquierda que se corresponden con la opción “Siguiente”, y su consecuente “Anterior” para volver a un mapa previo. En cada mapa podemos variar las posiciones de las unidades, así como eliminarlas, para representar de la manera más fehaciente posible los hechos que sucedieron en la batalla recreada. Si deseamos marcar un mapa como el último de la batalla, obviando los posibles mapas posteriores ya creados, pulsamos, cuando nos encontramos en el mapa deseado, el botón “Último mapa”.



En el mapa número 1 es en el único en el que se nos presenta la opción “Cambiar unidades”. Pinchando sobre la misma, podemos cambiar el número de unidades de cada bando, así como el nombre de la batalla y de los mismos.



Al pulsar sobre el botón “Siguiente”, accedemos a un nuevo mapa. En ese momento el mapa ya ha sido creado y pasa a formar parte de la secuencia de la batalla, grabándose en la misma, al pulsar en “Siguiente”. De ahí que exista la opción “Último mapa”, para eliminar posibles mapas posteriores superfluos. Cabe destacar, por tanto, que si, por ejemplo, deseamos que nuestra batalla conste de 10 mapas, deberemos llegar hasta el mapa número 11 para que el 10 sea grabado. Para ayudarnos mejor en conocer el número de mapas de nuestra batalla, en el carrusel aparece en la parte inferior de cada mapa el número que le corresponde, así como el número total de mapas de que consta la batalla hasta el momento. Una batalla consta como mínimo de un mapa, por lo que deberemos pulsar como mínimo una vez el botón “Siguiente”, para poder visualizar posteriormente nuestra batalla.

3.3. Audio de la batalla

Es posible agregar música o sonidos mp3 a la recreación de una batalla. Estos ficheros mp3 son reproducidos durante la visualización de la batalla, y pueden existir tres distintos, para la introducción, el desarrollo y el epílogo de la batalla. Para asociar estos ficheros mp3, pinchamos en el botón “Añadir audio”. Una pantalla con las 3 opciones posibles se nos desplegará, pudiendo agregar el audio a la parte de la batalla deseada (Figura 7).



Figura 7. Pantalla de inclusión de audio

3.4. Comentarios de la batalla

Es posible agregar comentarios a cada mapa de la batalla, e incluso generar una presentación y un epílogo a la misma. Al pulsar sobre cualquiera de estas opciones, se nos abre un editor de textos con todo lo necesario para generar comentarios HTML, lo que nos permite introducir enlaces, tablas, imágenes, vídeos... (Figura 8).

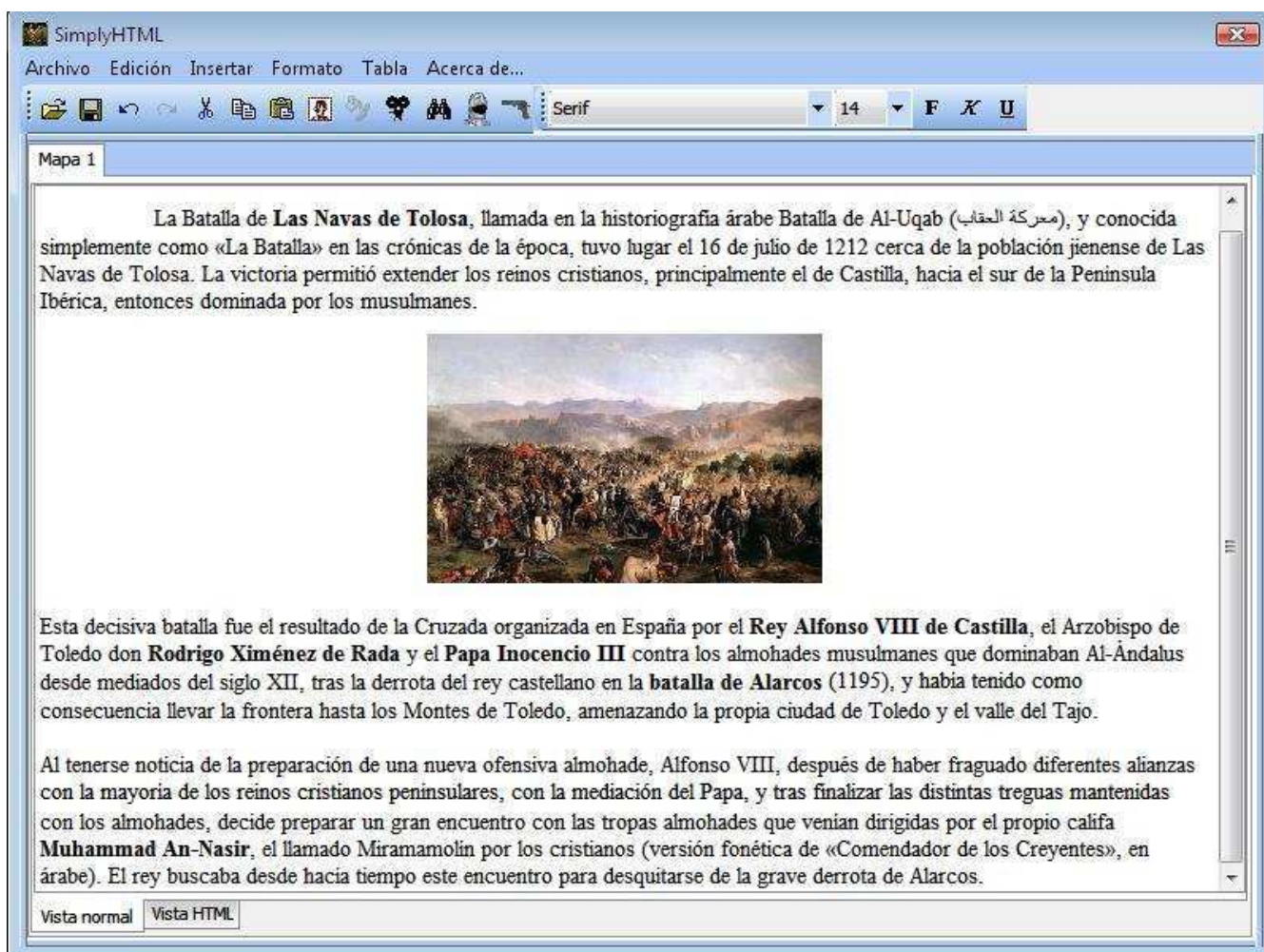


Figura 8. Pantalla de inserción de comentarios

Su funcionamiento es similar al de cualquier editor de textos, por lo que no detallaremos todas las opciones presentes, pero sí aquellas que son características de SimuBattle, y que son las siguientes:

- Inserción de video: Esta funcionalidad nos permite insertar videos del conocido portal Youtube en nuestro archivo HTML de comentarios. Al pulsar sobre esta opción, se nos presenta una sencilla pantalla donde se nos pide un título del video, una anchura y altura de la pantalla visualizadora del video medidas en píxeles, y un color de fondo. El campo principal es URL, donde debemos insertar la dirección de la página del video que deseamos insertar, tal y como vemos en la figura 9.

