Universidad de las Ciencias Informáticas
Entornos Interactivos 3D Vertex, Facultad 5

Algoritmos de generación procedural para la creación de mapas en videojuegos 2D de navegador

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Danier Lorenzo Pérez

Tutores: Dr. Omar Correa Madrigal

Ing. Reinaldo Garcia Maturell

La Habana, 2016

Quien quiera construir torres altas deberá ahondar mucho en los fundamentos
Anton Bruckner

Dedicatoria

A mi familia por ser la motivación para intentar alcanzar cada una de mis metas.

Agradecimientos

Le agradezco a mi familia por estar siempre presente. A mis amigos que son mi segunda familia. A mis tutores que han sabido guiarme desde los inicios de la investigación. A mis profesores que han compartido sus conocimientos desinteresadamente, y a todos aquellos que de una forma u otra han permitido que me encuentre hoy en este lugar.

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Danier Lorenzo Pérez
Autor

Dr. Omar Correa Madrigal
Tutor

Ing. Reinaldo Garcia Maturell
Tutor

Resumen

Los videojuegos de navegador en la actualidad son cada vez más ambiciosos, contando con historias mejores elaboradas y una jugabilidad capaz de compararse con sus homólogos de consola. Este desarrollo ha sido impulsado, en parte por las posibilidades que brindan las actuales tecnologías y por las exigencias de los usuarios, que desean ser sorprendidos de forma constante. Pero la gran mayoría de estos juegos necesitan de un servidor web para ejecutarse, dependiendo de la velocidad de conexión para su correcta ejecución y este aspecto precisamente es uno de los principales problemas que enfrentan estas aplicaciones. Este trabajo propone la adaptación de un algoritmo de generación de laberintos para la obtención de diferentes mapas para videojuegos de navegador, de forma tal que dichos mapas de juego se generen en el cliente, con el objetivo de minimizar de esta forma el intercambio de archivos en la red y con esto mejorar el desempeño del videojuego. Se realizaron pruebas para validar los resultados obtenidos con el algoritmo propuesto, mostrando las potencialidades de aplicar la generación procedural a videojuegos de navegador.

Palabras clave: algoritmo, generación procedural, videojuego de navegador.

Índice general

Índice de figuras

Índice de tablas

Lista de algoritmos

Introducción

Los videojuegos en la web, también conocidos como videojuegos de navegador son una modalidad que se ha desarrollado en los últimos años con la evolución de los servicios de Internet y el creciente auge que ha tenido la red de redes en la vida de la personas (solamente en Cuba se ha visto un aumento del 14 % en los usuarios con acceso a internet en el 2015 con respecto al año 2011) (BANCO MUNDIAL, 2016). Estos videojuegos de navegador facilitan que el usuario no necesite realizar instalaciones extras para jugar; los archivos se ejecutan en el mismo navegador, ya que son independientes de la plataforma, basados exclusivamente en tecnologías usadas por el lado del cliente, siendo esta la gran diferencia con los videojuegos de consola o desktop.

Además, brindan la posibilidad de que el usuario se conecte de manera remota desde cualquier lugar y sin la necesidad de disponer de un dispositivo de altas prestaciones.

Cuba no ha quedado ajena a estos avances, en los últimos años ha potenciado el desarrollo de *software* y ha incursionado recientemente en el ámbito de la producción de sistemas de realidad virtual como videojuegos terapéuticos y entrenadores, siendo algunas de estas producciones los videojuegos de navegador. La Universidad de las Ciencias Informáticas (UCI) se ha convertido en centro esencial de este desarrollo y dentro de la misma el Centro de Entornos Interactivos 3D, VERTEX de la Facultad 5.

Aunque estos tipos de videojuegos ofrecen varias ventajas como se ha expresado anteriormente, hay elementos que empañan su desempeño. Estas limitaciones están dadas por las características de su propio funcionamiento. Para lograr la ejecución de un videojuego de navegador 2D desde un servidor es necesario:

- Realizar la descarga de todos los ficheros para la creación de sus mapas y personajes hacia el cliente que hace la petición.
- Su ejecución dependerá de la velocidad de conexión, condicionando el tiempo de respuesta del videojuego y limitando, de ser lenta, su ejecución en tiempo real.
- A medida que aumenta la cantidad de contenido utilizado por un videojuego, es mayor el espacio necesario para su almacenamiento en el servidor.

Teniendo en cuenta la situación antes descrita se plantea como **problema de la investigación**: ¿cómo minimizar el volumen de información en el servidor y el tiempo de respuesta para videojuegos de navegador en dos dimensiones?

Para dar solución al problema de la presente investigación se plantea el siguiente **objetivo**: Desarrollar varios algoritmos que permitan mediante la generación procedural de mapas, disminuir el tiempo de

respuesta y el tamaño de los archivos en el servidor para videojuegos 2D de navegador.

Se define como **objeto de estudio** la generación procedural de mapas, y como **campo de acción**: la generación procedural de mapas para videojuegos 2D de navegador.

Para dar cumplimiento al objetivo planteado se definen las siguientes **tareas de investigación:**.

- Elaboración del marco teórico de la investigación a través del estudio del estado del arte que existe sobre el tema.
- Identificación de los principales algoritmos de generación procedural de mapas para la creación de videojuegos.
- Caracterización de los videojuegos que utilicen la generación procedural de mapas existentes tanto a nivel nacional como internacional.
- Identificación y caracterización de las tecnologías y herramientas web que se utilizan para lograr la generación procedural de mapas para videojuegos 2D de navegador.
- Selección de las herramientas y tecnologías para la creación de una inaplicación web que implemente los algoritmos propuestos.
- Implementación de los algoritmos que brinden la solución al problema planteado.
- Realización de pruebas de rendimiento para validar la solución desarrollada.

Para el correcto desarrollo de la investigación, se emplean los siguientes métodos científicos:

Métodos teóricos

Método deductivo: se utilizó de forma discursiva y descendente pasando de lo general a lo particular, para llegar a conclusiones sobre la generación procedural partiendo de los trabajos previos sobre la misma.

Método analítico-sintético: se utilizó para estudiar la bibliografía, obteniendo de esta varias ideas separadas que luego serían unidas para conformar una síntesis, transitando de las causas a los efectos y de los principios a las conclusiones.

Análisis histórico-lógico: fue utilizado con el objetivo de interpretar el problema para realizar la investigación teniendo en cuenta que los problemas tienen un enfoque lógico: existe la relación causa-efecto y cuentan además con un enfoque histórico: tienen un pasado, presente y futuro.

Métodos empíricos

Observación: se emplea con el objetivo de observar la generación procedural de contenido utilizada en videojuegos. Este método proporciona la vía para realizar un registro visual de las características comunes en este tipo de técnica e identificar las que puedan formar parte de la solución.

Método de la experimentación: se utilizó con el objetivo probar formas de generar contenido proceduralmente en videojuegos para navegadores.

El presente trabajo de diploma está compuesto por tres capítulos en los cuales se agrupan la investigación realizada sobre el estado del arte para dar solución al problema de la investigación, la propuesta de solución y las pruebas que validan la efectividad de la solución planteada.

- **Capítulo 1 Fundamentación teórica**: en este capítulo se realizan las investigaciones concernientes a los videojuegos de navegador, sus principales características así como la forma de generar contenido en estos de forma automática. Se realizó un estudio del estado del arte de los principales algoritmos y su relación con este tipo de videojuego. También se definieron las tecnologías a utilizar y los motivos que dieron paso a esta selección.
- **Capítulo 2 Propuesta de Solución**: en esta segunda parte se plantea la solución propuesta al problema planteado, se muestran los algoritmos desarrollados a partir de modificaciones aplicadas al algoritmo escogido y los efectos que se pretenden lograr con estos. También se definió el funcionamiento de cada algoritmo apoyándose en imágenes y pseudocódigo.
- **Capítulo 3 Validación de la solución propuesta**: En este último capítulo se le aplican diferentes pruebas a una aplicación web que hace uso de los algoritmos desarrollados para generar varios tipos de mapas de videojuegos.

CAPÍTULO 1

Fundamentación Teórica

1.1. Generación Procedural de Contenido

La Generación Procedural de Contenido (GPC), a pesar de ser un campo todavía en investigación, no es una técnica reciente. Desde hace varias décadas se utilizan procedimientos automáticos con el fin de crear contenido. En el año 1983 Tim Martin y *MicroGraphicImage* crearon el videojuego para ordenadores *Atari 400* y *Atari 800*: Spelunker, el cual es uno de los pioneros en la utilización de Generación Procedural (TOZALGAMES, 2015). La GPC también es utilizada en varias áreas de investigación como la estética y creatividad computacional, así como en los sistemas de recomendación (YANNAKAKIS y TOGELIUS, 2011).

La GPC es usada para crear contenido automáticamente usando algoritmos predefinidos y esta se ha convertido en un foco importante de investigación en los últimos años. Esta técnica tiene la viabilidad para reducir esfuerzos manuales del diseño de cada parte del juego y el tiempo requerido para el desarrollo del mismo y aunque no puede crear un contenido coherente e interesante de juego automáticamente, da soporte a los diseñadores para desarrollar el contenido con costo y esfuerzo reducido. También puede ayudar a los jugadores a disfrutar del contenido dinámico dado que la utilización de GPC para la creación automática de historias de juego, provee un espacio de juego complejo y diverso (LEE y CHO, 2011).

Mediante la GPC se puede generar todo el contenido de un videojuego. Este contenido se refiere a todos los aspectos que afectan la jugabilidad pero no son comportamientos del personaje o el motor de juego. Esta definición incluye aspectos tales como terrenos, mapas, niveles, historias, diálogo, búsquedas, personajes, reglas de juego, perfiles de cámara, dinámica, música y armas. La definición explícitamente excluye el término personaje no jugable Non Player Character (NPC) que se controla mediante la utilización de inteligencia artificial en los videojuegos (YANNAKAKIS y TOGELIUS, 2011).

La tecnología del videojuego ha avanzado mucho desde sus inicios, cuando los gráficos eran casi inexistentes y se contaba con una interacción entre el jugador y el juego muy simple. Hoy día los entornos virtuales son elaboraciones realistas en 3 dimensiones, existe una dinámica enriquecedora y un aumento en el detalle gráfico, pero la creación de contenido en los videojuegos actuales sigue siendo grandemente manual (ibíd.).

La GPC es el método mediante el cual los creadores de videojuegos pueden satisfacer, con un contenido siempre variado, a los exigentes usuarios, ahorrándose en este proceso tiempo y recursos durante la producción.

1.2. Tipos de Generación Procedural de Contenido

Existen diferentes formas de generar contenido proceduralmente. En la presente sección se han expresado estos tipos de GPC en forma de confrontación "vs" (TOGELIUS; YANNAKAKIS; STANLEY y BROWNE, 2010).

En tiempo de ejecución vs en tiempo de creación: una distinción a tener en cuenta es si la generación de contenido se realiza durante la ejecución del juego o durante el desarrollo del mismo. Un ejemplo de lo anterior es cuando el jugador entra por la puerta de un edificio y el juego instantáneamente genera la parte interior del edificio, que no estuvo allí antes. En este último caso, un algoritmo sugiere trazados interiores que son entonces editados y perfeccionados por un diseñador humano antes de que el videojuego salga al mercado. Los casos intermedios son posibles, en donde un algoritmo sugiere nuevos mapas previamente creados para ser usados en esta habitación en la que el jugador acaba de entrar.

Contenido necesario vs opcional: otra distinción es si el contenido generado es necesario u optativo. El contenido necesario es requerido por los jugadores para progresar en el juego, por ejemplo las mazmorras que necesitan ser atravesadas, los monstruos que necesitan ser eliminados, las reglas cruciales del juego; mientras que el contenido optativo es ese que el jugador puede elegir evitar y aun así ganar el juego. La diferencia aquí es que el contenido necesario siempre necesita ser correcto; por ejemplo no es aceptable generar una mazmorra que no tenga salida si esto hace imposible que el jugador progrese. Por otra parte, es posible utilizar un algoritmo que a veces genere armas inutilizables y trazados irrazonables del piso, dado que esto no impide que el jugador cumpla con las metas del videojuego.

Semillas aleatorias vs vectores de parámetros: otra distinción concerniente al algoritmo de generación es hasta qué punto puede ser parametrizado. Todos los algoritmos que permiten la GPC crean contenido expandido de cierto tipo basado en una representación mucho más compacta. En un extremo, el algoritmo simplemente podría tomar una semilla como entrada a su generador de números aleatorios; en el otro, el algoritmo podría tomar como entrada un vector multidimensional de parámetros con valores reales que especifican las propiedades del contenido que genera.

Generación estocástica vs determinista: un valor a tener en cuenta en la generación procedural es la aleatoriedad en el contenido generado, dado que la variación en los resultados de las ejecuciones de algoritmos con idénticos valores de entrada repercute en el diseño del videojuego. Es posible imaginar algoritmos de generación deterministas que siempre producen el mismo contenido dado los mismos parámetros; sin considerar la semilla del generador de números aleatorios un parámetro; pero es bien sabido que muchos algoritmos no lo hacen.

Constructivo vs generación y prueba: Existen algoritmos que pueden llamarse constructivos y aquellos descritos como *generar y prueba*. Un algoritmo constructivo genera el contenido una vez y termina; sin embargo, se necesita estar seguro que el contenido que se está generando es correcto o al menos lo suficientemente bueno como para permitir al usuario concluir el videojuego. Un algoritmo de generación y prueba incorpora un algoritmo constructivo y un mecanismo de prueba. Después de que una instancia del contenido candidato es generado, es puesto a prueba según algunos criterios. Si la prueba fracasa, todo o una parte del contenido candidato es descartado y regenerado. Este proceso continúa hasta que el contenido sea lo suficientemente correcto para el desarrollo del videojuego.

