

Agentes Cartógrafos

Sistemas Inteligentes



Autores

Jorge O. Blanchard Cruz, Laura A. Gómez González

Índice

[Introducción 2](#_Toc408420655)

[Descripción del problema a tratar 3](#_Toc408420656)

[Respecto al desarrollo 3](#_Toc408420657)

[Equipo de desarrollo 3](#_Toc408420658)

[Herramientas de programación 3](#_Toc408420659)

[Árbol de directorios 4](#_Toc408420660)

[Técnicas de IA utilizadas 4](#_Toc408420661)

[Diseño de los agentes 5](#_Toc408420662)

[Estructura de clases desarrollada 5](#_Toc408420663)

[Pizarra 5](#_Toc408420664)

[Robots Cartógrafos 6](#_Toc408420665)

[Mapa 6](#_Toc408420666)

[Cronograma del proyecto 7](#_Toc408420667)

[Funcionamiento 8](#_Toc408420668)

[Generador de mapas 8](#_Toc408420669)

[Aplicación Web 9](#_Toc408420670)

[Interfaz 9](#_Toc408420671)

[Mapeo 10](#_Toc408420672)

[Testing en diversos mapas 11](#_Toc408420673)

[16x16 – Un robot 11](#_Toc408420674)

[30x30 – Cinco robots 11](#_Toc408420675)

[30x30 – Cuatro robots 12](#_Toc408420676)

[26x20 – Un robot 12](#_Toc408420677)

[34x58 – Tres robots 13](#_Toc408420678)

[40x50 – Seis robots 13](#_Toc408420679)

# Introducción

Uno de los problemas más abordados en robótica, así como en inteligencia artificial, es el problema de mapeo. Tanto es así, que es tratado mundialmente, no presentando ninguna dificultad encontrar proyectos relacionados. Empecemos, pues, definiendo el problema y sus posibles variantes.

Un problema de mapeo, consiste en generar un mapa de un espacio, a priori desconocido, es decir, consiste en cartografiar un espacio. Para lograr dicho objetivo, dicha meta, hay que tener varias consideraciones:

1. **Ser capaces de definir el espacio a cartografiar.** Definir un espacio se puede llegar a hacer de muchas maneras, no necesariamente precisas. Imaginemos un espacio definido por unas coordenadas de latitud, lo cual sería enormemente preciso, frente a una indicación de una dirección, hasta el final de la calle. En la segunda variante nos encontramos con la dificultad adicional de descubrir qué es el final de la calle y dónde está.
2. **El conocimiento necesario de los cartógrafos acerca del entorno a cartografiar.** Cierto conocimiento del entorno es absolutamente necesario para que el cartógrafo pueda moverse por éste. Sin éste, podría ser que el agente chocara contra una pared.
3. **La representación del mapa y los elementos presentes en el entorno.** Los robots cartógrafos han de saber cómo escribir el mapa de manera legible para quien vaya a leerlo después y, por tanto, ha de reconocer ciertos elementos de su entorno para poder representarlos en dicho mapa.
4. **Cómo interactuarán dichos robots entre ellos**, es decir, cómo se comunicarán entre ellos. Hace falta aclarar el medio para transmitir los mensajes, los mensajes que los robots pueden generar y el significado que tienen para ellos dichos mensajes.

Tal como vemos, el problema de generar un mapa de una localización es bastante complejo de abordar a la hora de definir todos los factores influyentes. También, existe una variante de éste, que también incluye la deslocalización del robot respecto del mapa, conocido por SLAM (Simultaneous Location And Mapping), localización y mapeo simultáneo.

En el presente informe, explicaremos detalladamente la variación del problema de mapeo que se ha tratado, así como su funcionamiento, desde el punto de vista teórico. Posteriormente, se hablará de las herramientas de desarrollo y su elección acorde a las características específicas del problema, haciendo hincapié en aspectos técnicos. Seguidamente, se procederá a explicar el funcionamiento de los programas desarrollados y sus interfaces, para acabar con algunos tests realizados para demostrar la validez del programa.