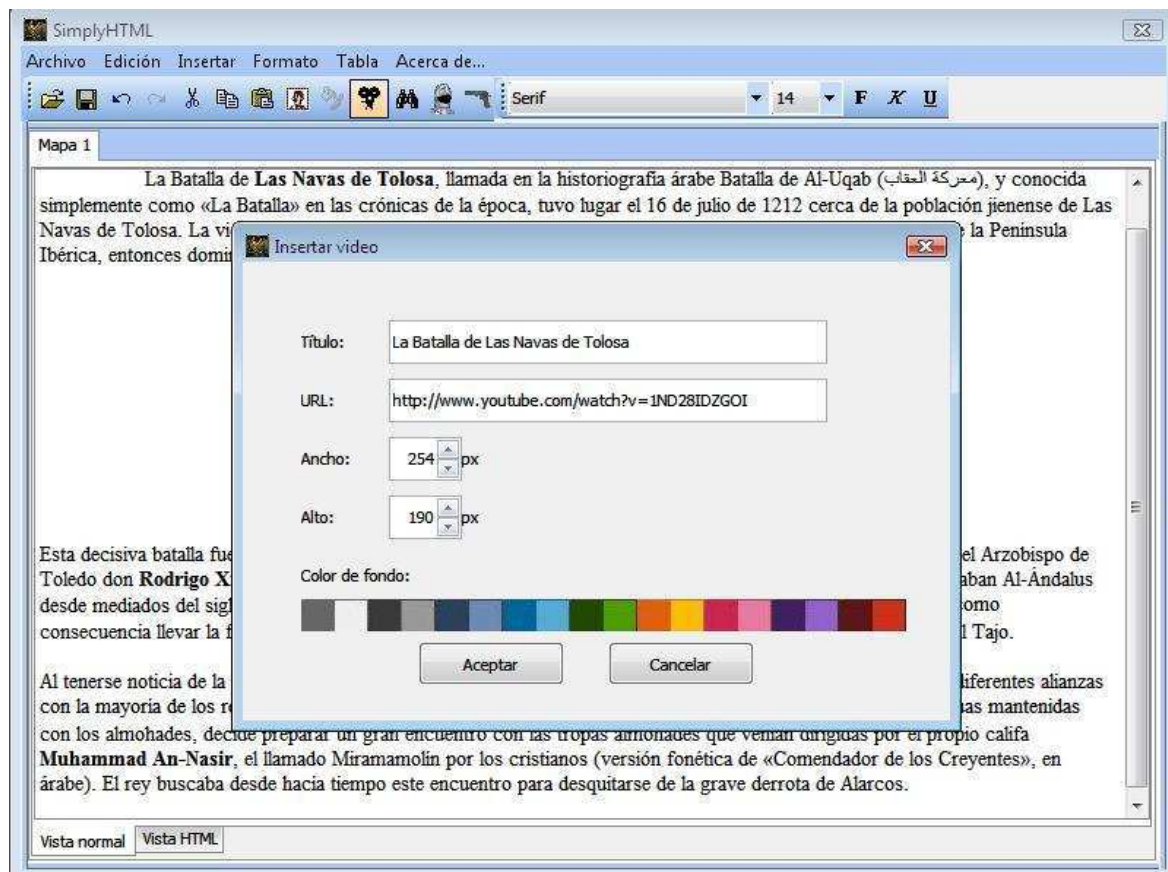


Figura 9. Pantalla de inserción de video

- Asociar comentarios a una unidad: La visualización de una batalla, se realiza, mediante la presentación de la secuencia de mapas creados, pero los movimientos de las distintas unidades se realiza de uno en uno. Esto nos da la posibilidad de seleccionar el orden en que se mueven estas piezas, así como de asignarle comentarios específicos a ese movimiento.

Mediante la opción “Asociar unidad” podemos realizar esta acción. Al pulsar sobre el icono (🖱️), se nos abre una representación en pequeño del mapa, con todas las unidades presentes. Pinchando sobre la unidad deseada, en nuestro editor aparecerá el nombre de dicha unidad como un enlace de color azul y subrayado. Esta marca, que no se visualizará en los posteriores comentarios, sino que es de uso “interno” de la aplicación, es la que indica la unidad a la que se asocia el comentario. Este comentario serán todos aquellos anteriores a la marca de unidad que sean texto subrayado. En la Figura 10 vemos un pantallazo de la edición y de cómo se visualizaría posteriormente durante la batalla (Figura 11).

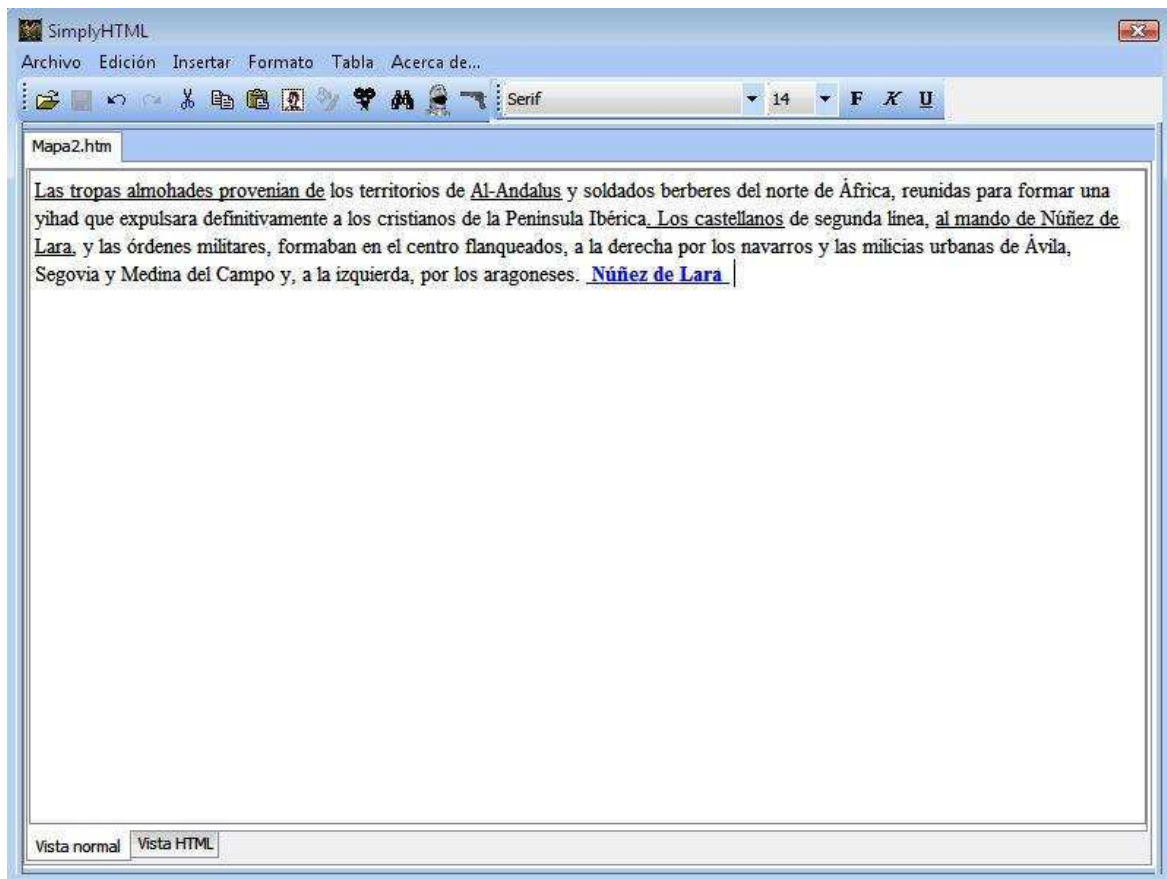


Figura 10. Ejemplo de comentarios asociados a una unidad

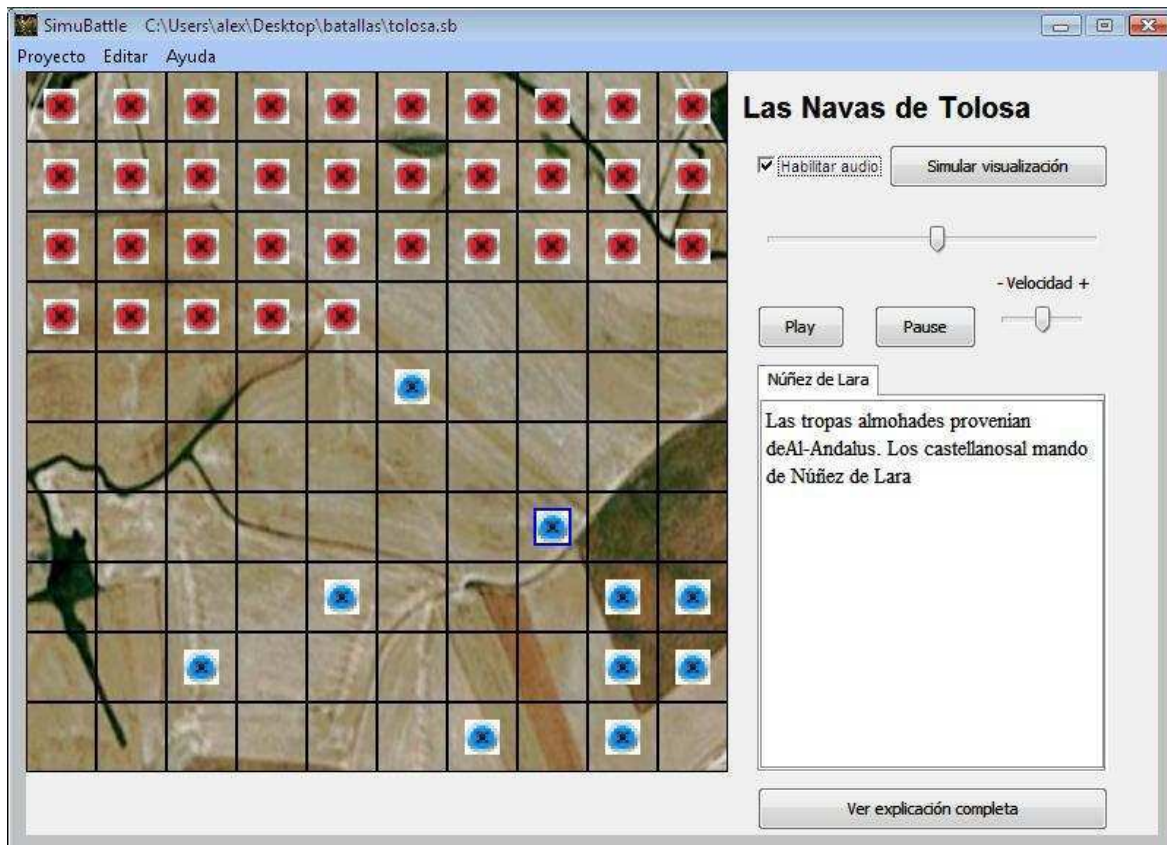


Figura 11. Visualización de los comentarios asociados a una unidad

Además tenemos la posibilidad de representar un disparo a enemigos, para ello lo indicamos mediante marcas de enemigos, pulsando la opción “Asociar enemigo”, cuyo funcionamiento es similar. Así pues, en el siguiente ejemplo (Figura 12) podemos apreciar cómo la unidad a la que está asociada el movimiento, y que está marcada de color azul, realiza un disparo a las unidades marcadas en rojo, todo ello expresado en el editor de texto tal y como aquí se ha explicado.