1.3. Los videojuegos de navegador y la Generación de Contenido Procedural

En sus más de 30 años de vida, el videojuego como medio artístico y, sobre todo, lúdico, ha ampliado sus posibilidades hasta límites inicialmente insospechados. A través de esta evolución, el videojuego ha pasado por varios soportes tecnológicos en forma de ordenadores o videoconsolas que han mejorado cualitativamente las posibilidades estéticas de los videojuegos y que han permitido desarrollar nuevas dimensiones del mismo. A raíz de ser dotado de nuevas fronteras creativas en torno a lo puramente audiovisual, el juego ha desarrollado su propio lenguaje narrativo y ha absorbido otros inicialmente habituales en otras artes, llegando incluso a utilizar elementos provenientes del cine o de la literatura de forma evidente. El videojuego ha utilizado incluso las facilidades de conexión que brinda Internet mediante los navegadores web (DOMÍNGUEZ y SÁEZ, 2012).

Un juego de navegador o browser game es todo aquel que se juega mediante un navegador web (Internet Explorer, Firefox, Opera, etc.) y por ende independientemente de la plataforma. Funciona en cualquier equipo que tenga un navegador web y no es necesario instalar programas extras en el ordenador ni realizar descargas de contenido para poder jugar con ellos. Tan sólo es preciso ir a la página del juego en cuestión y en algunos casos: crear una cuenta de usuario. Dada la tecnología necesaria para su uso, los *browser games* existen prácticamente desde que se inventó la *World Wide Web* en 1993 y con ella el primer navegador web, aunque han ido evolucionando al mismo tiempo que Internet (POMBO, 2009). Mucho antes en 1979, un grupo de estudiantes de la Universidad de Essex crearon una versión informática del célebre juego de rol Dungeons & Dragons. Esta versión electrónica era multiusuario y estaba basada en el uso de textos alfanuméricos. Así surgió un nuevo tipo de juegos conocidos como MUD (Multi-User Dungeons o Domains) que se desarrollaría rápidamente por la aún poco conocida Internet, surgiendo así las primeras comunidades virtuales (LINKEDIN, 2016).

Es amplia la lista de videojuegos de navegador que han tenido grandes éxitos, algunos de estos son: **Sparta: War of Empires**, **Wonderputt**, **Super Hot**, **Diablo** este último famoso por usar la GPC en la creación de sus universos jugables (GAMEPEDIA, 2012).

La Generación de Contenido generado de forma automática también se ha estado utilizando en los navegadores web, dado el reciente uso de estos en la ejecución de videojuegos. **Spelunky** es una versión

modernizada de **Spelunker**, un juego de 1983, el cual hace un uso efectivo de la generación procedural de mapas, es uno de los primeros juegos que implementó el uso de la generación procedural. Cada parte de las cavernas en el juego es generada por un algoritmo de generación de terrenos (ZAHUMENSZKY, 2015). Spelunky cuenta entre sus plataformas la de navegador web y es uno de los pocos *browser game* que implementan este tipo de técnica. Son menos los casos de videojuegos de navegador que hacen uso de esta técnica automatizada, en comparación con sus homólogos desarrollados en plataformas de consolas o escritorio.

1.4. Tipo de mapas en videojuegos 2D

En un videojuego hay que diseñar escenarios que definan adecuadamente cómo es el mundo y cómo son los lugares que visitan los personajes. A menudo el mundo de un videojuego se diseña antes que sus niveles y misiones. El mapa tiene un papel protagónico en los juegos 2D, en él se desarrolla la historia que rige el juego. Para un diseñador de juegos, el terreno no es un simple fondo secundario, puede ejercer como factor de decisión para el jugador (BA, 2009).

Todo mapa se divide en unidades mínimas, un ejemplo de esto pudiera ser una celda. Estas celdas pueden servir para conformar el terreno de juego. Las celdas además pueden tener diferente formas, siendo las más comunes, la rectangular y la hexagonal, muy populares en juegos de tableros. Otro elemento que conforma un mapa es la vista o sea, el ángulo en que el usuario visualiza el mapa. Existen dos tipos de vistas básicas en los videojuegos 2D:

- *Top down*, conocida también como vista de águila, es la más fácil de desarrollar. Generalmente utiliza un arreglo de 2 dimensiones que se transforma en un mapa visto desde arriba.
- Vista Axonométrica, consiste en poner un ángulo a una vista *top down*. Es más difícil de desarrollar si no se cuenta con un motor gráfico de 3D, pero los gráficos tienden a ser más realistas que en la vista *top down* ya que le añade un valor de profundidad. Existen 3 tipos de vistas axonométricas, la isométrica, la trimétrica y la dimétrica, esta última es también conocida como vista tres cuartos (3/4). Las diferencias entre cada una de las vistas axonométricas son fáciles de apreciar. En la vista isométrica, sus 3 dimensiones tienen el mismo tamaño. En la dimétrica dos dimensiones tienen el mismo tamaño mientras que la tercera dimensión puede dibujarse más grande o más pequeña. La trimétrica trata básicamente de que las 3 dimensiones sean representadas con una longitud diferente.

1.5. Algoritmos de Generación Procedural de Contenido utilizados en videojuegos

Un algoritmo es un conjunto finito de instrucciones ejecutables, no ambiguas, que dirige una actividad que termina o resuelve un problema en tiempo limitado (JIMÉNEZ REY, 2005). Es la elección correcta de un algoritmo un paso decisivo a la hora de solucionar un problema determinado. En la generación de contenido de forma automática, el método o algoritmo que se utiliza para generar contenido influye directamente

en los elementos generados y su calidad. A continuación se presentan algoritmos usados para generar procedimientos en videojuegos.

Perlin Noise: ideado por Ken Perlin para generar las texturas de la película Tron (Disney, 1982) y ampliamente usado en efectos de fuego, humo o polvo en videojuegos. Se basa en la generación de varias funciones de ruido que son superpuestas posteriormente (sumadas) para la obtención del resultado final (PEDREÑO MOYA, 2014). Debido a que la función de Perlin Noise se ha convertido en una herramienta omnipresente en la síntesis de la textura, se ha optimizado para una gran variedad de plataformas de propósito general y especial. (HART, 2001). A pesar de ser muy usado este método para generar contenido en videojuegos de forma automática, generalmente elementos del paisaje en mapas, existe muy poca documentación que lo vincule con videojuegos de navegador. Los tipos de mapas generados por este tipo de algoritmo son generalmente terrenos, permitiendo muy poca variabilidad a la hora de querer crear un mapa de otro tipo.

Figura 1.1. Ejemplo de terreno generado con el algoritmo Perlin Noise.

Diamond-Square: también conocido como "fractal de plasma" por el efecto que suele crear, fue presentado por Alain Fournier, Don Fussell y Loren Carpenter en 1982. Pese a que posteriormente se demostrara que tiene una tendencia a crear líneas rectas (analizado por Gavin S.P. Miller en 1986), el uso de sus variantes es muy extendido a la hora de generar terrenos de aspecto realista, debido a su rapidez y a la facilidad de implementación. Este tipo de algoritmo se ha utilizado junto a Autómatas Celulares para la GPC de mapas de cavernas para videojuegos en 3D (JOHNSON; YANNAKAKIS y TOGELIUS, 2010). En las investigaciones realizadas hasta el momento no se ha encontrado información que vincule este algoritmo con los juegos de navegador. La generación que permite este algoritmo está enfocada principalmente a los terrenos 3D por lo que no es un buen candidato para el presente trabajo.

Autómatas Celulares: en la década de 1970, un matemático llamado John Conway publicó una descripción de El juego de la vida, a veces simplemente llamado Vida. Vida no era realmente un juego; era más como una simulación que tuvo una rejilla de celdas, cada celda contenía una célula que podía estar viva o muerta. A estas células se le aplicaban reglas simples en cada paso de la simulación:
Si una célula viva con menos de dos vecinos que viven, esta muere en la próxima generación.
Si una célula viva tiene dos o tres vecinos que viven, se mantiene con vida.

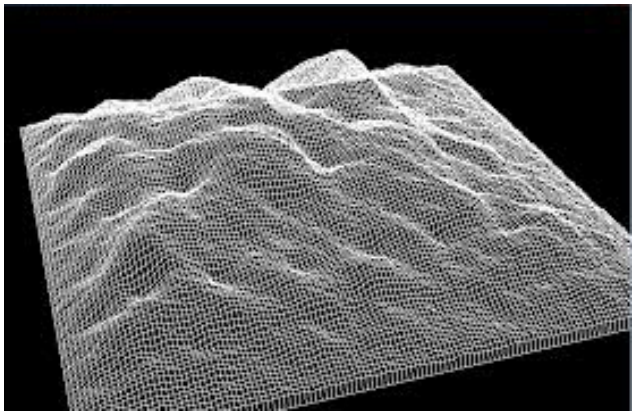

Figura 1.2. Imagen de un mapa en 3D generado mediante el algoritmo Diamond-Square.

Si una célula viva tiene más de tres vecinos que viven, esta muere en la próxima generación.
Si una célula muerta tiene exactamente tres vecinos que viven, se convierte en viva.

El juego de la vida es un ejemplo de un autómata celular, una red de células que se rigen por ciertas normas (COOK, 2013). Los autómatas celulares se han usado en varios juegos para generar terrenos, comportamientos de NPC y simular comportamientos de estructuras vivas que mimetizan la realidad. Se han encontrado referencias en la investigación de implementaciones de autómatas celulares para generar cuevas en videojuegos. Durante la investigación no se encontraron ejemplos de videojuegos de navegador que utilicen autómatas celulares para generar contenido (BUCK, 2012). Generar diferentes tipos de mapas haciendo uso de autómatas celulares, representaría una compleja definición de reglas que rijan el sistema.

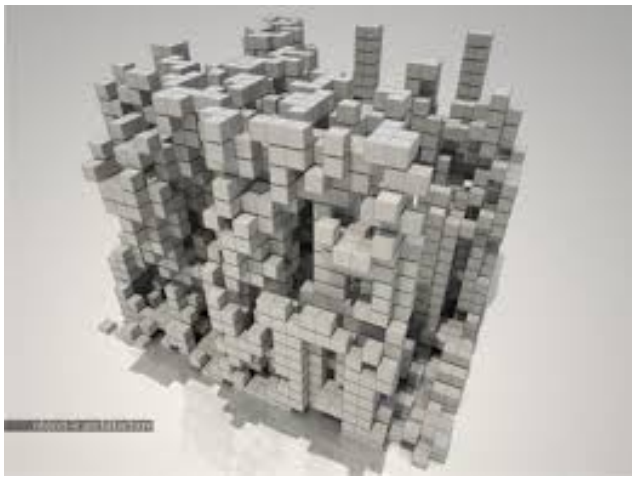

Figura 1.3. Imagen de un cubo en 3D generado a partir de autómatas celulares.

Algoritmos genéticos: a lo largo de los años, los algoritmos genéticos Genetic Algorithm (GA) han sido utilizados en una gran variedad de campos para encontrar soluciones a problemas de optimización. Este

tipo de algoritmo es encontrado en la resolución de problemas clásicos, como el problema de enrutamiento de vehículos o el problema del viajante de comercio. En el campo de la inteligencia artificial, los GA tratan de simular el comportamiento de la evolución natural de las especies. Para ello, parten de un conjunto de individuos lo más variado posible, denominado generalmente *población*. Cada individuo de la *población* representa una posible solución al problema propuesto. A su vez, cada individuo consiste en un conjunto de características o genes que pueden tomar diferentes valores. Los valores de estos genes constituyen el genotipo de la solución y su representación física el fenotipo. El objetivo del GA es evolucionar las mejores soluciones con el fin de encontrar una solución óptima al problema. Esta evolución se lleva a cabo mediante la aplicación de tres operadores: selección, cruzamiento y mutación (ARRIBAS, 2013). Los GA pueden utilizarse en la generación de mapas en videojuegos, con la correcta selección de una función de *aptitud* que es la que determina a qué tipo de solución converge el algoritmo, ya que es la que realiza la evaluación de los individuos de la población (ibíd.). A pesar de ser algoritmos utilizados en videojuegos, hasta el momento no se han encontrado ejemplos de la utilización de los GA en juegos de navegador. Para lograr esta sería necesario establecer otro algoritmo que evalúe la calidad del mapa generado a partir de la evolución de la población y determine si es correcto (si tiene solución) o no, teniendo que repetir el proceso en caso de no ser correcto el mapa. Los GA responden al tipo de algoritmo *de generación y prueba*.

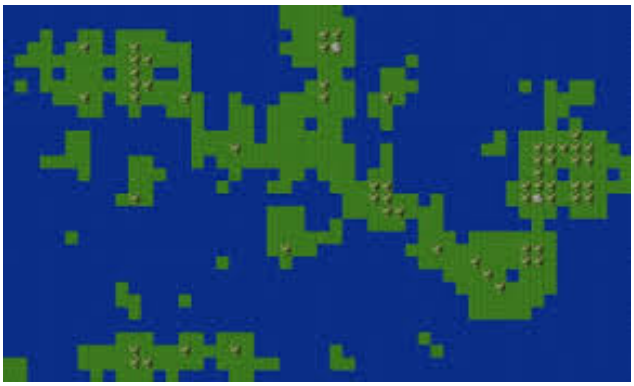

Figura 1.4. Mapa generado mediante Algoritmos Genéticos.