# Descripción del problema a tratar

En este proyecto se tratará de abordar un problema de mapeo, con localización conocida. Concretamente, simularemos el entorno de una cueva desconocida que un grupo de robots habrán de cartografiar. Así pues, procedamos a definir los cuatro puntos característicos, mencionados anteriormente.

1. **El espacio a cartografiar** será, como hemos mencionado anteriormente, una cueva. El área total del mapa se considerará conocida, concretamente, estaremos hablando de un mapa de rejilla donde cada robot podrá ubicarse en una coordenada (x,y) de este mapa.
2. **El conocimiento del entorno necesario.** Nuestros agentes serán capaces de poder girar y de poder avanzar únicamente en una dirección, hacia delante. También podrán diferenciar entre paredes, agua, obstáculos removibles y camino libre.
3. **Representación del mapa y los elementos del entorno.** El mapa será representado como una matriz en la que se indicarán las casillas libres, es decir, los caminos de la cueva. Tal como hemos mencionado en el punto anterior, el entorno constará de paredes, agua, obstáculos removibles y camino libre. El robot podrá avanzar sólo por los caminos libres, es decir, sin obstáculos, aunque también podrá destruir y apartar de su camino aquellos obstáculos que identifique como obstáculos removibles.
4. **Interacción entre robots.** Los robots se comunicarán entre ellos mediante un sistema de pizarra, comunicando las rutas inexploradas para su posterior asignación a robots que estén libres. También, dichos robots serán capaces de detectar colisiones con los de su especie, evitando así la colisión entre dos robots.

# Respecto al desarrollo

En este apartado trataremos sobre los aspectos más técnicos del proyecto realizado. Hablaremos acerca de las herramientas utilizadas, algoritmos, construcción del programa… También incluiremos el cronograma del proyecto, con todas las distintas tareas y pasos seguidos en su realización

## Equipo de desarrollo

* CPU: Intel Core i7 3.60GHz (caché 10MB)
* Memoria RAM: 16,0 GB distribuidos en 4 canales.
* GPU: AMD ATI Radeon R9 290 4GB GDDR5 DirectX 11.2
* O.S: Windows 8.1 Pro N x64

Con este equipo se configura la aplicación con todas las opciones gráficas activadas siendo el rendimiento óptimo con mapas de dimensión 99x99.

## Herramientas de programación

Para el desarrollo de este proyecto, se ha optado por un desarrollo web. Esto permite una gran portabilidad del proyecto, aunque también los navegadores difieren en características y esto presenta algunos problemas. También se han escogido, debido a la facilidad de diseño y debido a la simplicidad del problema abordado entornos, 3D sencillos.

El navegador para el cual se ha desarrollado este proyecto es Mozilla Firefox, sabiendo que presenta errores con Google Chrome. Este proyecto ha sido testado bajo las últimas versiones de Mozilla Firefox presentes hasta la fecha, esto es la versión: 34.0.5.

Por tanto, la página web se ha escrito en HTML, utilizando también CSS. El código funcional se ha desarrollado en Javascript. Sin embargo, este proyecto incluye un soporte para generar mapas, el cual ha sido desarrollado enteramente en Visual Basic.

Para generar el entorno 3D, así como para el *look and feel* de la web, se han utilizado los siguientes recursos:

* **jQuery:** es una librería utilizada ampliamente en la web que facilita el acceso a los elementos del árbol DOM de HTML.
* **Metro UI:** es una librería que proporciona una estética similar a la utilizada en la interfaz metro de Windows 8 y Windows 8.1. Dotando así a la página web de cierto estilo característico
* **Three.js:** se han utilizado ciertos componentes de esta librería para generar los gráficos en 3D.

## Árbol de directorios

* **AppIssues:** Se encuentra la aplicación GeneradorMapa.
* **css:** Ficheros de estilos de la aplicación web.
* **data:** base de datos necesarios para el correcto funcionamiento de la aplicación.
* **fonts:** estilos de letras.
* **js:** Contiene los ficheros de código fuente de la aplicación en lenguaje *javascript* así como las librerías necesarias.
  + **main.js:** Fichero de código principal de funcionamiento del UI y creación inicial de las instancias de clase.
  + **C3DWorld.js:** Clase que contiene las estructuras y métodos para el funcionamiento de la escena en 3d.
  + **CMothership.js:** Clase que provee de información al robot.
  + **CAgent.js:** Clase que implementa al robot y su algoritmo de búsqueda A\*.