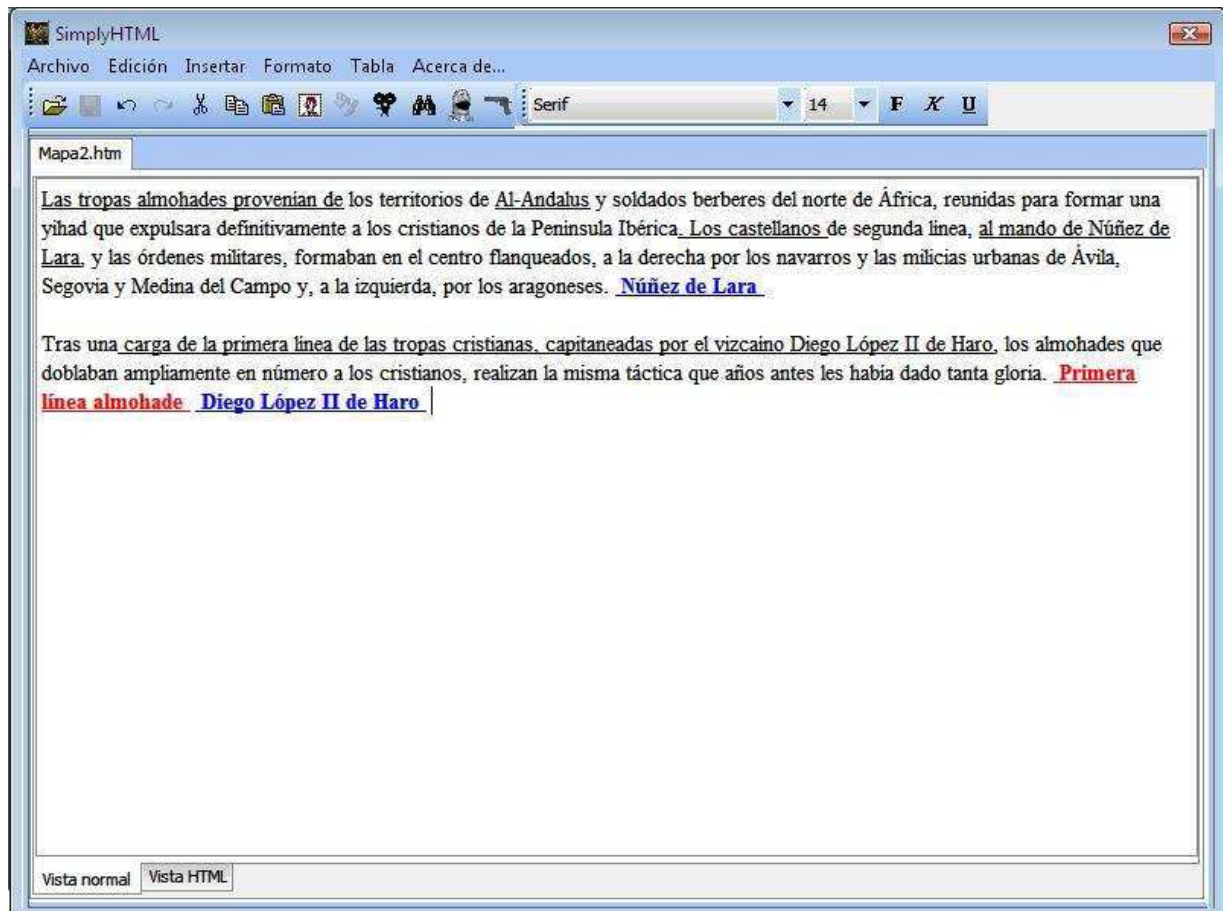


Figura 12. Comentarios asociados a unidades y enemigos



Figura 13. Visualización de comentarios asociados a unidad y enemigo

Antes hemos indicado que en el editor es posible indicar el orden de los movimientos en el tablero durante su visualización. Esto es así porque la visualización reproduce los distintos movimientos siguiendo el orden de las marcas de unidades, siempre y cuando estos movimientos sean posibles. A continuación se realizará el movimiento de las unidades sin marca en los comentarios asociados al mapa, de manera aleatoria.



A continuación podemos ver un ejemplo de este funcionamiento: Las siguientes imágenes muestran dos mapas consecutivos, y se puede apreciar que las unidades se mueven todas a su derecha.