Algoritmos de generación de laberintos: desde los inicio del desarrollo de videojuegos los laberintos han sido protagonistas en sus historias. Un ejemplo de esto es Pac-Man, un videojuego arcade creado por el diseñador de videojuegos Toru Iwatani en los años 1980 cuyos mapas estaban formados por laberintos (FRUGONI, 2012). Existe un gran número de algoritmos de generación de laberintos que pueden adaptarse a las necesidades de un videojuego de dos dimensiones. A continuación se exponen algunos de estos algoritmos.

Backtracker Recursivo o RB por sus siglas en inglés: es rápido y fácil de implementar, para su funcionamiento realiza un recorrido no secuencial de una matriz. Es uno de los algoritmos que menos callejones sin salida genera, y como resultado tiene particularmente largos y sinuosos pasajes (BUCK, 2011). Si se tuviera como premisa del juego la llegada de un punto A a otro B en el mapa, este tipo de algoritmos se ajusta a la GPC de tipo Constructiva, ya que los pasajes están interconectados entre sí. Su desempeño en juegos de navegador todavía no está evaluado.

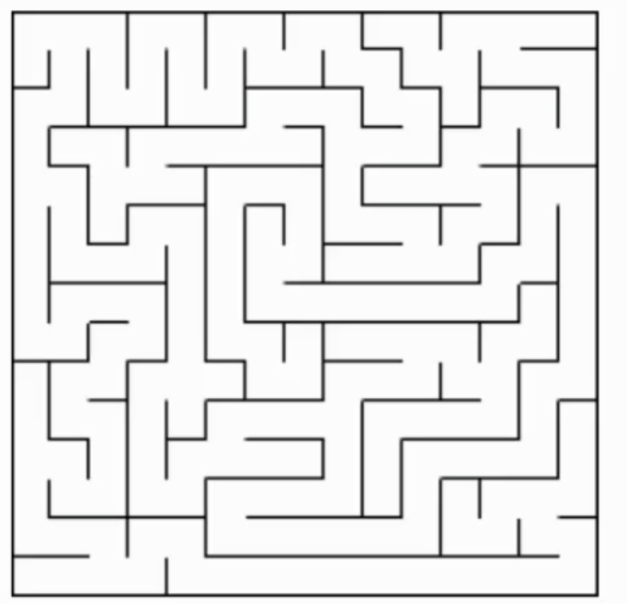

Figura 1.5. Laberinto generado mediante el algoritmo RB.

Algoritmo de Prim: es una variante del algoritmo de Árbol de Expansión Mínima que permite la selección de las aristas de manera aleatoria. Su enfoque hacia la creación del laberinto se toma desde un enfoque diferente al del RB, comienza en un punto y se expande hacia el exterior de manera simultánea. Una de las desventajas del algoritmo de Prim es que tiende a generar laberintos con muchos caminos cortos.

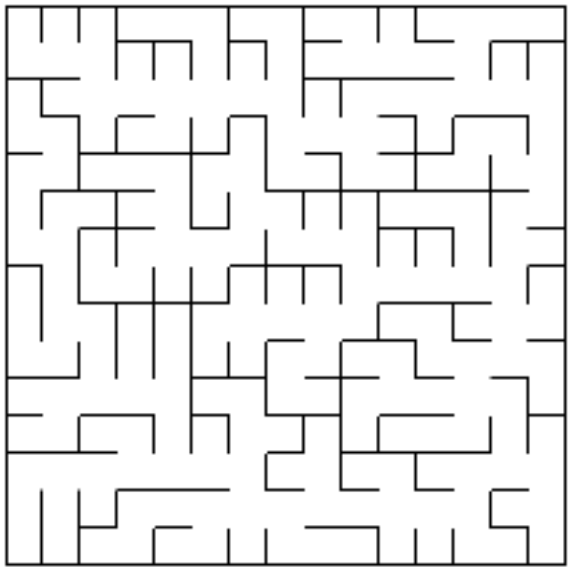

Figura 1.6. Laberinto generado mediante el algoritmo Prim.

Algoritmo de división recursiva: es un algoritmo de creación de laberintos diferente a los antes expues-

tos. Su diferencia se centra en que en vez de tallar pasaje partiendo de una celda, este inicia en un espacio vacío y añade paredes a este. Este algoritmo tiende a generar una gran cantidad de caminos sin salidas (BUCK, 2011).

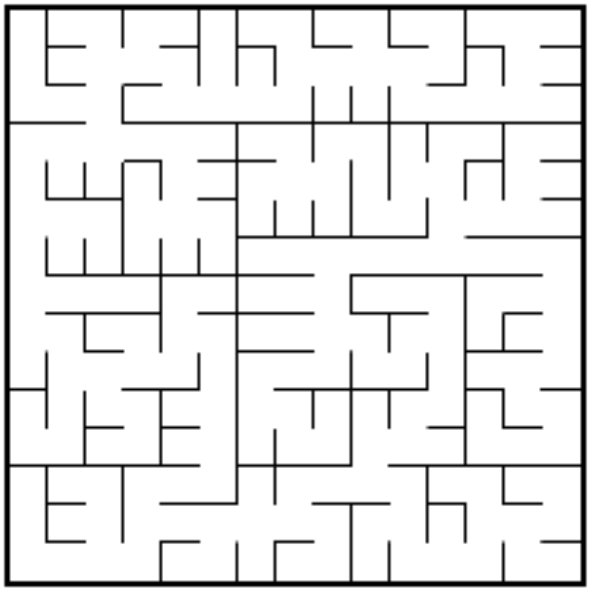

Figura 1.7. Laberinto generado mediante el algoritmo División Recursiva.

1.5.1. Algoritmo seleccionado para la generación de contenido

De los algoritmos antes expuestos es necesario seleccionar uno que permita la generación de contenido en videojuegos 2D para navegador. Es necesario que dicho algoritmo pueda modificarse y que permita generar varios mapas de juegos en la aplicación. El algoritmo escogido para realizar este proceso es **Backtracker Recursivo**, dado que para su funcionamiento utiliza un recorrido no secuencial de una matriz y esta cualidad puede ser utilizada favorablemente para lograr variedad en los mapas generados. También se analizó su facilidad de implementación y la rapidez para generar largos y sinuosos pasajes en los laberintos. Estas condiciones son favorables para, mediante modificaciones, lograr obtener otros mapas de juegos.

1.6. Herramientas y tecnologías

A continuación se hace una evaluación de bibliotecas para el desarrollo de videojuegos de navegador y se expresan ventajas y desventajas de estas. Al final se explica cuál será la escogida para realizar el presente trabajo.

Impact.js es una de las bibliotecas más utilizadas para crear videojuegos usando *JavaScript*. Maneja *sprites* (animaciones), mapas con patrones (*tiles*), colisiones, sonidos y cuenta con un conveniente editor de

niveles llamado *Weltmeister*. Además cuenta con un sistema de *plugins* con el que se puede extender aún más las funcionalidades. Impact.js permite crear juegos que pueden ejecutarse en cualquier navegador con soporte para HTML5 como Firefox, Chrome, Safari, Opera e Internet Explorer 9. Trabaja bajo un esquema de código cerrado y la licencia tiene un costo de US $ 99 (ANDREARRS, 2014).

Phaser es uno de los motores de juegos para JavaScript más recientes. Utiliza *Pixi.js* para el rendering. Está diseñado para que los juegos se puedan ejecutar tanto en ordenadores como en dispositivos móviles, siendo este último su principal objetivo. Phaser soporta Canvas y WebGL y permite cambiar de uno a otro automáticamente según la compatibilidad del navegador. Esta es una de las ventajas que posee Phaser en cuanto a *rendering* y a velocidad de respuesta contribuyendo a una mejor experiencia de usuario. También gestiona física, colisiones, animaciones, sistema de partículas, mapas de patrones, sonidos y permite escalar el juego para que se ajuste a la resolución de cualquier dispositivo sin alterar la relación de aspecto. Al igual que Impact.js maneja un sistema de plugins que permite extender las funcionalidades del motor. Phaser es Open Source (PHASER, 2016).

Kiwi.js viene con soporte para animaciones, sprites, cámaras, sonidos, texturas y un módulo muy útil para crear interfaces de usuario. Sin embargo carece de motor de física y colisiones (KIWIJS, 2013).

Game.js permite manejar *sprites*, animaciones, sonidos, mapas de patrones y colisiones, contiene métodos muy útiles para búsqueda de rutas (necesario para la inteligencia artificial) y comunicaciones vía HTTP. Es *Open Source* (ANDREARRS, 2014).

MelonJS es una biblioteca *Open Source* en desarrollo que es compatible con Chrome, Safari, Firefox, Opera e incluso Internet Explorer en versiones igual o superiores a la 9. Esta posee mecanismos básicos de física y colisión para asegurar bajos requerimientos de CPU. MelonJS integra el popular formato de mapas tiled, lo que permite diseñar fácilmente niveles usando el editor de mapas. Este motor 2D basado en *sprites*, también cuenta con soporte para múltiples canales de audio, matemática básica de vectores y efectos de transición, entre otras características (MELONJS, 2016).

1.6.1. Biblioteca a utilizar

La biblioteca web escogida para el desarrollo de una aplicación que vincule los algoritmos a la generación de mapas es Phaser.js por ser *Open Source* (de código abierto), contar con tres motores de física que abarcan una amplia gama de efectos necesarios en videojuegos. Además gestiona física, colisiones, animaciones, sistema de partículas, mapas de patrones, sonidos y permite escalar el juego para que se ajuste a la resolución de cualquier dispositivo sin alterar la relación de aspecto. También soporta *Canvas* y *WebGL* y puede utilizar uno u otro según la compatibilidad del navegador mejorando de esta forma el *rendering* y la velocidad de respuesta. También es válido destacar que cuenta con abundante documentación y tutoriales que facilitan y agilizan el proceso de desarrollo.

1.7. Conclusiones del Capítulo

En este capítulo se presentaron conceptos fundamentales a la hora de comprender qué es la generación de contenido y cómo funciona. A partir de la investigación realizada sobre los diferentes algoritmos que permiten la GPC en videojuegos, se logró determinar cuál es el algoritmo a utilizar, para mediante modificaciones lograr generar varios tipos de mapas. Se seleccionó *Phaser* como motor para la creación de videojuegos. Este se utilizará para implementar una aplicación web que muestre la generación de varios tipos de mapas a partir de los algoritmos propuestos. Teniendo en cuanta el algoritmo escogido se puede clasificar la GPC que se logrará en:

- *Realizada en tiempo de ejecución*, dado que los mapas serán generados en el cliente durante la ejecución de la aplicación.
- *De semilla aleatoria*, teniendo en cuenta que el contenido será generado a partir de valores aleatorios.
- Estocástica, porque no es posible determinar *a priori* cual será la salida exacta del algoritmo.
- Constructiva, debido a que el contenido generado por el algoritmo escogido siempre es correcto (los laberintos siempre tienen solución), por lo que la generación se realiza una sola vez para cada mapa.

Teniendo en cuenta que el algoritmo RB ofrece como salida una matriz bidimensional, los diferentes mapas que serán generados serán apreciados desde una vista: *top down*. Dado que es el tipo de vista, para videojuegos de dos dimensiones, de más fácil uso y utiliza arreglos bidimensionales para la confección de los mapas.

CAPÍTULO 2

Propuesta de solución

2.1. Propuesta de solución

A partir de las necesidades anteriormente analizadas y de las herramientas escogidas, se desglosarán los pasos del algoritmo escogido para dar solución al problema planteado. Este algoritmo permite la generación de los laberintos que servirán como mapas de juego y a partir de modificaciones que le serán realizadas, se podrán obtener también otros mapas para videojuegos de navegador.

2.1.1. Algoritmo Backtracker Recursivo

A continuación, se describen los pasos del algoritmo utilizado y posteriormente se detallan las modificaciones realizadas a este para generar los mapas del videojuego en el presente trabajo.
1 Escoger un punto de manera aleatoria en una matriz de dimensión N x M (ver figura 2.1).
2 Elegir de manera aleatoria una celda adyacente, (Norte, Sur, Este, Oeste) y tallar una pared (un paso) a través de esta, pero sólo si la celda adyacente no ha sido visitada aún. Esta nueva celda se convierte en la celda actual (ver figura 2.3).
3 Si se han visitado todas las celdas adyacentes, dar vuelta atrás hasta la última celda que tiene paredes sin tallar y repetir el paso anterior (ver figura 2.4).
4 El algoritmo termina cuando hayan sido visitadas todas las celdas de la matriz (ver figura 2.6).

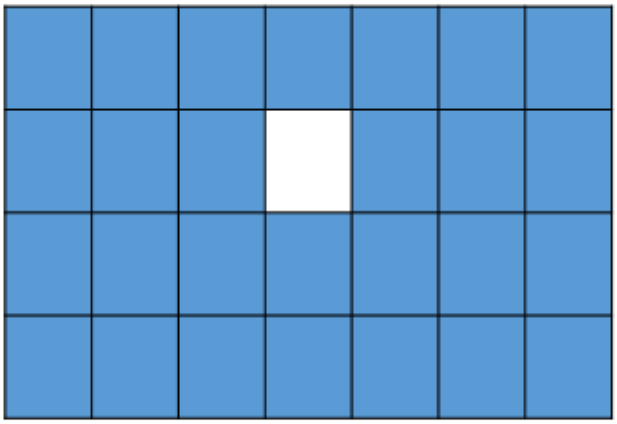

Figura 2.1. Escoger un punto de manera aleatoria en una matriz de dimensión NxM.