## Técnicas de IA utilizadas

Para realizar este proyecto se han utilizado diversos algoritmos pertenecientes a la inteligencia artificial, así como diversas técnicas pertenecientes a esta área.

El primer paso en todo sistema, ya que es básico, es el concepto de vida artificial. Este concepto consiste en dotar al agente de percepción del entorno, así como de acciones para interactuar con él. En nuestro proyecto, no podía faltar esta técnica ya que los agentes cartógrafos interactuarán con la cueva y entre ellos.

Para interactuar entre ellos, se ha implementado un sistema de pizarra, consistente en un sistema de apuntes de rutas inexploradas. De esta manera, los robots han de ser dotados de la capacidad para escribir y leer de esta pizarra.

Otra técnica empleada, da la solución al problema de los caminos mínimos. Para ello, se utiliza el algoritmo A\*, el cual permite hallar el camino mínimo entre dos puntos. Esto se utilizará para posicionar a los distintos agentes al principio de la ruta a explorar.

Cabe destacar también que el comportamiento de los robots ha sido inspirado en un algoritmo de búsqueda en profundidad, sin embargo, no se halla implementado directamente, sino que emerge un símil de él del comportamiento colectivo de los agentes.

## Diseño de los agentes

Los agentes implementados en este proyecto, se basan en una arquitectura híbrida. Esto se debe a las necesidades de planificación, a la vez de autonomía, que precisa el problema.

Los agentes, a nivel individual, presentarán una tabla de reacción-acción, sin embargo, dichos agentes también se panificarán para comunicarse entre ellos, decidir rutas y llegar a ellas. Para ello, el agente debe de disponer de diversos modos de actuación, que le permitan acatar órdenes y también gobernarse por sí mismo.

Para la comunicación entre éstos, se utilizará un sistema de pizarra, de tal manera que todos los agentes puedan escribir en ella, y que ellos mismo puedan leerla y borrar las distintas entradas esccritas. También se tendrá en cuenta que el mapa que se generará será compartido para todos los agentes, con permisos de lectura y escritura en él.

## Estructura de clases desarrollada

Ilustración 1: Vista General de las clases involucradas en el proyecto

En la ilustración 1, podemos ver de manera gráfica los distintos elementos presentes en el proyecto. Procederemos, entonces, a explicar cada una de ellas en detalle.

### Pizarra

La pizarra es un elemento público para todos los agentes, de manera que ellos tengan acceso a ella indistintamente. Así pues, la pizarra se presentará internamente como una lista de rutas, de tamaño variable. Para ser concretos, tendremos una serie de coordenadas y direcciones almacenadas.

### Robots Cartógrafos

Ilustración 2: Robot cartógrafo

Cada robot cartógrafo presente en el mapa será una instancia de la clase robot. Dicha clase, constará de los métodos y atributos necesarios para que cada robot se administre a sí mismo.

Los parámetros comunes que describen al robot, serán los siguientes:

* Su posición inicial, o de partida.
* Su localización actual, expresada como dos coordenadas.
* Su nombre identificativo.
* Sus parámetros de movimiento, los cuales determinará cuánto ha de avanzar en cada paso o el giro máximo que puede girar.
* Lista de trayectorias ordenadas según la distancia del robot a ellas.