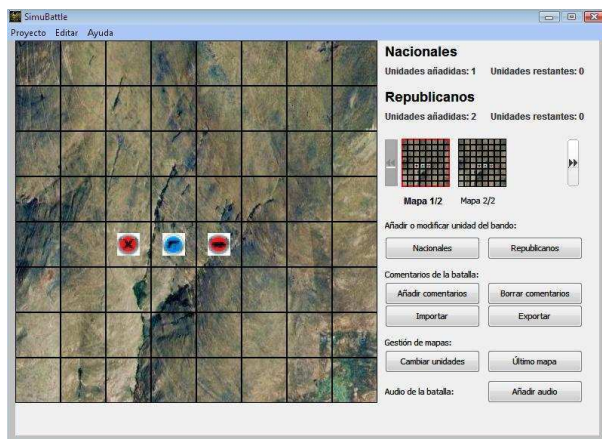


Figura 14. Ejemplo Mapa 1

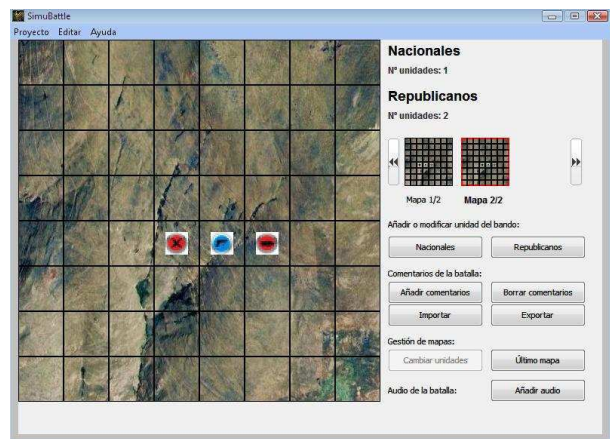


Figura 15. Ejemplo Mapa 2

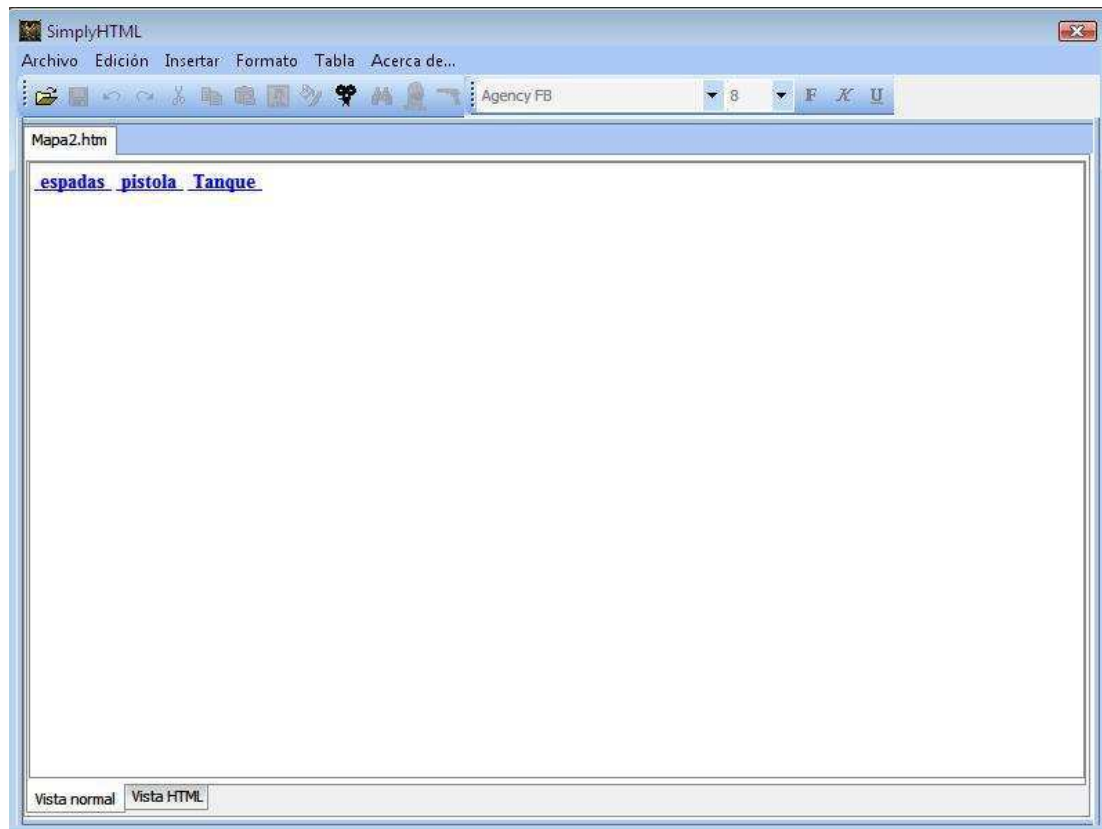


Figura 16. Comentarios asociados al mapa

Según el editor, el orden de movimiento de las unidades será: espadas, pistola, tanque, y a continuación aleatoriamente el resto de unidades si existieran (que en este caso no existen).

En cambio vemos como el movimiento final es: Tanque, pistola, espadas. ¿Por qué ha sucedido esto?

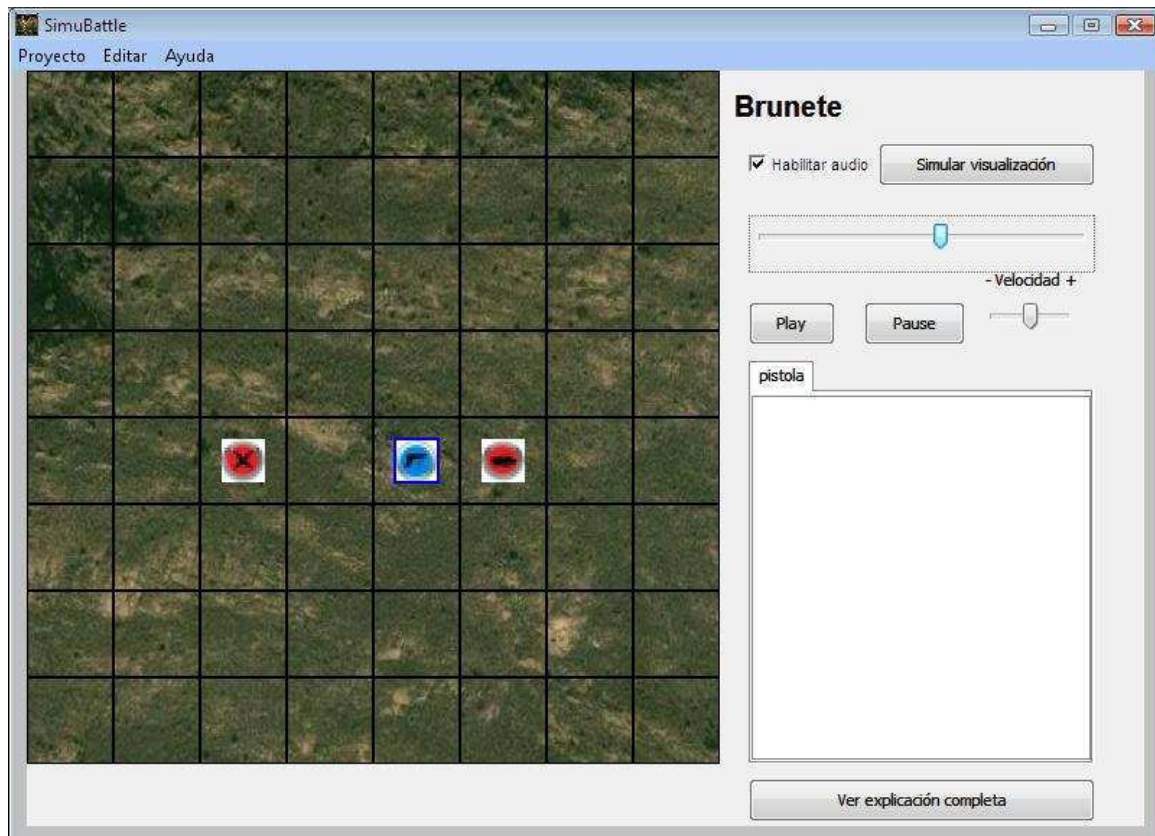


Figura 17. Visualización de la batalla

La explicación es muy sencilla. En primer lugar tratan de mover aquellas unidades que tienen asociadas comentarios, en el orden en que en ellos aparecen, pero en caso de que no puedan realizar su movimiento, porque la celda de destino está en ese momento ocupada por otra unidad, reserva su movimiento para más tarde. Esto es precisamente lo que ocurre aquí: En primer lugar trata de mover la unidad “espadas”, como su celda de destino está en ese momento ocupada por la unidad “pistola”, cede su turno. Lo mismo ocurre con “pistola”, cuya celda de destino está ocupada por “Tanque”. Finalmente “Tanque” sí puede realizar su movimiento, posteriormente “pistola” ya puede efectuar el suyo, y finalmente es “espadas” quien mueve a su siguiente celda.

Tras crear tantos mapas como deseemos, y haber creado los comentarios o la introducción y/o el epílogo, ya estamos en disposición de guardar nuestra batalla. Para ello pulsamos en Proyecto -> Guardar, y seleccionamos un nombre para nuestro proyecto, que se guardará con extensión SimuBattle (sb). Ya tenemos nuestra primera batalla creada y disponible en un fichero para trabajar con ella posteriormente.

4. Visualización de la batalla

Con tan solo un mapa creado, ya se considera creada nuestra batalla y podemos visualizarla. Al pulsar en esta opción, vemos como en el mapa de la izquierda, se sitúan las unidades tal y como las hemos dispuesto en nuestro

primer mapa. Mientras que a la derecha cambian las opciones, presentándonos las correspondientes a la visualización.



Figura 18. Pantalla de visualización de la batalla

Se nos presenta la opción de habilitar o no el audio durante la visualización, esto es, los ficheros de sonido agregados durante la edición de la batalla serán o no reproducidos. Mediante el selector de velocidad, dotamos a la visualización de mayor o menor rapidez, a nuestro antojo.

Al pulsar el botón de “Play”, la batalla empezará a desarrollarse, sucediéndose los movimientos de las unidades tal y como hemos especificado en la edición de la batalla. En cualquier momento podemos detener el desarrollo de la batalla pulsando el botón “Pause”.

Una barra de desplazamiento marca en todo momento el tiempo restante de la batalla y el que se lleva visualizado, pudiendo moverla hasta el momento temporal deseado por el usuario.

Si observamos, los movimientos de cada unidad son marcados en el mapa mediante un sombreado de color azul, mientras que las piezas atacadas lo son de color rojo. Así, en la parte derecha, aparecen los comentarios asociados a cada unidad y que se generaron en el editor de textos HTML. Estos comentarios mostrados en esta pantalla, no son más que un breve retazo en texto plano de los comentarios asociados al mapa. Para poder ampliar la información, podemos pulsar la opción “Ver explicación completa”. Al pulsar en ella nos aparecerá una pantalla de mayor tamaño donde podemos apreciar todo el texto asociado al mapa. Para poder disfrutar de toda la funcionalidad que el texto HTML nos ofrece, pulsando el botón “Ver en

navegador”, se nos abrirá el navegador que por defecto tenga definido nuestro ordenador, y podremos apreciar en su totalidad los comentarios diseñados en el editor de textos con anterioridad.