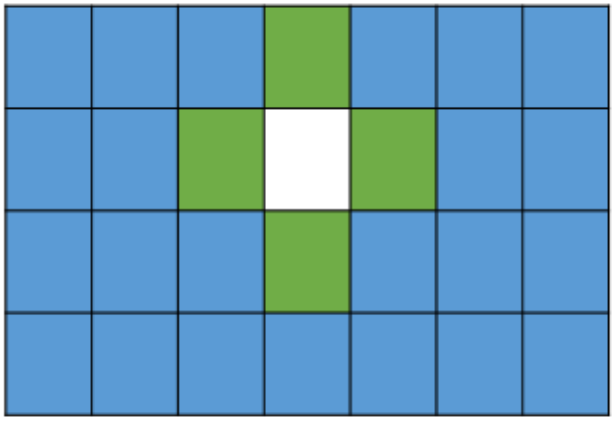

Figura 2.2. Elegir al azar una pared de las que conforman esa celda.

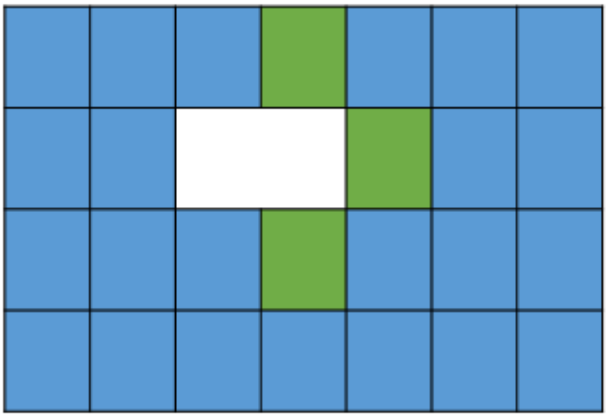

Figura 2.3. Tallar un paso a través de la celda adyacente, si la celda adyacente no ha sido visitada.

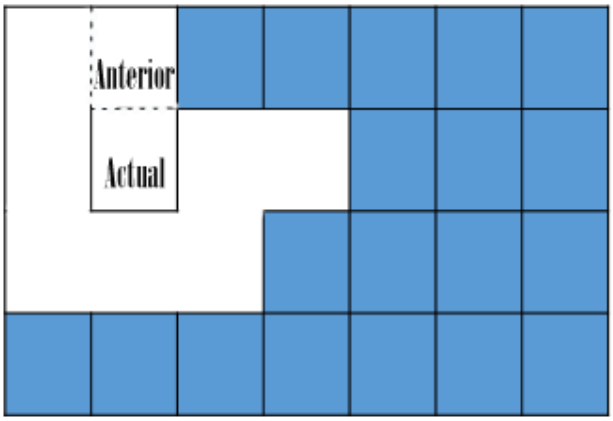

Figura 2.4. Si se han visitado todas las celdas adyacentes, dar vuelta hasta la última celda que tiene paredes sin tallar.

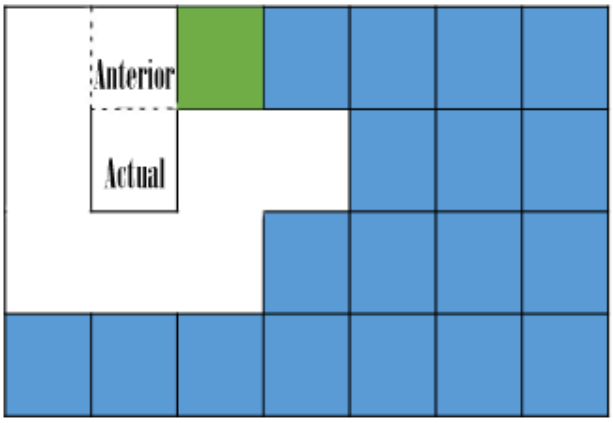

Figura 2.5. Repetir el paso 2.

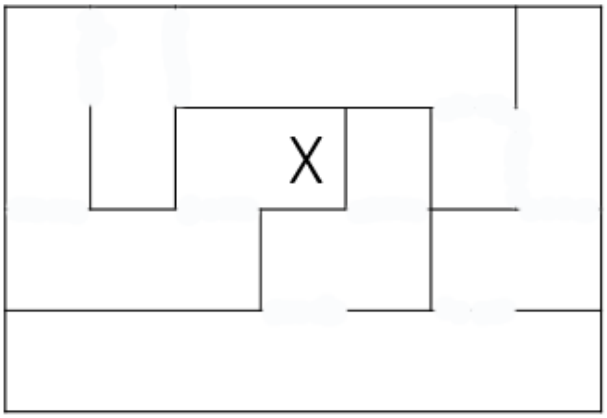

Figura 2.6. El algoritmo termina cuando hayan sido visitadas todas las celdas de la matriz.

El algoritmo retorna una matriz de 3 dimensiones, en la que cada celda contendrá un arreglo de 4 elementos binarios de la siguiente forma: [0, 0, 0, 0], representando **1** que existe relación y **0** que no existe relación con la celda continua en las direcciones Norte (primer elemento del arreglo), Este (segundo elemento del arreglo), Sur (tercer elemento del arreglo) y Oeste (último elemento del arreglo). Con esta matriz de resultados se podrá posteriormente conformar el laberinto.

A continuación se presenta el pseudocódigo del algoritmo backtracker recursivo. En el mismo se utiliza la estructura de datos Pila contenida en la variable **camino** para simular la recursividad en el recorrido de la matriz.

2.1.2. Pseudocódigo del algoritmo Backtracker Recursivo

Algoritmo 1 Backtracker Recursivo

```
1:  procedure NewMaze(X, Y)
2:      totalCeldas ← X *Y
3:      celdas ← Arreglo()
4:      sinVisitar ← Arreglo()
5:      for i ← 0, i < Y do
6:          celdas[i] ← Arreglo()
7:          sinVisitar[i] ← Arreglo()
8:          for j ← 0, j < X do
9:              celdas[i][j] ← [0, 0, 0, 0]
10:             sinVisitar[i][j] ← verdadero
11:         end for
12:     end for
13:     celdaActual ← [Ramdom(X), Random(Y)]
14:     camino ← celdaActual
15:     sinVisitar[celdaActual[0]][celdaActual[1]] ← falso
16:     visitados ← 1
17:     repeat
18:         posiblesVecinos ← celdas_adyacentes
19:         vecinos ← Arreglo()
20:         for i ← 0, i < 4 do
21:             if posiblesVecinos[i][0] < limitesyposiblesVecinos == sinVisitar then
22:                 vecinos.Push(posiblesVecinos[i])
23:             end if
24:         end for
25:         if vecinos.length() > 0 then
26:             proximaCelda ← Ramdom(vecinos)
27:             celda[celdaActual[0]][celdaActual[1]] ← 1
28:             celda[proximaCelda[0]][proximaCelda[1]] ← 1
29:             sinVisitar[proximaCelda[0]][proximCelda[1]] ← falso
30:             visitados + +
31:             celdaActual ← [[proximaCelda[0]], [proximaCelda[1]]]
32:             camino.push(celdaActual)
33:         end if
34:         if no_existen_vecinos_disponibles then
35:             celdaActual ← camino.pop()
36:         end if
37:     until visitados < totalCeldas
38:     devolver(celdas)
39: end procedure
```

2.2. Modificaciones aplicadas al algoritmo Backtracker recursivo para la obtención de mapas.

2.2.1. Mapas de mar

Se modificó el algoritmo escogido para generar islas rodeadas por mar. Este tipo de mapa es muy utilizado en videojuegos de navegador como es el caso de Lagoonia, un videojuego de estrategia que se desarrolla utilizando mapas de islas rodeadas por mar (BERMÚDEZ, 2012). Las principales modificaciones realizadas al algoritmo RB para la obtención de mapas de mar consiste en que cada celda contendrá un arreglo de 2 elementos, el primero contendrá **0** o **1** dependiendo de si se trata de una porción de isla o de mar, el segundo elemento del arreglo será modificado una vez se ejecute el algoritmo y contendrá información de la arena que rodea las islas. La distribución de contenido se hará siguiendo un factor determinado durante la primera mitad del recorrido de la matriz, una vez se hayan visitado la mitad de las celdas se sustituirá el modo de distribución de islas por reglas aplicadas a cada celda visitada. Cada celda contiene la información referente a dicha posición en pantalla atendiendo a las coordenadas **X** y **Y** en las que se encontrarán valores de **1** en caso de tratarse de una isla y **0** en caso de tratarse de mar, un conjunto de **1**s adyacentes en la matriz supondrá una isla proporcional en tamaño a la cantidad de **1**s.

A continuación se describen las modificaciones realizadas al algoritmo RB para generar mapas de islas para videojuegos 2D de navegador.

1 Escoger una celda de manera aleatoria en una matriz de dimensión NxM (figura 2.7).

2 Mientras la cantidad de celdas recorridas sea menor a la cantidad de celdas de la matriz y la celda no esté visitada, se asignará **0** o **1** siguiendo un factor de distribución F (luego de F celdas puestas a 0 se pondrá una a 1) el cual determina la densidad de islas que contendrá el mapa, pasando a la celda adyacente si esta no ha sido visitada (Ver Figura 2.8).

3 Al haberse recorrido la mitad de la matriz con este método se eliminará la distribución utilizada y se asignarán reglas al llenado de cada celda que se recorra. Estas reglas definen lo siguiente: la celda actual será puesta a **1** si adyacente a esta se encuentra alguna celda anteriormente recorrida que contenga un valor **1**, de lo contrario será puesta a 0. Dichas reglas permiten crear islas nuevas solo donde existan islas adyacentes, permitiendo de esta forma crear islas más grandes formadas a partir de la unión de islas pequeñas. Luego se visita la siguiente celda adyacente si esta no ha sido visitada (Ver Figura 2.9).

4 Si se han visitado todas las celdas adyacentes, dar vuelta hasta la última celda que contenga celdas adyacentes sin visitar y repetir el paso **2** y **3**.

5 El algoritmo termina cuando hayan sido visitadas todas las celdas de la matriz (Ver Figura 2.10).

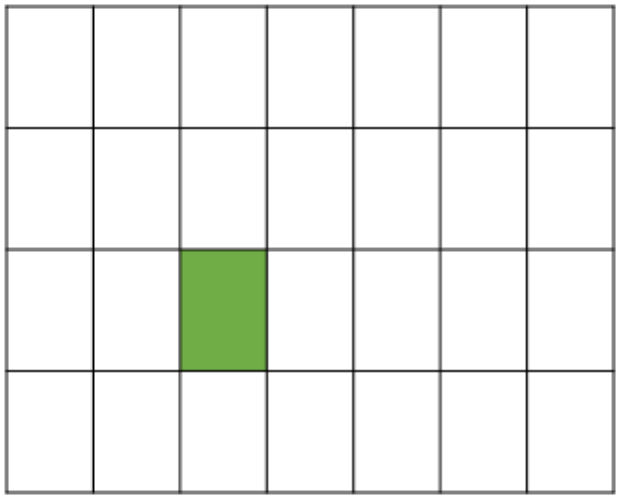

Figura 2.7. Escoger un punto de manera aleatoria en una matriz de dimensión NxM.

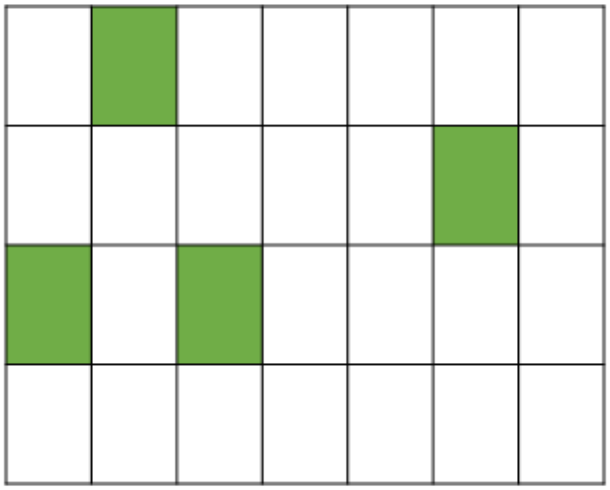

Figura 2.8. Se asignará **0** o **1** siguiendo una distribución determinada para cada celda que se visite, pasando a la celda adyacente si esta no ha sido visitada.

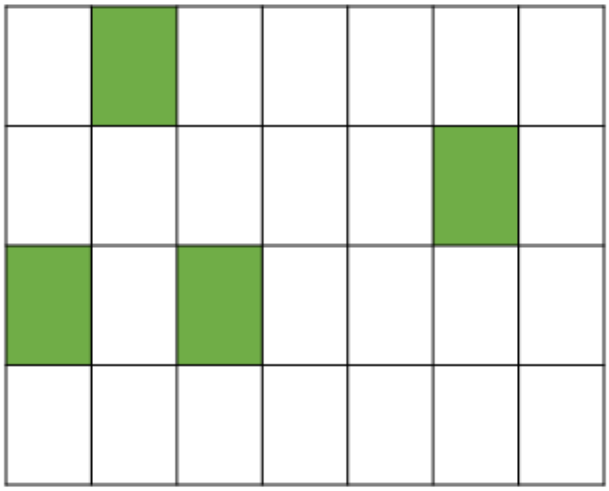

Figura 2.9. Al haberse recorrido la mitad de la matriz con este método, se eliminará la distribución utilizada y se asignarán reglas al llenado de cada celda que se recorra.

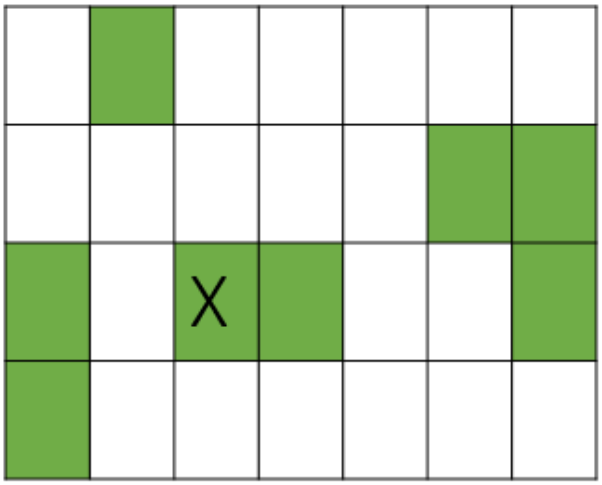

Figura 2.10. El algoritmo termina cuando hayan sido visitadas todas las celdas de la matriz.

A continuación se presenta el pseudocódigo del algoritmo para generar mapas de mar, en el mismo se utiliza la estructura de datos *Pila* contenida en la variable **camino**, para simular la recursividad en el recorrido de la matriz. También se utiliza la variable F que regula la densidad de islas distribuidas, permitiendo que cada F celdas de mar una sea de isla.

Pseudocódigo del algoritmo para generar mapas de mar

Algoritmo 2 Mapas de mar

```
1:  procedure NewMap(X, Y)
2:      celdas[i][j] ← [0, 0]
3:      Inicializar resto de variables
4:      repeat
5:          posiblesVecinos ← vecinos_adyacentes
6:          vecinos ← Arreglo()
7:          vecinosVisitados ← Arreglo()
8:          for i ← 0, i < 4 do
9:              if posiblesVecinos[i][0] > limiteMenoryposiblesVecinos[i][0] < limiteMayor then
10:                 if posiblesVecinos == sinVisitar then
11:                     vecinos.Push(posiblesVecinos[i])
12:                 end if
13:                 if posiblesVecinos == Visitados then
14:                     vecinosVisitados.Push(posiblesVecinos[i])
15:                 end if
16:             end if
17:         end for
18:         if vecinos.length() > 0 then
19:             proximaCelda ← Random(vecinos)
20:         end if
21:         if visitados == Fy < totalCeldas/2 then
22:             celdas[celdaActual[0], celdaActual[1][0]] ← 1
23:         end if
24:         if visitados > totalCeldas/2yvecinosVisitados.length() then
25:             mientrasvecinosVisitados.length() > 0
26:             l ← 0
27:             if vecinosVisitados[l] == 1 then
28:                 celdas[celdaActual[0]][celdaActual[1]][0] ← 1
29:                 l + +
30:             end if
31:             FinMientras
32:         end if
33:         sinVisitar[proximaCelda[0]][proximCelda[1]] ← falso
34:         visitados + +
35:         celdaActual ← [[proximaCelda[0]], [proximaCelda[1]]]
36:         camino.push(celdaActual)
37:         if vecinos.length == 0 then
38:             celdaActual ← camino.pop()
39:         end if
40:     until visitados < totalCeldas
41:     devolver(celdas)
42: end procedure
```

2.2.2. Mapas de asteroides

Para posibilitar la generación de mapas de asteroides, se modificó el algoritmo RB. El nuevo algoritmo retorna una matriz donde la entrada [i][j] contiene el valor **1** si la casilla contiene un asteroide o **0** en caso contrario. A continuación se describen los pasos del algoritmo RB modificado:

1 Escoger un punto de manera aleatoria en una matriz de dimensión NxM (Ver Figura 2.11).

2 Mientras la cantidad de celdas recorridas sea menor a la cantidad de celdas de la matriz se le asignará **0** o **1** a cada una de las celdas siguiendo un factor de distribución F (luego de F celdas puestas a 0 se pondrá una a 1), pasando luego a la celda adyacente si esta no está visitada (Ver Figura 2.12).

3 Si se han visitado todas las celda adyacentes, dar vuelta hasta la última celda que contenga celdas adyacentes sin visitar y repetir el paso **2**.

4 El algoritmo termina cuando hayan sido visitadas todas las celdas de la matriz (Ver Figura 2.13).

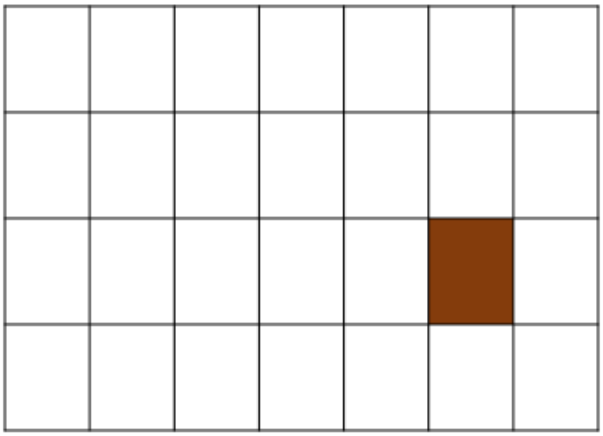

Figura 2.11. Escoger un punto de manera aleatoria en una matriz de dimensión NxM.

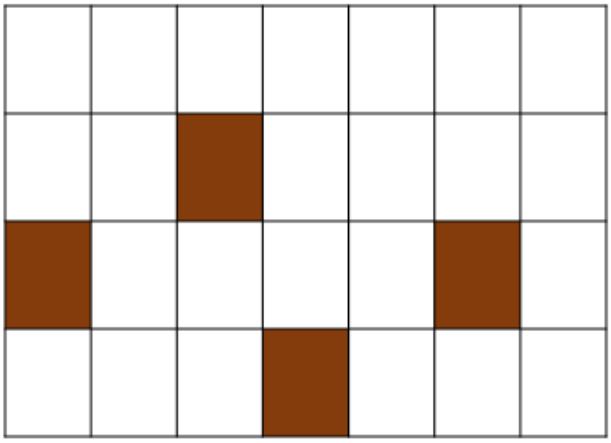

Figura 2.12. Se le asignará **0** o **1** a la celda siguiendo una distribución determinada, pasando luego a la celda adyacente si esta no está visitada.

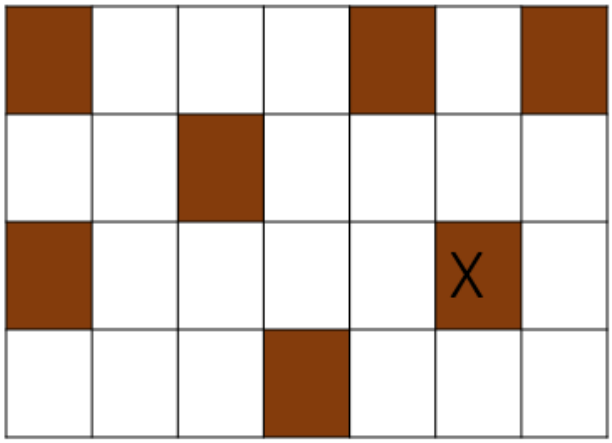

Figura 2.13. El algoritmo termina cuando hayan sido visitadas todas las celdas de la matriz.

A continuación se presenta el pseudocódigo del algoritmo que permite generar mapas de asteroides, en el mismo se utiliza la estructura de datos *Pila* contenida en la variable **camino** para simular la recursividad en el recorrido de la matriz. También se utiliza la variable F que regula la densidad de asteroides distribuidos, permitiendo que cada F celdas de espacio una sea de asteroide.

Pseudocódigo del algoritmo para generar mapas de asteroides

Algoritmo 3 Mapas de asteroides

```
1:  procedure NewMap(X, Y)
2:      celdas[i][j] ← [0]
3:      Inicializar resto de variables
4:      repeat
5:          posiblesVecinos ← vecinos_adyacentes
6:          vecinos ← Arreglo()
7:          for i ← 0, i < 4 do
8:              if posiblesVecinos ← en_limites y posiblesVecinos == sinVisitar then
9:                  vecinos.Push(posiblesVecinos[i])
10:             end if
11:         end for
12:         if vecinos.length() > 0 then
13:             proximaCelda ← Random(vecinos)
14:             if visitados == F then
15:                 celda[celdaActual[0]][celdaActual[1]] ← 1
16:             end if
17:             celda[proximaCelda[0]][proximaCelda[1]][proximaCelda[3]
18:             sinVisitar[proximaCelda[0]][proximCelda[1]] ← falso
19:             visitados + +
20:             celdaActual ← [[proximaCelda[0]], [proximaCelda[1]]]
21:             camino.push(celdaActual)
22:         end if
23:         if vecinos.length() == 0 then
24:             celdaActual ← camino.pop()
25:         end if
26:     until visitados < totalCeldas
27:     devolver(celdas)
28: end procedure
```

2.2.3. Mapas de ruinas

Este tipo de mapa es muy utilizado en videojuegos multijugador en los que varios jugadores luchan entre sí, o de manera cooperativa, controlando sus personajes desde una tercera persona. Para lograr la generación de los mapas de ruinas, no fue necesario modificar el algoritmo RB solo se modificó la forma en que se muestra en pantalla el contenido generado por dicho algoritmo. No es necesario mostrar completamente los laberintos sino que solo se mostrarán partes de estos. Se escogerán áreas de la pantalla aleatoriamente, para mostrar el laberinto generado y en el resto de la pantalla se ubicarán imágenes de vegetación.

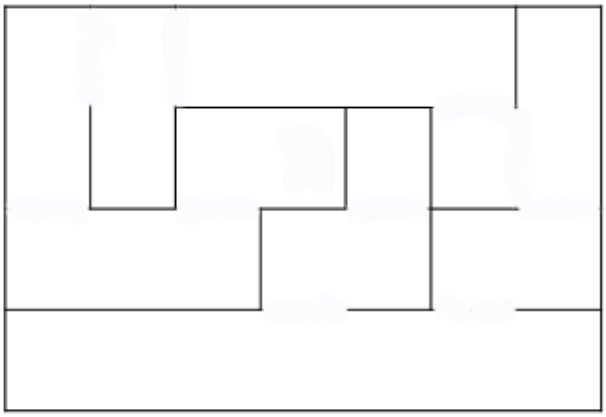

Figura 2.14. Laberinto generado a partir de la salida del algoritmo RB.

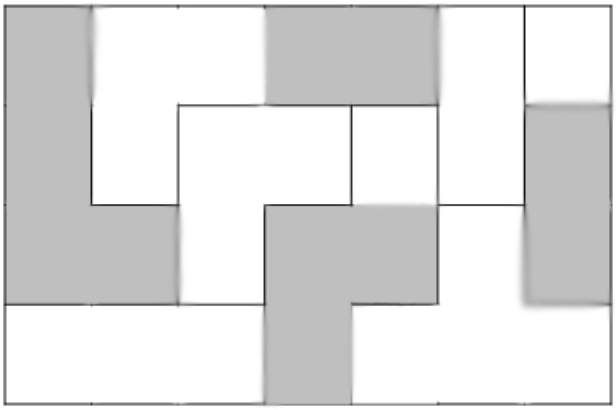

Figura 2.15. Se muestran solo partes del laberinto.

2.3. Conclusiones del capítulo

En este capítulo se proponen cuatro algoritmos para generar mapas para videojuegos 2D de navegador. Los algoritmos propuestos tienen como base el recorrido no lineal del algoritmo RB con algunas modificaciones realizadas para generar diferentes tipos de mapas como los mapas de mar, los mapas de asteroides, los mapas de ruinas y los mapas de laberintos. Los algoritmos propuestos tienen complejidad lineal en el tamaño del mapa que se desea generar, por lo que resulta eficiente su utilización para la generación de mapas para videojuegos 2D de navegador en tiempo real.

CAPÍTULO 3

Validación de la solución propuesta

3.1. Introducción al capítulo

Para poder medir las capacidades que ofrecen los algoritmos desarrollados en la generación procedural de mapas para videojuegos 2D de navegador, se ha creado una aplicación web que permite generar 4 tipos de mapas correspondientes a los algoritmos propuestos. Es posible generar mediante la aplicación un gran numero de mapas de cada tipo. En el presente capítulo se realizarán pruebas a dicha aplicación para medir los tiempos de respuesta y el tamaño de los archivos enviados desde el servidor hacia el cliente para la generación de mapas. Dichas variables serán comparadas con los datos arrojados por los mapas diseñados de forma manual. Para esto se creó una variante de la aplicación que no utiliza la GPC, para poder comparar los tiempos de respuesta y el tráfico de paquetes en la red. Las pruebas serán divididas en 3 etapas y contienen distintos escenarios. En cada escenario varían los niveles de juego generados, el tamaño de los mapas y el tipo de navegador utilizado (Firefox o Chrome).

Las 3 etapas en las que serán divididas las pruebas son las siguientes:

- Etapa 1, pruebas de carga realizadas a cada mapa en un entorno local. En los diferentes casos de pruebas que contendrá esta etapa variará el navegador web utilizado (Chrome o Firefox), el escenario de prueba (para varios niveles o para varias dimensiones de mapas) y el aspecto a medir (tiempo de respuesta o total de archivos descargados)
- Etapa 2, pruebas de carga realizadas a la aplicación en un entorno local, donde el acceso a esta será de forma concurrente. En esta etapa se medirá el tiempo de respuesta.
- Etapa 3, pruebas de carga realizadas a la aplicación en un entorno real, donde el demo creado estará disponible en un servidor de la universidad. En esta etapa de prueba se medirá el tiempo de respuesta y el total de archivos descargados desde el servidor.

Figura 3.1. Vista del Demo que permite la generación procedural de mapas.

3.1.1. Pruebas de Carga

El objetivo de las pruebas de carga es verificar el tiempo de respuesta del sistema para transacciones, bajo diferentes condiciones de carga. En el presente trabajo este tipo de pruebas servirá para demostrar las ventajas de los mapas generados de forma automática en comparación con otros diseñados de forma manual a la hora de la transferencia de estos datos hacia la máquina cliente.