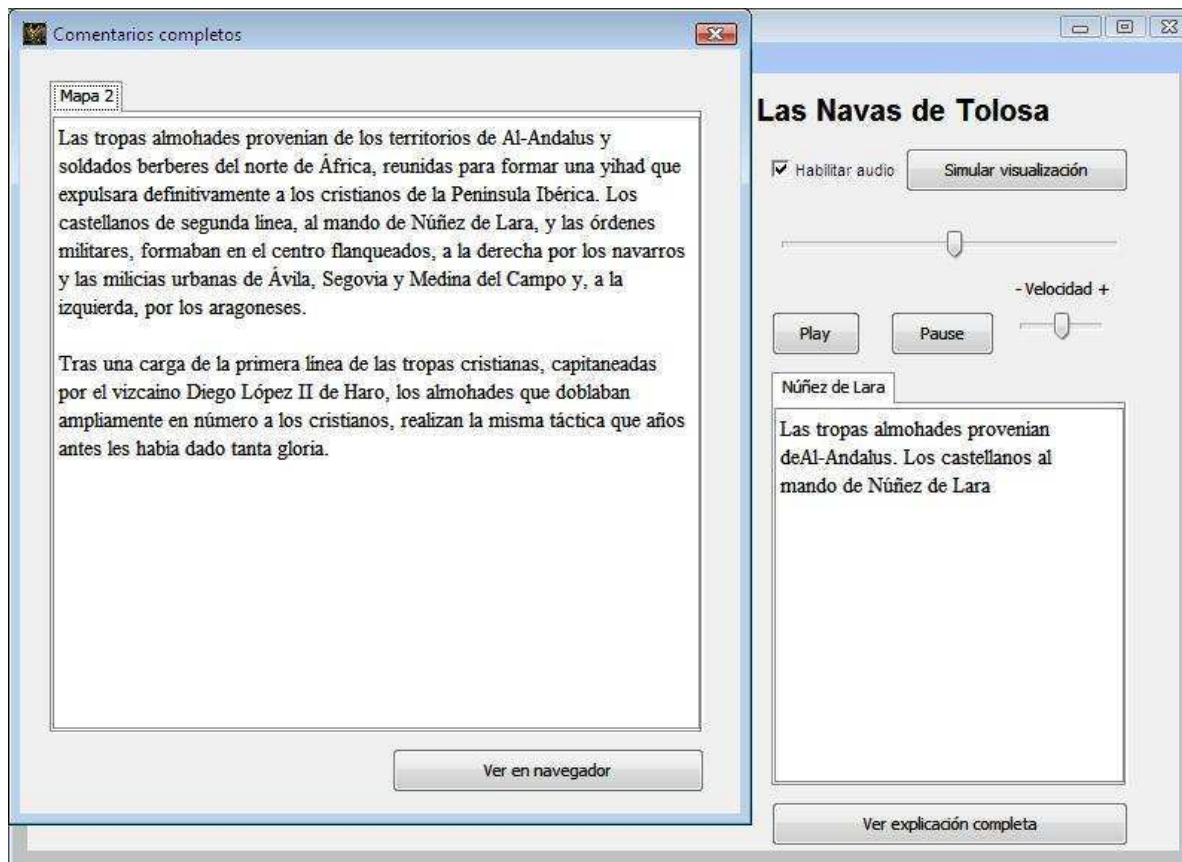


Figura 19. Visualización de los comentarios completos de un mapa

Una opción muy importante que se nos ofrece es “Simular visualización”. Esta opción permite tomar la visualización en un momento exacto que se nos antoje, y llevarla al simulador para representar, a partir de ese punto, la batalla a nuestro antojo. Las posibilidades que se nos ofrecen durante la simulación serán explicadas más adelante.

5. Parametrización

Esta funcionalidad nos permite asignar valores a los parámetros de las celdas y las unidades, parámetros que serán los que determinen en la simulación la forma de actuar de los distintos contendientes de la batalla. Esta funcionalidad no está activa hasta que la batalla no contiene, como mínimo, un mapa. Tal y como vemos a continuación, la pantalla de la izquierda nos muestra el mapa inicial de la batalla, mientras que a la derecha se nos muestran todas las posibles opciones a realizar.

Podemos seleccionar una o varias celdas, o una o varias unidades de una sola vez. Existe la opción de seleccionar todas las celdas terrestres o aéreas de una sola vez, o todas las unidades de uno o de los dos bandos, terrestres o aéreas nuevamente, según la combinación de botones que se seleccione. Según la combinación seleccionada, en el mapa aparecen

marcadas en color rojo las celdas o unidades que se ven afectadas por la acción, tal y como apreciamos en la imagen anterior.



Figura 20. Pantalla principal de parametrización

Al pulsar el botón “parametrizar”, se nos abre un nuevo grupo de opciones en el lado derecho, y que pueden ser:

- Celdas:
 - Transitabilidad: Valor entre 1 y 9, que indica lo accesible que es la celda. Cuanto mayor es el valor, las unidades atacan y se defienden mejor de sus enemigos. Por tanto, las unidades tienden a situarse en celdas lo más transitables posibles.
 - Accesible: Parámetro que marca si una celda es accesible o no, es decir, si las unidades de batalla pueden ocupar dicha celda.
 - Naval: Parámetro presente sólo en las celdas de tipo terrestre (no en las aéreas) y que indica si una celda es de tipo naval, lo que provoca que sólo puedan ocupar dicha celda unidades navales.
- Unidades terrestres:
 - Tipo de unidad: Permite seleccionar el tipo de la unidad, es decir, si es una unidad terrestre normal, un cañón o una unidad de tipo naval.

- Vida: Parámetro entre 1 y 11 que indica la cantidad de vida de cada unidad, siendo 11 el valor máximo.
- Moral: Parámetro entre 0 y 10 que indica la cantidad de moral de cada unidad. Cuanto mayor es la moral, mejores son los ataques y defensas de la unidad.
- Ataque: Parámetro entre 0 y 10 que indica la calidad de los ataques de la unidad.
- Defensa: Parámetro entre 0 y 10 que indica la calidad de los ataques de la unidad.
- Visión: Valor entre 1 y 9 que indica la distancia, medida en número de celdas, a la que puede atacar a un enemigo. Este valor no está presente para las unidades de tipo “Cañón”.
- Inteligencia: Valor entre 0 y 10 que indica la capacidad de la unidad para conocer los parámetros de una unidad rival. Cuanto mayor es el valor, menos errores cometerá en sus cálculos.
- Solidaridad: Parámetro que indica si una unidad acude en ayuda de compañeras suyas en caso de que éstas sean atacadas y reclamen ayuda.
- Radio: Parámetro sólo presente en unidades de tipo “Cañón”. Estas unidades, al atacar, pueden causar daños colaterales a celdas adyacentes. Radio es un valor que varía entre 1 y 9. En el caso de valor 1, los ataques no causan daños colaterales. En caso máximo, ataca a la unidad en concreto, y causa daños a las 8 celdas adyacentes. Las celdas adyacentes donde se generan los daños colaterales, son escogidas aleatoriamente en cada ataque.
- Alcance mínimo: Parámetro sólo presente en unidades de tipo “Cañón”. Estas unidades no pueden atacar a una unidad situada en una celda contigua a ellas, por lo que el valor mínimo es 2 y el máximo 10. Indica el número mínimo de celdas contiguas donde puede efectuar su ataque.
- Alcance máximo: Parámetro sólo presente en unidades de tipo “Cañón”. Indica el número máximo de celdas contiguas donde puede efectuar su ataque. Forma junto con el “alcance mínimo”, el radio de acción de la unidad “Cañón”. Los valores de “Alcance mínimo” y “Alcance máximo”, son el equivalente al parámetro “Visión” en el caso de los cañones. En caso de que el alcance mínimo sea mayor que el máximo, el primero adoptará el mismo valor que el segundo.
- Unidades aéreas: En este caso, los parámetros presentes, cuya funcionalidad es similar a la de las unidades terrestres, son: Vida, Moral, Ataque, Defensa, Visión, Inteligencia y Solidaridad.