Para la presente sección de pruebas se simularán varios niveles de juego con los diferentes algoritmos de generación de mapas utilizados en la aplicación y se compararán con los tiempos de cargas al realizar la misma operación con mapas de diseño manual. Se utilizarán las herramientas para desarrolladores contenidas en el navegador Chrome, especialmente la encargada de analizar el tráfico de red.

3.2. Etapa de pruebas 1

Condiciones de ejecución: para la realización de las siguientes secciones de prueba es necesario iniciar el servidor de la aplicación. Las características del servidor tienen que ser 4 gb de RAM o superior. Las características de la red: 100.0 Mbps y se utilizará Xampp en su versión 1.8.

3.2.1. Secciones a probar: tiempos de respuesta en varios niveles, en un entorno local

En la presente sección se medirán y compararán los tiempos de respuesta para varios niveles de juegos en mapas generados automáticamente y mapas diseñados de forma manual. Se utilizarán las herramientas para el desarrollador del navegador web Chrome, versión 51.0.

Tabla 3.1. Pruebas a mapas de mar, tiempos de respuesta

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC1.1.Generar un mapa de mar equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la misma estarán los recursos necesarios para generar un mapa de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de mar está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.97/seg. Rendimiento total de un mapa de diseño manual: 1.97/seg.
CE1.2.Generar 5 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están diseñados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.08/seg. Rendimiento total de un mapa de diseño manual: 2.16/seg.
EC1.3.Generar 10 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están diseñados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.02/seg. Rendimiento total de un mapa de diseño manual: 2.31/seg.
EC1.4.Generar 15 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están diseñados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.10/seg. Rendimiento total de un mapa de diseño manual: 2.41/seg.

Tabla 3.2. Pruebas a mapas de asteroides, tiempos de respuesta

ID del escenario	**Escenario de la sección**	**Descripción**	**Resultado esperado**	**Resultado de la prueba**
EC2.1.Generar un mapa de asteroides equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de asteroides está desarrollado manualmente.	Al generar un mapa se simula el tránsito de un jugador por el primer nivel de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.68/seg. Rendimiento total de un mapa de diseño manual: 1.79/seg.
EC2.2.Generar 5 mapas de asteroides equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están diseñados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.68/seg. Rendimiento total de un mapa de diseño manual: 1.82/seg.
EC2.3.Generar 10 mapas de asteroides equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están diseñados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.69/seg. Rendimiento total de un mapa de diseño manual: 1.85/seg.
EC2.4.Generar 15 mapas de asteroides equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están diseñados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.69/seg. Rendimiento total de un mapa de diseño manual: 1.89/seg.

Tabla 3.3. Pruebas a mapas de ruinas, tiempos de respuesta

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC3.1.Generar un mapa de ruinas equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de ruinas está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.89/seg. Rendimiento total de un mapa de diseño manual: 1.91/seg.
EC3.2.Generar 5 mapas de ruinas equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están diseñados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.89/seg. Rendimiento total de un mapa de diseño manual: 1.98/seg.
EC3.3.Generar 10 mapas de ruinas equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están diseñados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.89/seg. Rendimiento total de un mapa de diseño manual: 2.03/seg.
EC3.4.Generar 15 mapas de ruinas equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están diseñados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.89/seg. Rendimiento total de un mapa de diseño manual: 2.08/seg.

Tabla 3.4. Pruebas a mapas de laberintos, tiempos de respuesta

ID del escenario	**Escenario de la sección**	**Descripción**	**Resultado esperado**	**Resultado de la prueba**
EC4.1.Generar un mapas de laberinto equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de laberintos está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.67/seg. Rendimiento total de un mapa de diseño manual: 1.83/seg.
EC4.2.Generar 5 mapas de laberintos equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están diseñados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.70/seg. Rendimiento total de un mapa de diseño manual: 1.87/seg.
EC4.3.Generar 10 mapas de laberintos equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están diseñados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.70/seg. Rendimiento total de un mapa de diseño manual: 1.92/seg.
EC4.4.Generar 15 mapas de laberintos equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están diseñados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.70/seg. Rendimiento total de un mapa de diseño manual: 1.98/seg.

3.2.2. Secciones a probar: tamaño total de transferencia de archivos, desde el servidor al cliente, en varios niveles para un entorno local

En la presente sección se analizará el tamaño total de los archivos descargados desde el servidor hacia el cliente, necesarios para la construcción de cada mapa, comparando la generación automática con la manual. Se utilizarán las herramientas para el desarrollador del navegador web: Chrome, versión 51.0.

Tabla 3.5. Pruebas a mapas de mar, tamaño total de archivos transferidos

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC5.1.Generar un mapa de mar equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de mar está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 5.3 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 5.6 mb.
EC5.2.Generar 5 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están diseñados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 5.3 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 5.6 mb.
EC5.3.Generar 10 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están diseñados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 5.3 mb.Tamaño total de los archivos descargados para mapas de diseño manual: 5.6 mb.

Continúa en la próxima página

Tabla 3.5 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC5.4.Generar 15 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están diseñados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 5.3 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 5.6 mb.

Tabla 3.6. Pruebas a mapas de asteroides, tamaño total de archivos transferidos

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC6.1.Generar un mapa de asteroides equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de asteroides está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.4 mb.
EC6.2.Generar 5 mapas de asteroides equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están diseñados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.4 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 4.4 mb.

Continúa en la próxima página

Tabla 3.6 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC6.3.Generar 10 mapas de asteroides equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están diseñados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.4 mb.Tamaño total de los archivos descargados para mapas de diseño manual: 4.4 mb.
EC6.4.Generar 15 mapas de asteroides equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están diseñados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.4 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 4.5 mb.

Tabla 3.7. Pruebas a mapas de ruinas, tamaño total de archivos transferidos

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC7.1.Generar un mapa de ruinas equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de ruinas está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.8 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.8 mb.

Continúa en la próxima página

Tabla 3.7 – *Continuación de la página anterior*

ID del escenario	**Escenario de la sección**	**Descripción**	**Resultado esperado**	**Resultado de la prueba**
EC7.2.Generar 5 mapas de ruinas equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están diseñados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.8 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 4.8 mb.
EC7.3.Generar 10 mapas de ruinas equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están diseñados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.8 mb.Tamaño total de los archivos descargados para mapas de diseño manual: 4.8 mb.
EC7.4.Generar 15 mapas de ruinas equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están diseñados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.8 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 4.9 mb.

Tabla 3.8. Pruebas a mapas de laberintos, tamaño total de archivos transferidos

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC8.1.Generar un mapa de laberintos equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de laberintos está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.1 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.1 mb.
EC8.2.Generar 5 mapas de laberintos equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están diseñados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.1 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 4.1 mb.
EC8.3.Generar 10 mapas de laberintos equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están diseñados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.1 mb.Tamaño total de los archivos descargados para mapas de diseño manual: 4.1 mb.

Continúa en la próxima página

Tabla 3.8 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC8.4.Generar 15 mapas de laberintos equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están diseñados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.1 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 4.2 mb.

3.2.3. Secciones a probar: tiempos de respuesta para mapas de varias dimensiones, en un entorno local

En la presente sección se medirán y compararán las velocidades de los tiempos de respuesta para mapas con varias dimensiones, se comparará esta variable en mapas generados automáticamente y mapas diseñados de forma manual. Se utilizarán las herramientas para el desarrollador del navegador web: Chrome, versión 51.0

Tabla 3.9. Pruebas a mapas de mar, midiendo los tiempos de respuesta para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC9.1.Generar un mapa de mar de dimensiones: 40 x 20 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de mar está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.85/seg. Rendimiento total de un mapa de diseño manual: 1.95/seg.

Continúa en la próxima página

Tabla 3.9 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC9.2.Generar un mapa de mar de dimensiones: 80 x 40 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de mar está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.84/seg. Rendimiento total de un mapa de diseño manual: 1.97/seg
EC9.3.Generar un mapa de mar de dimensiones: 160 x 80 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de mar está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.84/seg. Rendimiento total de un mapa de diseño manual: 2.13/seg.
EC9.4.Generar un mapa de mar de dimensiones: 1000 x 1000 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de mar está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.00/seg. Rendimiento total de un mapa de diseño manual: 2.51/seg.

Tabla 3.10. Pruebas a mapas de asteroides, midiendo los tiempos de respuesta para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC10.1.Generar un mapa de asteroides de dimensiones: 40 x 20 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de asteroides está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.66/seg. Rendimiento total de un mapa de diseño manual: 1.76/seg.
EC10.2.Generar un mapa de asteroides de dimensiones: 80 x 40 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de asteroides está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.66/seg. Rendimiento total de un mapa de diseño manual: 1.79/seg.
EC10.3.Generar un mapa de asteroides de dimensiones: 160 x 80 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de asteroides está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.70/seg. Rendimiento total de un mapa de diseño manual: 1.81/seg.

Continúa en la próxima página

Tabla 3.10 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC10.4.Generar un mapa de asteroides de dimensiones: 1000 x 1000 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de asteroides está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.75/seg. Rendimiento total de un mapa de diseño manual: 1.87/seg.

Tabla 3.11. Pruebas a mapas de ruinas, midiendo los tiempos de respuesta para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC11.1.Generar un mapa de ruinas de dimensiones: 40 x 20 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de ruinas está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.79/seg. Rendimiento total de un mapa de diseño manual: 1.87/seg.
EC11.2.Generar un mapa de ruinas de dimensiones: 80 x 40 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de ruinas está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.79/seg. Rendimiento total de un mapa de diseño manual: 1.89/seg.

Continúa en la próxima página

Tabla 3.11 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC11.3.Generar un mapa de ruinas de dimensiones: 160 x 80 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de ruinas está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.79/seg. Rendimiento total de un mapa de diseño manual: 1.91/seg.
EC11.4.Generar un mapa de ruinas de dimensiones: 1000 x 1000 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de ruinas está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.80/seg. Rendimiento total de un mapa de diseño manual: 1.96/seg.

Tabla 3.12. Pruebas a mapas de laberintos, midiendo los tiempos de respuesta para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC12.1.Generar un mapa de laberintos de dimensiones: 40 x 20 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de laberintos está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.63/seg. Rendimiento total de un mapa de diseño manual: 1.75/seg.

Continúa en la próxima página

Tabla 3.12 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC12.2.Generar un mapa de laberintos de dimensiones: 80 x 40 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de laberintos está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.64/seg. Rendimiento total de un mapa de diseño manual: 1.80/seg.
EC12.3.Generar un mapa de laberintos de dimensiones: 160 x 80 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de laberintos está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.64/seg. Rendimiento total de un mapa de diseño manual: 1.83/seg.
EC12.4.Generar un mapa de laberintos de dimensiones: 1000 x 1000 celdas.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de laberintos está desarrollado manualmente.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 1.68/seg. Rendimiento total de un mapa de diseño manual: 1.90.

3.3. Etapa de pruebas 2

Condiciones de ejecución: para la realización de la siguiente sección de prueba se utilizará la herramienta JMeter en la versión 2.4.

3.3.1. Sección a probar: tiempos de respuesta para el acceso concurrente a la aplicación, en un entorno local

En la presente sección se medirán y compararán las velocidades de los tiempos de respuesta para mapas de dimensiones 1000 x 1000, durante la simulación del acceso concurrente por parte de 50 usuarios a la aplicación. Se comparará dicha variable en mapas generados automáticamente y mapas diseñados de forma manual, la aplicación contendrá los recursos necesarios para crear 10 mapas de cada tipo (mar, asteroides, ruinas y laberintos). Se utilizará el navegador Chrome, versión 51.0.

Tabla 3.13. Pruebas midiendo los tiempos de respuesta para mapas de dimensiones 1000 x 1000

ID del escenario	**Escenario de la sección**	**Descripción**	**Resultado esperado**	**Resultado de la prueba**
EC12.4.1 Acceder a la aplicación.	Se accederá a la aplicación a través del navegador. En la aplicación se encuentra los recursos necesarios para generar todos los mapas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas estén diseñados manualmente.	Para esta prueba los parámetros de entrada de los algoritmos serán de y=1000 y x=1000 para cada mapa.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 7.06/seg. Rendimiento total de un mapa de diseño manual: 8.1/seg

3.4. Etapa de pruebas 3

Condiciones de ejecución: para la realización de la siguiente sección de prueba es necesario acceder al servidor web donde se encuentra la aplicación. Las características de la red: 100.0 Mbps.

3.4.1. Sección a probar: tiempos de respuesta para la aplicación en un entorno real

En la presente sección se analizarán los tiempos de respuesta para mapas de dimensiones 1000 x 1000. Se comparará dicha variable en mapas generados automáticamente y mapas diseñados de forma manual. La aplicación contendrá los recursos necesarios para crear 10 mapas de cada tipo (mar, asteroides, ruinas y laberintos). Para el desarrollo de las pruebas, se utilizarán las herramientas para el desarrollador del navegador web: Chrome, versión 51.0.

Tabla 3.14. Pruebas midiendo los tiempos de respuesta para mapas de dimensiones 1000 x 1000 en un entorno real.

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC12.4.2 Acceder a la aplicación.	Se accederá a la aplicación a través del navegador. En la aplicación se encuentra los recursos necesarios para generar todos los mapas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas estén diseñados manualmente.	Para esta prueba los parámetros de entrada de los algoritmos serán de y=1000 y x=1000 para cada mapa.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 6.40/seg. Rendimiento total de un mapa de diseño manual: 32.96/seg

3.4.2. Sección a probar: tamaño de archivos descargados desde el servidor durante la ejecución de la aplicación en un entorno real

En la presente sección se analiza el tamaño total de los archivos descargados desde el servidor durante la ejecución de la aplicación para mapas de dimensiones 1000 x 1000. Se comparará esta variable en mapas generados automáticamente y mapas diseñados de forma manual. La aplicación contendrá los recursos necesarios para crear 10 mapas de cada tipo (mar, asteroides, ruinas y laberintos). Durante las pruebas se utilizarán las herramientas para el desarrollador del navegador web Chrome en su versión 51.0.