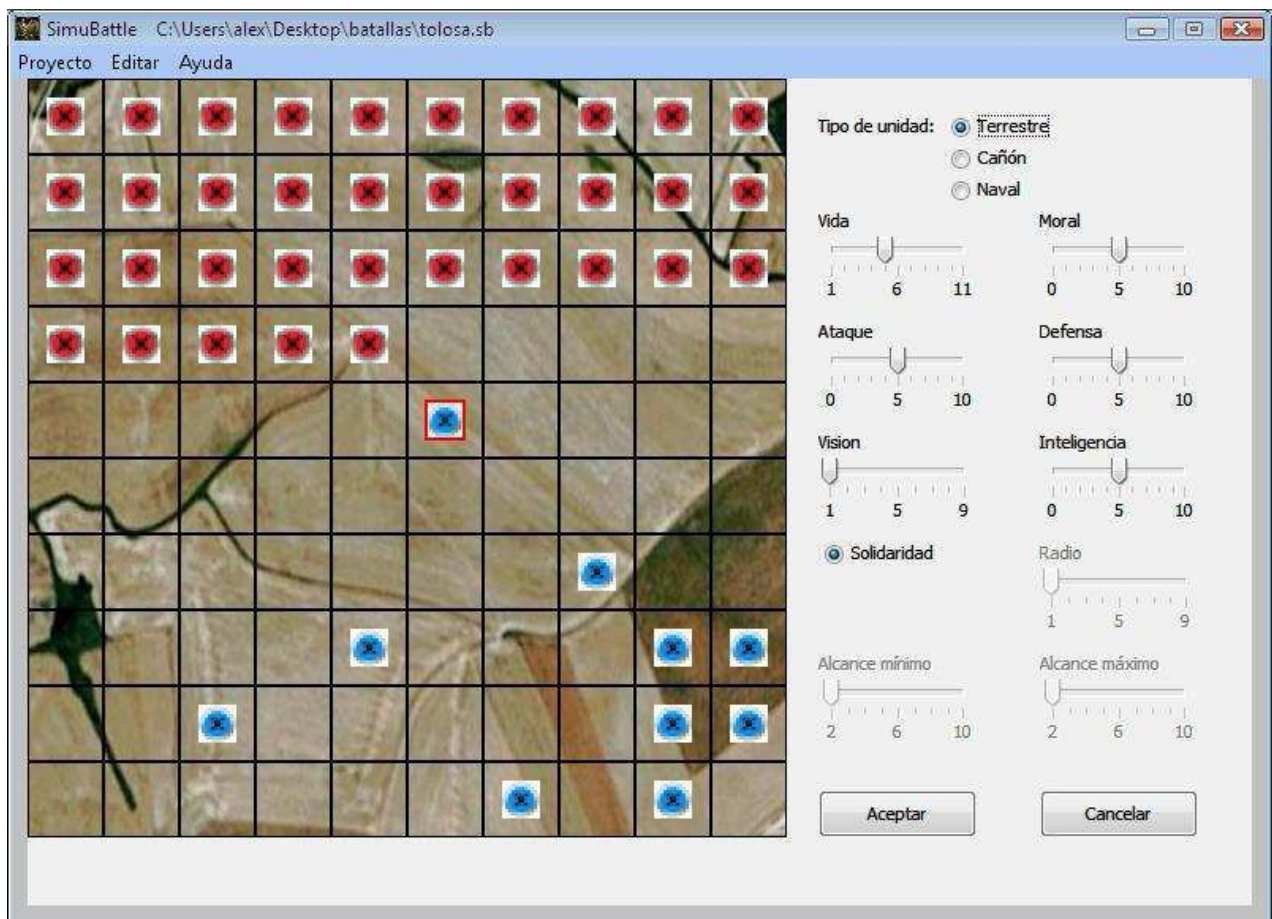


Figura 21. Pantalla de parametrización de una unidad

Al pulsar “Aceptar”, quedan grabados los parámetros para las unidades seleccionadas. Hay que destacar que, puesto que es posible seleccionar varias unidades de una sola vez, en caso de tener valores de parámetros diferentes, éstos se visualizarán en sus valores por defecto al seleccionar dichas unidades.

Tras asignar los valores deseados a los parámetros, podemos validar nuestra parametrización pulsando el botón “Validar”. En caso de ser correcta, el programa saldrá de la pantalla de “Parametrización”, en caso contrario un mensaje nos anunciará los posibles errores cometidos para poder proceder a su corrección, como por ejemplo que una unidad naval está ocupando una celda que no lo es.

6. Simulación

La pantalla sigue un formato como el que se puede apreciar en la Figura 22, donde el mapa de la batalla se sitúa en el lado izquierdo, y se presentan distintos botones a la derecha y bajo el mapa.

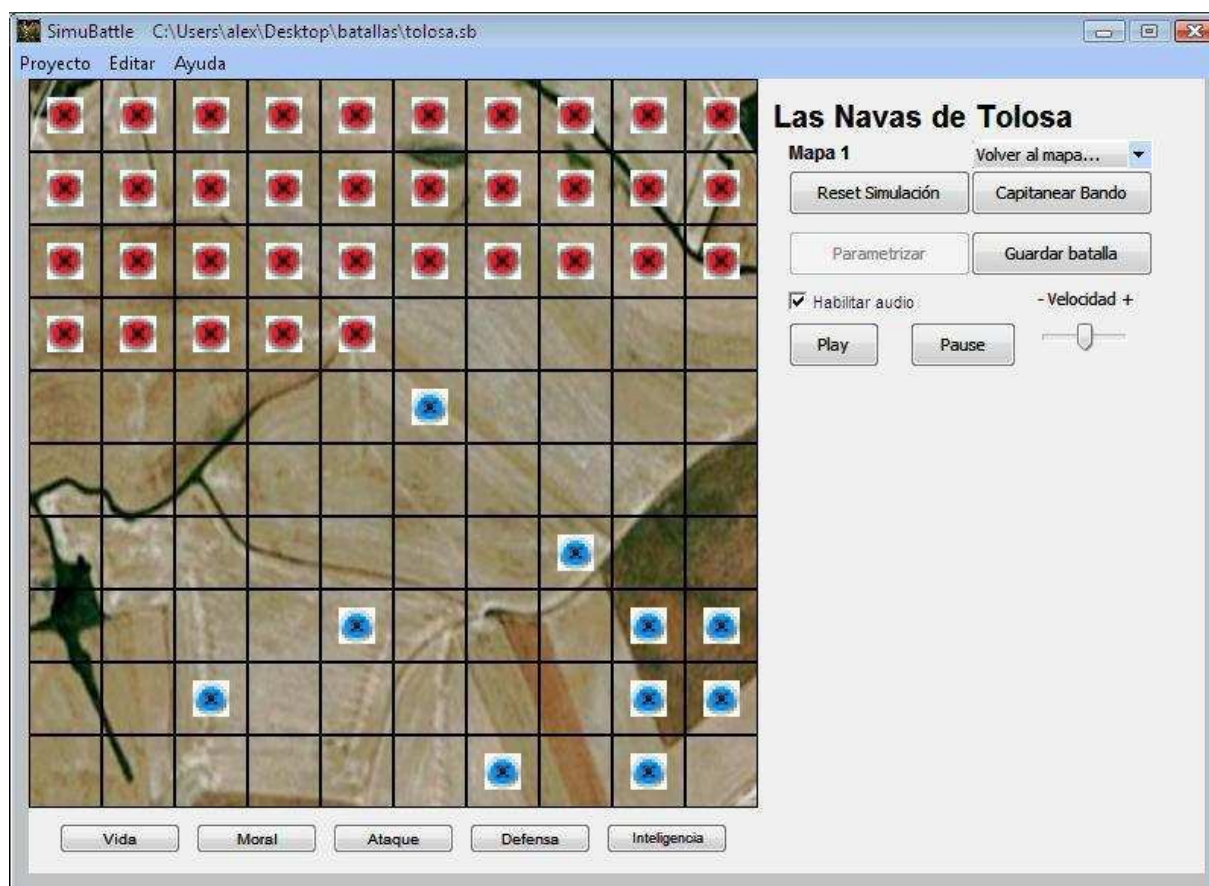


Figura 22. Pantalla principal de simulación

La primera vez que accedemos a la simulación, el mapa presentado es igual al primer mapa de la batalla editada, y cada unidad tiene por parámetros, los especificados anteriormente en la parametrización. Si pulsamos el botón “Play”, la simulación automática comenzará (notar que podemos fijar la velocidad de la misma a nuestro antojo, al igual que habilitar o no el audio de la batalla). En cualquier momento podemos detener la simulación pulsando el botón “Pause”, al igual que se hacía con la visualización. Podemos apreciar como los movimientos de las tropas se efectúa de una en una, marcándose en color azul en el mapa, la unidad que realiza el movimiento, y en rojo la unidad que recibe el ataque en caso de que así sucediera. En todo momento, en la parte derecha aparece un texto explicativo del movimiento que realiza la unidad.

En la parte inferior del mapa, aparecen 5 botones que coinciden con los 5 parámetros o características básicas de una unidad y que permiten conocer el estado de la misma (Vida, Moral, Ataque, Defensa e Inteligencia). Al pulsar sobre cualquiera de ellos, en el mapa aparece mediante un código de colores el estado de cada unidad para ese parámetro, oscilando entre el color verde (máximo) y el color rojo (mínimo) pasando por una gama de tonos amarillos que indican valores intermedios.

El botón “Reset simulación” nos permite volver al estado inicial de la misma y comenzar de nuevo la simulación. Al hacer esto, las unidades tomarán nuevamente como valores a sus características, la parametrización efectuada con anterioridad. Esto nos permite, cambiar la parametrización inicial y realizar una nueva simulación con estos nuevos datos.

También es posible retroceder al mapa anterior que deseemos de una simulación en que nos encontremos, seleccionando el mismo en el combo “Volver al mapa...”.

En un momento dado de la simulación podemos desear cambiar los parámetros de la batalla, sin tener que volver al mapa inicial o que reiniciar la simulación. Esta opción es posible, pulsando en el botón “Parametrizar”, accederemos a una pantalla similar a la parametrización ya conocida y explicada con anterioridad, pero con la particularidad de que en este caso dotamos de valores a las características de las unidades en ese momento exacto de la simulación, con dicha disposición de las unidades. Tras validar la parametrización, podemos comprobar que retornamos a la Simulación en el mismo punto en que lo dejamos, con la salvedad de que los parámetros de las unidades ahora son los que hemos asignado en esta parametrización. Al resetear la simulación, volveremos a la parametrización inicial de la batalla. Lo mismo ocurrirá si regresamos a un mapa anterior a esta parametrización de la simulación, pues al retroceder volvemos a la simulación tal y como se encontraba en dicho mapa.

Tenemos la posibilidad de participar activamente en la simulación de la batalla, tomando parte de la misma. Al pulsar el botón “Capitanear bando”, se nos abre una pantalla donde se nos muestra el mapa actual en un menor tamaño tal y como podemos ver en la Figura 23.

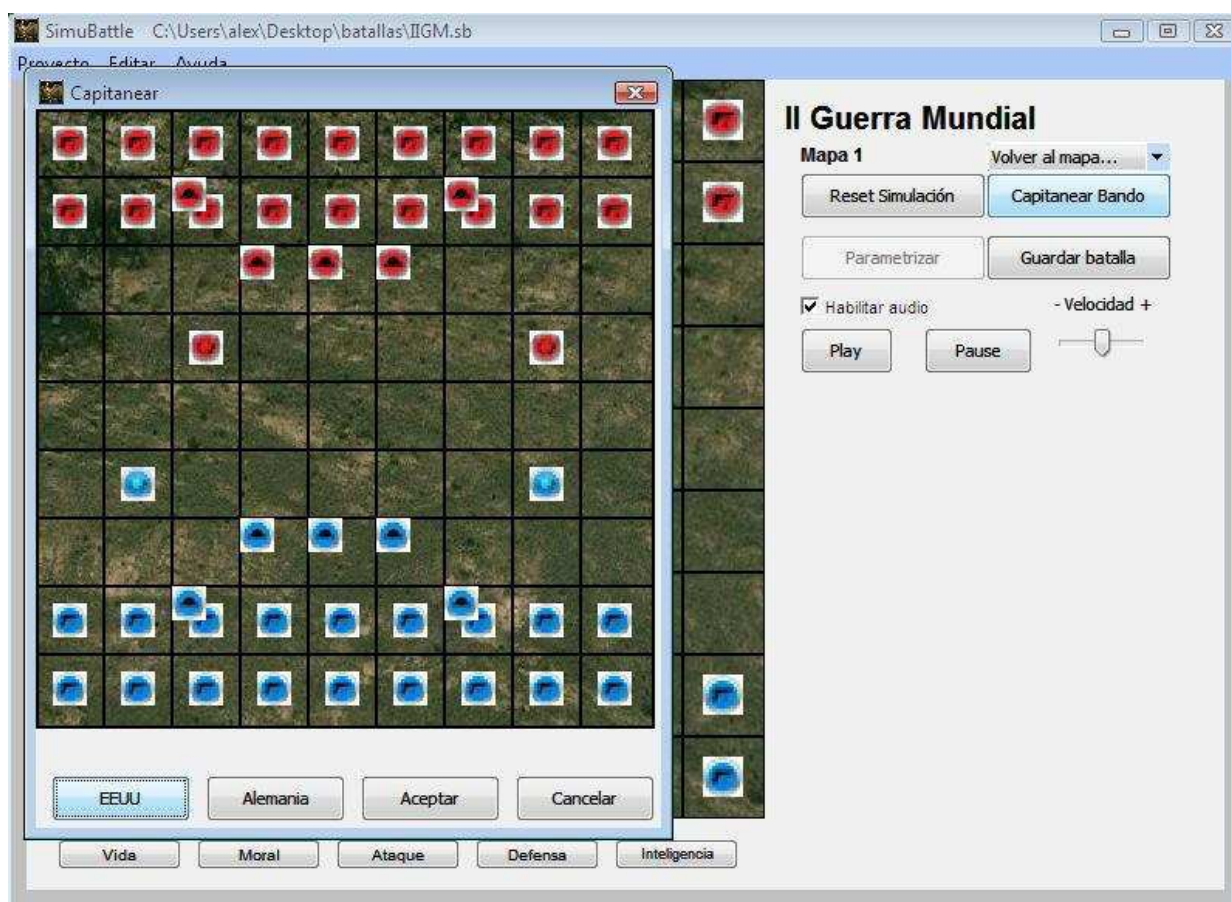


Figura 23. Pantalla de selección de unidades a capitanear

En dicho mapa, podemos seleccionar, pulsando sobre las unidades, aquellas que deseemos dirigir, o incluso seleccionar uno o los dos bandos por

completo pulsando los botones con los nombres de los mismos. Así, durante la simulación, cuando sea el turno de una unidad capitaneada por el usuario, se presentará un combo con las posibles acciones a realizar por la unidad, entre las que se encuentran movimientos en los distintos puntos cardinales, permanecer inmóvil o disparar tanto horizontal como verticalmente.

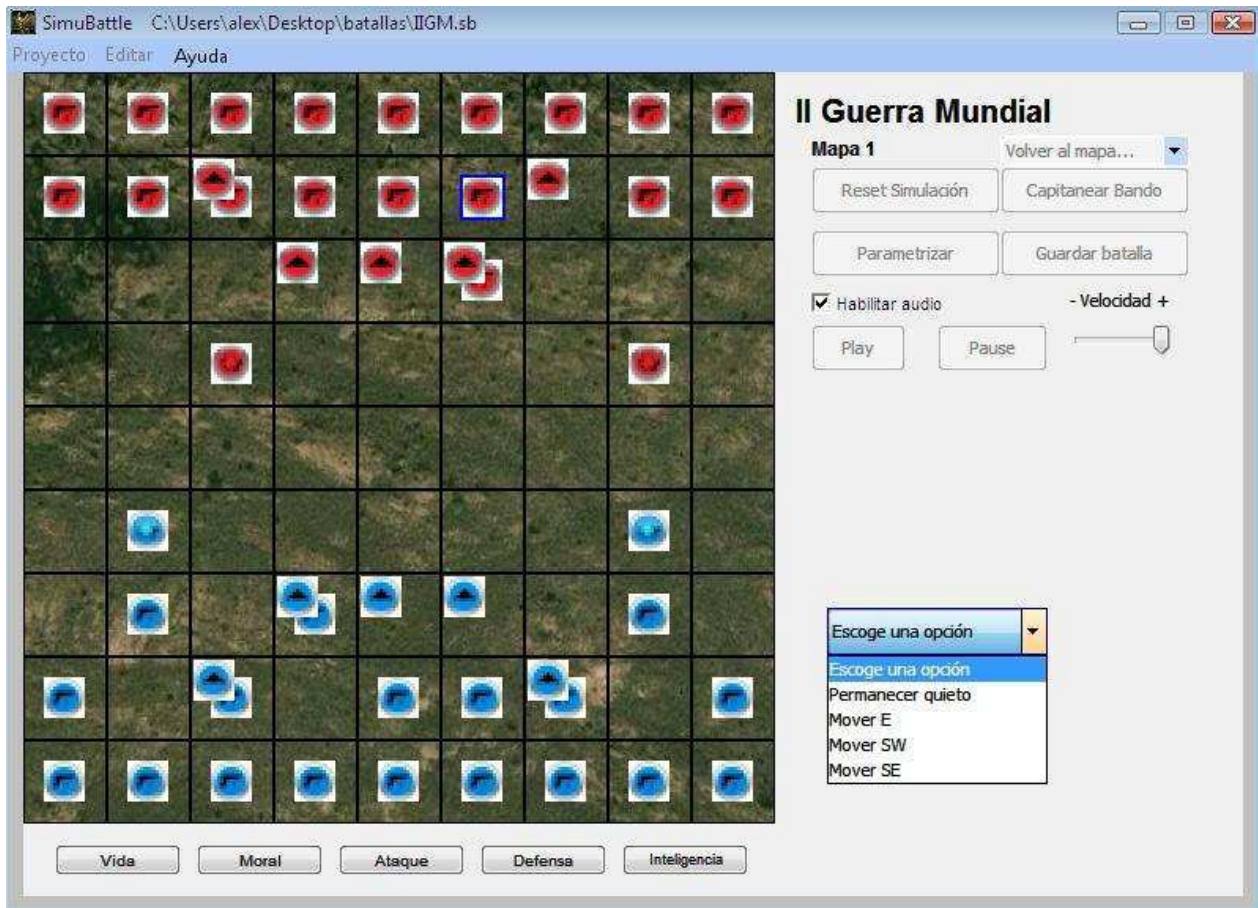


Figura 24. Pantalla de simulación donde se selecciona manualmente el movimiento de una unidad capitaneada

En el caso de seleccionar un disparo, se abrirá nuevamente un mapa en menor tamaño, donde sólo estarán representadas las unidades que están a nuestro alcance para ser atacadas. Al pasar el ratón sobre las mismas, se nos muestra información sobre el enemigo. ¡Ojo! Esta información no es fiable al cien por cien, pues es la que la unidad capitaneada, dada su inteligencia, ha calculado como posible. Recordemos que cuanto mayor inteligencia posea la unidad, más acertados serán sus cálculos respecto a los enemigos. Una vez decidida la unidad que queremos atacar, pulsamos sobre la misma y la simulación continuará.