Tabla 3.15. Pruebas para medir el tamaño total de archivos descargados desde el servidor para mapas de 1000 x 1000 en un entorno real.

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC12.4.3 Acceder a la aplicación.	Se accederá a la aplicación a través del navegador. En la aplicación se encuentra los recursos necesarios para generar todos los mapas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas estén diseñados manualmente.	Para esta prueba los parámetros de entrada de los algoritmos serán de y=1000 y x=1000 para cada mapa.	El total de archivos descargados desde el servidor tiene que ser menor en la aplicación que utiliza la GPC.	Tamaño total de archivos descargados durante la generación automática: 7,0 mb. Tamaño total de archivos descargados para la aplicación que utiliza mapas de desarrollo manual: 121,0 mb.

3.5. Resultados de las pruebas

En las pruebas realizadas los resultados fueron favorables en todos los casos de prueba. Los tiempos de respuesta se mantuvieron, en los mapas generados de forma automática, siempre iguales o menores a los tiempos de respuesta de sus homólogos diseñados manualmente, para varios niveles y varias dimensiones. También se pudo observar en todos los casos, una reducción en el tamaño total de archivos descargados desde el servidor hacia el cliente al utilizar los algoritmos propuestos.

3.6. Conclusiones del capítulo

En este capítulo se realizaron pruebas de carga a la aplicación, definiendo 32 casos de prueba para los mapas de videojuegos obtenidos a partir de algoritmos de GPC. Las pruebas se dividieron en 3 etapas.

- Durante la primera etapa de pruebas se demostró que la velocidad de respuesta de los mapas que utilizan la GPC es menor o igual que la velocidad de respuesta cuando se utilizan mapas diseñados de forma manual, logrando reducir hasta 0.31 segundos. También se logró probar que con la GPC se minimiza el tamaño total de los archivos descargados desde el servidor hacia el cliente, alcanzando diferencias de hasta 3.4 mega bytes. También se observó que al utilizar el navegador web Chrome, se obtienen menores tiempos de respuesta que al utilizar el navegador Firefox. La factibilidad de los mapas generados con GPC se hizo más evidente a partir de dimensiones de 1000 x 1000 celdas.
- Durante la segunda etapa de pruebas se midió el tiempo de respuesta en el acceso concurrente a la aplicación, comparando esta variable con el tiempo de respuesta de acceder a una versión de la aplicación donde los mapas son creados manualmente. Se demostró que los tiempos de respuesta en la aplicación que utiliza los algoritmos de GPC son más rápidos. Alcanzando una reducción de 1.04 segundos.
- Durante la tercera etapa de pruebas se comparó el tamaño total de archivos descargados desde el servidor hacia el cliente y los tiempos de respuesta en un entorno real. Se realizó una comparación midiendo estos mismos datos en una versión de la aplicación donde los mapas fueron diseñados manualmente. Se logró un reducción en los tiempos de respuesta con la utilización de los algoritmos creados de 26.56 segundos y se redujo el total de archivos descargados hacia el cliente en 114 mega bytes.

Conclusiones

- Las modificaciones realizadas al algoritmo Backtracker Recursivo posibilitaron la generación de mapas de asteroides, islas y ruinas, demostrando la flexibilidad del algoritmo Backtracker Recursivo.
- Las pruebas realizadas demostraron que la GPC para generar mapas de forma automática reduce el tiempo de respuesta y el tamaño total de los archivos almacenados en el servidor.

Recomendaciones

Se recomienda que se aplique la investigación realizada en videojuegos de navegador, donde se pueda apreciar los aportes de la GPC en la creación de contenido jugable. Se propone realizar pruebas concurrentes a la aplicación, realizadas en un entorno real.

Glosario

arcade máquinas de videojuegos disponibles en lugares como centros comerciales, restaurantes, bares, o salones recreativos especializados.. 11

browser game videojuego de navegador.. 6

Canvas permite la renderización interpretada dinámica de formas 2D en imágenes de mapas de bits.. 14

desktop escritorio, se utiliza para referirse a un videojuego de PC.. 1

HTTP protocolo de transferencia de hipertextos.. 14

JavaScript es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.. 14

Open Source es el software cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de software.. 14

Pila es una lista ordenada o estructura de datos en la que el modo de acceso a sus elementos es de tipo LIFO (del inglés Last In First Out, último en entrar, primero en salir) que permite almacenar y recuperar datos.. 19

plugins es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.. 14

rendering es un término informático utilizado para referirse al proceso de generar una imagen o vídeo.. 14

sprites cualquier mapa de bits que se dibuje en la pantalla, incluso si tiene que moverlo todo el procesador central y no cuenta con hardware especializado.. 14

tiled mosaico, referente a mapas de videojuegos formados a partir de imágenes pequeñas en forma de mosaico.. 14

videoconsolas es un sistema electrónico de entretenimiento para el hogar que permite ejecutar juegos.. 6

WebGL es una especificación estándar que está siendo desarrollada y es utilizada para mostrar gráficos 3D en navegadores web.. 14

Acrónimos

GA Genetic Algorithm. 9, 10

GPC Generación Procedural de Contenido. 4–6, 8, 11, 15, 30, 32–51, 58–80

NPC Non Player Character. 4, 9

RB Recursive Backtracker. 11, 12, 15, 21, 25, 27, 28

UCI Universidad de las Ciencias Informáticas. 1

Referencias bibliográficas

ANDREARRS. 2014. 2014. Dirección: ⟨http://hipertextual.com/archivo/2014/08/librerias-javascript-para-hacer-juegos/⟩.

ARRIBAS, G. 2013. COMPUTACIONAL APLICADA A LA GENERACION Y VALIDACION AUTOMATICA DE PANTALLAS DE VIDEOJUEGOS. 2013.

BA, Jee. 2009. *Guia de mapas en 2D*. 2009. Dirección: ⟨⟩.

BANCO MUNDIAL, Grupo del. 2016. *Usuarios de Internet (por cada 100 personas)*. 2016. Dirección: ⟨http://datos.bancomundial.org/indicador/IT.NET.USER.P2⟩.

BERMÚDEZ, Juan Luis. 2012. *Lagoonia, un entretenido juego de estrategia para tu navegador Chrome*. 2012. Dirección: ⟨http://www.softandapps.info/2012/04/22/lagoonia-un-entretenido-juego-de-estrategia-para-tu-navegador-chrome/⟩.

BUCK, Jamis. 2011. *Maze Generation: Algorithm Recap*. 2011. Dirección: ⟨http://weblog.jamisbuck.org/2011/2/7/maze-generation-algorithm-recap#article_body⟩.

BUCK, Jamis. 2012. *The Cellular Automaton Method for Cave Generation*. 2012. Dirección: ⟨http://jeremykun.com/2012/07/29/the-cellular-automaton-method-for-cave-generation/⟩.

COOK, Michael. 2013. *Generate Random Cave Levels Using Cellular Automata*. 2013. Dirección: ⟨http://gamedevelopment.tutsplus.com/tutorials/generate-random-cave-levels-using-cellular-automata--gamedev-9664⟩.

DOMÍNGUEZ, Francisco Ignacio Revuelta y SÁEZ, Alberto Bernabé. 2012. El videojuego en red social: un nuevo modelo de comunicación. *Tejuelo: Didáctica de la Lengua y la Literatura. Educación*. 2012, n.º 6, págs. 157-176.

ESCRIBANO, Gerardo Fernández. 2002. Introducción a Extreme Programming. 2002.

FRUGONI, Emilio. 2012. 2012. Dirección: ⟨http://oldiesfm.com.uy/los80.php?s=news&idx=333⟩.

GAMEPEDIA. 2012. *Procedural Generation*. 2012. Dirección: ⟨http://diablo.gamepedia.com/Procedural_Generation⟩.

HART, John C. 2001. Perlin noise pixel shaders. En. *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*. 2001, págs. 87-94.

HENDRIKX, Mark; MEIJER, Sebastiaan; VAN DER VELDEN, Joeri y IOSUP, Alexandru. 2013. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*. 2013, vol. 9, n.º 1, págs. 1.

JIMÉNEZ REY, Maréa Elizabeth. 2005. Un enfoque procedimental para la enseñanza de computación en carreras de Ingeniería. En. *I Jornadas de Educación en Informática y TICs en Argentina*. 2005.

JOHNSON, Lawrence; YANNAKAKIS, Georgios N y TOGELIUS, Julian. 2010. Cellular automata for real-time generation of infinite cave levels. En. *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. 2010, págs. 10.

JOSKOWICZ, José. 2008. Reglas y Prácticas en eXtreme Programming. 2008.

KIWIJS. 2013. 2013. Dirección: ⟨http://www.kiwijs.org/⟩.

LEE, Young-Seol y CHO, Sung-Bae. 2011. Context-aware petri net for dynamic procedural content generation in role-playing game. *Computational Intelligence Magazine, IEEE*. 2011, vol. 6, n.º 2, págs. 16-25.

LINKEDIN. 2016. 2016. Dirección: ⟨http://es.slideshare.net/mariapaulafernandezm/los-video-juegos-57991492⟩.

LONDOÑO, Jorge Hernan Abad. 2005. *TIPOS DE PRUEBAS DE SOFTWARE*. 2005. Dirección: ⟨http://www.slideshare.net/xpjair/tipos-de-pruebas-de-software-16980977⟩.

MELONJS. 2016. 2016. Dirección: ⟨http://melonjs.org/⟩.

PEDREÑO MOYA, Víctor. 2014. Generación procedural de mapes per a un JRPG. 2014.

PHASER. 2016. 2016. Dirección: ⟨http://phaser.io/⟩.

POMBO, Hector Lopez. 2009. *VIDEOJUEGO DE ESTRATEGIA PARA NAVEGADORES WEB EN TIEMPO REAL: WARBATTLE*. 2009.

ROZANSKI, Nick y WOODS, Eóin. 2012. *Software systems architecture: working with stakeholders using viewpoints and perspectives*. 2012.

TOGELIUS, Julian; YANNAKAKIS, Georgios N; STANLEY, Kenneth O y BROWNE, Cameron. 2010. Search-based procedural content generation. En. *Applications of Evolutionary Computation*. 2010, págs. 141-150.

TOZALGAMES. 2015. *Spelunker*. 2015. Url: ⟨http://www.spelunker-hd.com/⟩.

URRUTIA, Gerardo Abraham Morales; LÓPEZ, Claudia Esther Nava; MARTÍNEZ, Luis Felipe Fernández y CORRAL, Mirsha Aarón Rey. 2015. Procesos de desarrollo para videojuegos. *CULCyT*. 2015, n.º 37.

WIKIPEDIA. 2016. *Pac-Man*. 2016. Dirección: ⟨https://es.wikipedia.org/wiki/Pac-Man⟩.

YANNAKAKIS, Georgios N y TOGELIUS, Julian. 2011. Experience-driven procedural content generation. *Affective Computing, IEEE Transactions on*. 2011, vol. 2, n.º 3, págs. 147-161.

ZAHUMENSZKY, Carlos. 2015. *20 adictivos juegos gratuitos que puedes disfrutar desde tu navegador*. 2015. Dirección: ⟨http://es.gizmodo.com/20-adictivos-juegos-gratuitos-que-puedes-disfrutar-desd-1707789156⟩.

Apéndices

.1. Secciones a probar: tamaño total de transferencia de archivos desde el servidor al cliente, en mapas de varias dimensiones

En la presente sección se medirá y comparará el tamaño total de los archivos descargados desde el servidor hacia el navegador, para mapas con varias dimensiones, se comparará esta variable en mapas generados automáticamente y mapas desarrollados de forma manual. Se utilizarán las *herramientas para el desarrollador* del navegador web Chrome, versión 51.0.

Tabla 16. Pruebas a mapas de mar, midiendo el tamaño total de transferencia de archivos desde el servidor al cliente para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC13.1.Generar un mapa de mar de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 5.3 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 5.3 mb..
EC13.2.Generar un mapa de mar de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 5.3 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 5.3 mb.
EC13.3.Generar un mapa de mar de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 5.3 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 5.4 mb.

Continua en la próxima página

Tabla 16 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC13.4.Generar un mapa de mar de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 5.3 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 8.2 mb.

Tabla 17. Pruebas a mapas de asteroides, midiendo el tamaño total de transferencia de archivos desde el servidor al cliente para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC14.1.Generar un mapa de asteroides de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.4 mb.
EC14.2.Generar un mapa de asteroides de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.4 mb
EC14.3.Generar un mapa de asteroides de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.4 mb.

Continua en la próxima página

Tabla 17 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC14.4.Generar un mapa de asteroides de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 7.3 mb.

Tabla 18. Pruebas a mapas de ruinas, midiendo el tamaño total de transferencia de archivos desde el servidor al cliente para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC15.1.Generar un mapa de ruinas de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.8 mb.
EC15.2.Generar un mapa de ruinas de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.8 mb.
EC15.3.Generar un mapa de ruinas de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.8 mb.

Continua en la próxima página

Tabla 18 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC15.4.Generar un mapa de ruinas de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 7.8 mb.

Tabla 19. Pruebas a mapas de laberintos, midiendo el tamaño total de transferencia de archivos desde el servidor al cliente para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC16.1.Generar un mapa de laberintos de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.1 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.1 mb.
EC16.2.Generar un mapa de laberintos de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.1 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.1 mb.
EC16.3.Generar un mapa de laberintos de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.1 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.1 mb.