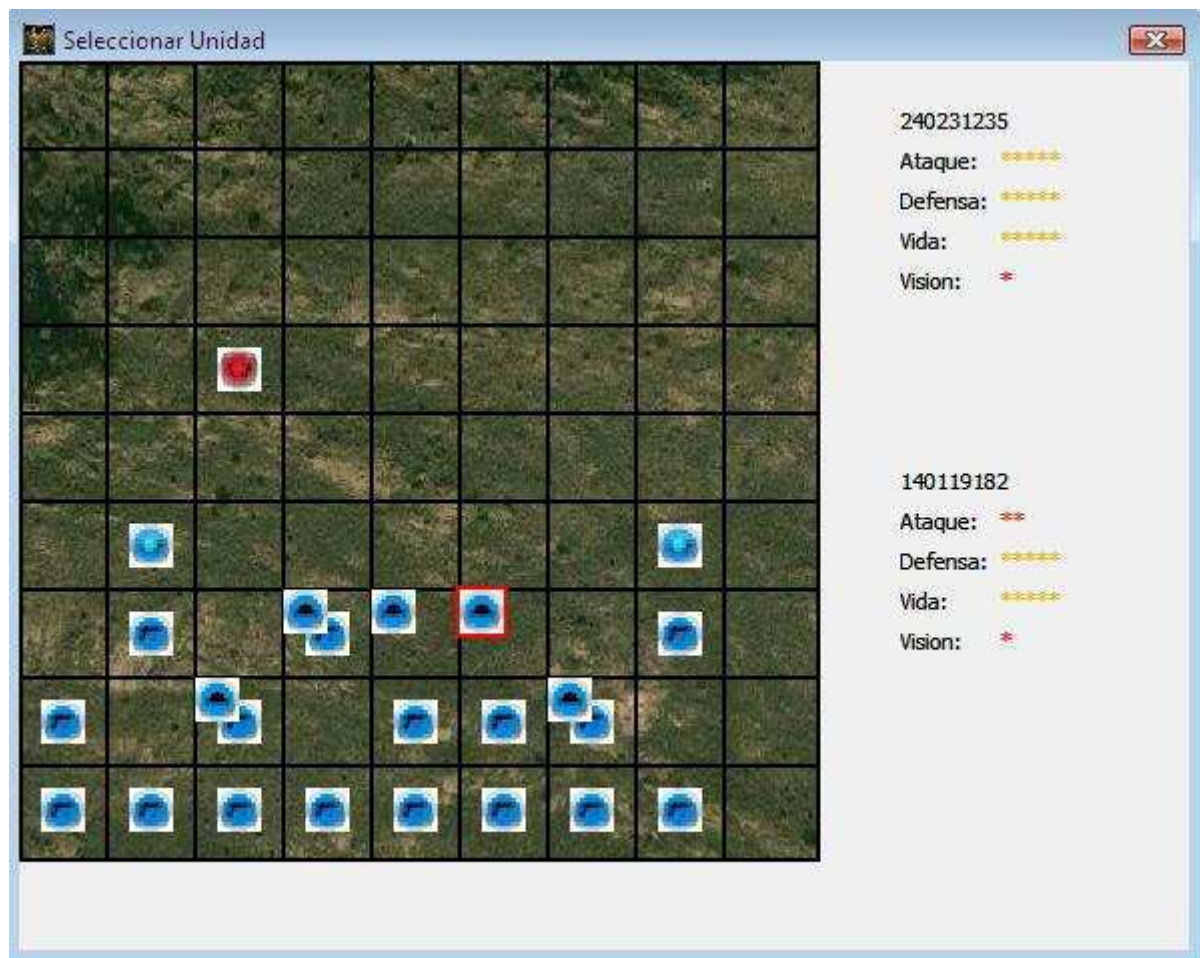


Figura 25. Pantalla de selección de enemigo al que atacar

Finalmente, destacar una funcionalidad capital. En cualquier momento de la simulación, podemos guardarla como si de un proyecto nuevo se tratara. Es decir, si pulsamos el botón "Guardar batalla" de la pantalla de Simulación, los mapas y los comentarios generados durante la simulación se guardarán en un archivo nuevo de tipo SimuBattle (extensión .sb). Así, cuando abramos dicho proyecto, la batalla aparecerá como si la hubiésemos editado nosotros mismos. Esto permite exportar cualquier simulación, completarla con nuevos comentarios, añadirle introducción y epílogo e incluso modificarla a nuestro antojo. Para entenderlo mejor, entendamos nuestro proyecto como una batalla editada y una simulada. Esta opción nos permite crear un nuevo proyecto donde la batalla editada es la simulada de otro anterior. De esta forma podemos exportar nuestros resultados de batalla simulada a otras personas para que éstas trabajen con ellos libremente, y generen nuevas simulaciones, realizando los cambios que consideren pertinentes.

Pliego IV

Presupuesto

Presupuesto

1. Costes por tiempo	203
2. Costes materiales.....	203
2.1. Software	203
2.2. Hardware	204
2.3. Otros	204
3. Presupuesto total.....	204

El presupuesto para la realización de este proyecto está constituido por los siguientes conceptos:

1. Costes por tiempo

Los costes de ejecución que se establecen en este proyecto como coste por tiempo se corresponden con las tablas salariales del Convenio Colectivo Nacional de empresas de ingeniería y oficinas de estudios técnicos y las últimas tablas para el año 2009 publicadas en el BOE por el Ministerio de Trabajo e Inmigración:

[<http://www.boe.es/boe/dias/2009/03/28/pdfs/BOE-A-2009-5211.pdf>]

⇒ Coste salarial anual para Nivel 1 (Licenciados y Titulados): 22937,51 €

⇒ Coste hora normal (asumiendo 1806 horas anuales): 12,70 €

Labor	Días totales	Horas/día	Horas totales	Importe
Estudios iniciales	50	4	200	2.540 €
Diseño de la aplicación	80	4	320	4.064 €
Implementación	250	4	1000	12.700 €
Documentación	70	4	280	3.556 €
TOTAL				22.860 €

Tabla 1. Desglose de costes de tiempo

2. Costes materiales

2.1. Software

⇒ **Software propietario.** Se incluyen en esta categoría el sistema operativo y las herramientas de desarrollo utilizadas.

Licencias	Coste total	Coste asociado
Windows XP Professional	300 €	30 €
Office Professional 2003	800 €	80 €
TOTAL		110 €

Tabla 2. Desglose de costes de software propietario

⇒ **Software de libre distribución.** En esta categoría se encuentran todas aquellas herramientas empleadas cuyo uso no conlleva ningún tipo de coste asociado. En este caso han sido Eclipse, Install4j y ObjectAid UML Explorer.

2.2. Hardware

En esta sección se incluyen los costes derivados del empleo de equipos.

Concepto	Coste total	Coste asociado
Equipo Pentium IV 2.8 GHz	2500 €	250 €
TOTAL		250 €

Tabla 3. Desglose de costes de hardware

2.3. Otros

En esta sección se incluyen todos aquellos costes materiales que no se pueden encuadrar en ninguno de los apartados anteriores.

Concepto	Coste total	Coste asociado
Conexión Internet ADSL	400 €	40 €
TOTAL		40 €

Tabla 4. Desglose de costes materiales encuadrados en "Otros"

3. Presupuesto total

Considerando todos los recursos humanos empleados contabilizados como costes en tiempo, así como los materiales empleados contabilizados como recursos materiales, el presupuesto necesario para la realización de este proyecto se corresponde con el valor mostrado en la Tabla 5.

Licencias	Importe total
Costes en tiempo	22.860 €
Costes materiales	400 €
TOTAL	23.260 €

Tabla 5. Presupuesto total