Continua en la próxima página

Tabla 19 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC16.4.Generar un mapa de laberintos de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.1 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 7.0 mb.

.2. Secciones a probar: tiempos de respuesta en varios niveles, usando Firefox

En la presente sección se medirán y compararán las velocidades de los tiempos de respuesta para varios niveles de juegos, en mapas generados automáticamente y mapas desarrollados de forma manual. Se utilizarán las herramientas para el desarrollador del navegador web Firefox, versión 46.0.1.

Tabla 20. Pruebas a mapas de mar, tiempos de respuesta

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC17.1.Generar un mapa de mar equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de mar está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 3.72/seg. Rendimiento total de un mapa de diseño manual: 4.52/seg.
EC17.2.Generar 5 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están desarrollados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 3.75/seg. Rendimiento total de un mapa de diseño manual: 4.83/seg.

Continua en la próxima página

Tabla 20 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC17.3.Generar 10 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están desarrollados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 3.73/seg. Rendimiento total de un mapa de diseño manual: 4.89/seg.
EC17.4.Generar 15 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están desarrollados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 3.72/seg. Rendimiento total de un mapa de diseño manual: 4.95/seg.

Tabla 21. Pruebas a mapas de asteroides, tiempos de respuesta

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC18.1.Generar un mapa de asteroides equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de asteroides está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.31/seg. Rendimiento total de un mapa de diseño manual: 2.58/seg.

Continua en la próxima página

Tabla 21 – *Continuación de la página anterior*

ID del escenario	**Escenario de la sección**	**Descripción**	**Resultado esperado**	**Resultado de la prueba**
EC18.2.Generar 5 mapas de asteroides equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están desarrollados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.36/seg. Rendimiento total de un mapa de diseño manual: 2.75/seg.
EC18.3.Generar 10 mapas de asteroides equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están desarrollados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.36/seg. Rendimiento total de un mapa de diseño manual: 2.88/seg.
EC18.4.Generar 15 mapas de asteroides equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están desarrollados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.36/seg. Rendimiento total de un mapa de diseño manual: 2.95/seg.

Tabla 22. Pruebas a mapas de ruinas, tiempos de respuesta

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC19.1.Generar un mapa de ruinas equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de ruinas está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.95/seg. Rendimiento total de un mapa de diseño manual: 3.27/seg.
EC19.2.Generar 5 mapas de ruinas equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están desarrollados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.95/seg. Rendimiento total de un mapa de diseño manual: 3.35/seg.
EC19.3.Generar 10 mapas de ruinas equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están desarrollados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.94/seg. Rendimiento total de un mapa de diseño manual: 3.41/seg.

Continua en la próxima página

Tabla 22 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC19.4.Generar 15 mapas de ruinas equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están desarrollados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.94/seg. Rendimiento total de un mapa de diseño manual: 3.69/seg.

Tabla 23. Pruebas a mapas de laberintos, tiempos de respuesta

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC20.1.Generar un mapas de laberinto equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de laberintos está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.8/seg. Rendimiento total de un mapa de diseño manual: 2.94/seg.
EC20.2.Generar 5 mapas de laberintos equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están desarrollados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.81/seg. Rendimiento total de un mapa de diseño manual: 2.98/seg.

Continua en la próxima página

Tabla 23 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC20.3.Generar 10 mapas de laberintos equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están desarrollados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.82/seg. Rendimiento total de un mapa de diseño manual: 3.01/seg.
EC20.4.Generar 15 mapas de laberintos equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están desarrollados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.81/seg. Rendimiento total de un mapa de diseño manual: 3.31/seg.

.3. Secciones a probar: tamaño total de transferencia de archivos, desde el servidor al cliente, en varios niveles, usando Firefox

En la presente sección se analizará el tamaño total de los archivos, descargados desde el servidor hacia el cliente, necesario para la construcción de cada mapa, comparando la generación automática con la manual. Se utilizarán las herramientas para el desarrollador del navegador web: Firefox, versión 46.0.1.

Tabla 24. Pruebas a mapas de mar, tamaño total de archivos transferidos

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC21.1.Generar un mapa de mar equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de mar está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.0 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.0 mb.
EC21.2.Generar 5 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están desarrollados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.0 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 4.01 mb.
EC21.3.Generar 10 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están desarrollados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.0 mb.Tamaño total de los archivos descargados para mapas de diseño manual: 4.02 mb.
EC21.4.Generar 15 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de mar y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de mar están desarrollados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 4.0 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 4.03 mb.

Tabla 25. Pruebas a mapas de asteroides, tamaño total de archivos transferidos

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC22.1.Generar un mapa de mar equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de asteroides está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 3.09 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 3.09 mb.
EC22.2.Generar 5 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están desarrollados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 3.09 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 3.09 mb.
EC22.3.Generar 10 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están desarrollados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 3.09 mb.Tamaño total de los archivos descargados para mapas de diseño manual: 3.09 mb.

Continua en la próxima página

Tabla 25 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC22.4.Generar 15 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de asteroides y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de asteroides están desarrollados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 3.09 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 3.10 mb.

Tabla 26. Pruebas a mapas de ruinas, tamaño total de archivos transferidos

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC23.1.Generar un mapa de mar equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de ruinas está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 3.47 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 3.47 mb.
EC23.2.Generar 5 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están desarrollados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 3.47 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 3.48 mb.

Continua en la próxima página

Tabla 26 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC23.3.Generar 10 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están desarrollados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 3.47 mb.Tamaño total de los archivos descargados para mapas de diseño manual: 3.51 mb.
EC23.4.Generar 15 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de ruinas y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de ruinas están desarrollados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 3.47 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 3.55 mb.

Tabla 27. Pruebas a mapas de laberintos, tamaño total de archivos transferidos

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC24.1.Generar un mapa de mar equivalente a un nivel de juego, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar un mapa de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde el mapa de laberintos está desarrollado manualmente.	Al generar un mapa estamos simulando el tránsito de un jugador por el primer nivel de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 2.7 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 2.7 mb.

Continua en la próxima página

Tabla 27 – *Continuación de la página anterior*

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC24.2.Generar 5 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 5 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están desarrollados manualmente.	Al generar 5 mapas estamos simulando el tránsito de un jugador por los 5 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 2.7 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 2.73 mb.
EC24.3.Generar 10 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 10 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están desarrollados manualmente.	Al generar 10 mapas estamos simulando el tránsito de un jugador por los 10 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 2.7 mb.Tamaño total de los archivos descargados para mapas de diseño manual: 2.76 mb.
EC24.4.Generar 15 mapas de mar equivalentes a igual número de niveles, matriz de 17x7.	Acceder a la aplicación desde el navegador. En la aplicación se encuentra los recursos necesarios para generar 15 mapas de laberintos y medir el tiempo de respuesta. Comparar realizando la misma operación accediendo a una versión de la aplicación donde los mapas de laberintos están desarrollados manualmente.	Al generar 15 mapas estamos simulando el tránsito de un jugador por los 15 primeros niveles de un videojuego.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de mapas: 2.72 mb. Tamaño total de los archivos descargados para mapas de diseño manual: 2.79 mb.

.4. Secciones a probar: tiempos de respuesta para mapas de varias dimensiones, usando Firefox

En la presente sección se medirán y compararán las velocidades de los tiempos de respuesta para mapas con varias dimensiones, se comparará esta variable en mapas generados automáticamente y mapas desarrollados de forma manual. Se utilizarán las herramientas para el desarrollador del navegador web Firefox, versión 46.0.1.

Tabla 28. Pruebas a mapas de mar, midiendo los tiempos de respuesta para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC25.1.Generar un mapa de mar de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 3.73/seg. Rendimiento total de un mapa de diseño manual: 4.12/seg.
EC25.2.Generar un mapa de mar de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 3.90/seg. Rendimiento total de un mapa de diseño manual: 4.15/seg.
EC25.3.Generar un mapa de mar de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 4.00/seg. Rendimiento total de un mapa de diseño manual: 4.20/seg.
EC25.4.Generar un mapa de mar de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 4.12/seg. Rendimiento total de un mapa de diseño manual: 4.55/seg.

Tabla 29. Pruebas a mapas de asteroides, midiendo los tiempos de respuesta para mapas de varias dimensiones

ID del escenario	**Escenario de la sección**	**Descripción**	**Resultado esperado**	**Resultado de la prueba**
EC26.1.Generar un mapa de asteroides de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.19/seg. Rendimiento total de un mapa de diseño manual: 2.36/seg.
EC26.2.Generar un mapa de asteroides de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.26/seg. Rendimiento total de un mapa de diseño manual: 2.53/seg.
EC26.3.Generar un mapa de asteroides de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.23/seg. Rendimiento total de un mapa de diseño manual: 2.60/seg.
EC26.4.Generar un mapa de asteroides de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.40/seg. Rendimiento total de un mapa de diseño manual: 2.70/seg.

Tabla 30. Pruebas a mapas de ruinas, midiendo los tiempos de respuesta para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC27.1.Generar un mapa de ruinas de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.84/seg. Rendimiento total de un mapa de diseño manual: 3.0/seg.
EC27.2.Generar un mapa de ruinas de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.9/seg. Rendimiento total de un mapa de diseño manual: 3.1/seg.
EC27.3.Generar un mapa de ruinas de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.94/seg. Rendimiento total de un mapa de diseño manual: 3.18/seg.
EC27.4.Generar un mapa de ruinas de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.93/seg. Rendimiento total de un mapa de diseño manual: 3.3/seg.

Tabla 31. Pruebas a mapas de laberintos, midiendo los tiempos de respuesta para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC28.1.Generar un mapa de laberintos de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.60/seg. Rendimiento total de un mapa de diseño manual: 2.66/seg.
EC28.2.Generar un mapa de laberintos de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.63/seg. Rendimiento total de un mapa de diseño manual: 2.70/seg.
EC28.3.Generar un mapa de laberintos de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar los tiempos de carga.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC.	Rendimiento total de la generación automática: 2.61/seg. Rendimiento total de un mapa de diseño manual: 2.73/seg.
EC28.4.Generar un mapa de laberintos de dimensiones: 1000 x 1000 celdas	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar los tiempos de carga	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000	El tiempo de carga tiene que ser menor en la aplicación que utiliza la GPC	Rendimiento total de la generación automática: 2.60/seg. Rendimiento total de un mapa de diseño manual: 2.80/seg.

.5. Secciones a probar: tamaño total de transferencia de archivos desde el servidor al cliente, en mapas de varias dimensiones, usando Firefox

En la presente sección se medirá y comparará el tamaño total de los archivos descargados desde el servidor hacia el navegador, para mapas con varias dimensiones, se comparará esta variable en mapas generados automáticamente y mapas desarrollados de forma manual. Se utilizarán las *herramientas para el desarrollador* del navegador web Firefox, versión 46.0.1.

Tabla 32. Pruebas a mapas de mar, midiendo el tamaño total de transferencia de archivos desde el servidor al cliente para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC29.1.Generar un mapa de mar de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.02 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.02 mb.
EC29.2.Generar un mapa de mar de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.02 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.02 mb
EC29.3.Generar un mapa de mar de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.02 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.04 mb.
EC29.4.Generar un mapa de mar de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **Mar** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de mar creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 4.02 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 5.53 mb.

Tabla 33. Pruebas a mapas de asteroides, midiendo el tamaño total de transferencia de archivos desde el servidor al cliente para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC30.1.Generar un mapa de asteroides de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 3.1 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 3.1 mb.
EC30.2.Generar un mapa de asteroides de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 3.1 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 3.1 mb.
EC30.3.Generar un mapa de asteroides de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 3.1 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 3.11 mb.
EC30.4.Generar un mapa de asteroides de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **asteroides** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de asteroides creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 3.1 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.61 mb.

Tabla 34. Pruebas a mapas de ruinas, midiendo el tamaño total de transferencia de archivos desde el servidor al cliente para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC31.1.Generar un mapa de ruinas de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 3.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 3.45 mb.
EC31.2.Generar un mapa de ruinas de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 3.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 3.46 mb.
EC31.3.Generar un mapa de ruinas de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 3.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 3.49 mb.
EC31.4.Generar un mapa de ruinas de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **ruinas** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de ruinas creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 3.4 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.98 mb.

Tabla 35. Pruebas a mapas de laberintos, midiendo el tamaño total de transferencia de archivos desde el servidor al cliente para mapas de varias dimensiones

ID del escenario	Escenario de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC32.1.Generar un mapa de laberintos de dimensiones: 40 x 20 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=40 y x=20.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 2.7 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 2.7 mb.
EC32.2.Generar un mapa de laberintos de dimensiones: 80 x 40 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=80 y x=40.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 2.7 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 2.71 mb.
EC32.3.Generar un mapa de laberintos de dimensiones: 160 x 80 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=160 y x=80.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 2.7 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 2.74 mb.
EC32.4.Generar un mapa de laberintos de dimensiones: 1000 x 1000 celdas.	Acceder a la pantalla Menú, pulsar el botón **laberintos** para generar un mapa de este tipo, luego cargar una versión de la aplicación con un mapa de laberintos creado de forma manual, ejecutar y comparar el tamaño total de los archivos descargados desde el servidor.	Para esta prueba los parámetros de entrada del algoritmo utilizado serán de y=1000 y x=1000.	El tamaño total de los archivos descargados tiene que ser menor o igual en la aplicación que utiliza la GPC.	Tamaño total de los archivos descargados para la generación automática de un mapa: 2.7 mb. Tamaño total de los archivos descargados para un mapa de diseño manual: 4.23 mb.