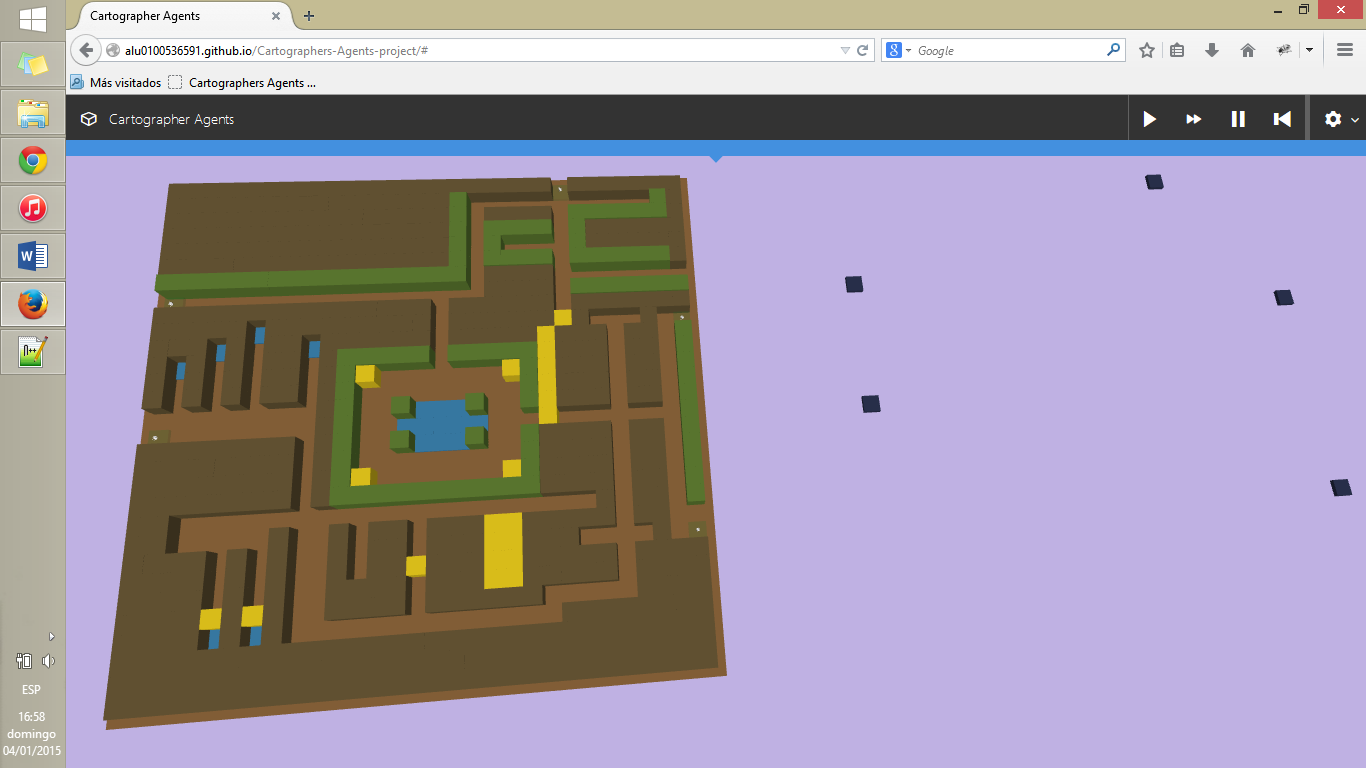
Como funciones comunes a todos los robots, se incluirán las siguientes:

* Función de inicialización.
* Funciones útiles, tales como para pasar de radianes a grados, o para redondear números en punto flotante.
* Función que permite representar el robot gráficamente.
* Funciones para el movimiento y la vida artificial del robot. Esto incluye funciones para moverse en una dirección bloque a bloque, girar y detectar colisiones.
* Funciones para la comunicación entre agentes, las cuales le permitirá acceder a la pizarra y obtener información de ella, a su vez que manipularla.
* Función implementación del A\* para calcular el camino mínimo.
* Funciones para determinar la trayectoria más cercana y accesible al robot. Las cuales incluyen métodos para ordenar y almacenar las trayectorias.

### Mapa

El mapa internamente está representado por una matriz, la cual contiene números que representan lo que en él hay. El mapa, como se ha mencionado antes, está formado por bloques, los cuales pueden ser de distintos tipos:

|  |  |  |
| --- | --- | --- |
| Número | Tipo | Representación gráfica |
| -1 | Camino libre | Ausencia de bloque, dejando ver la base del mapa |
| 1 | Bloque obstáculo | Bloque de color marrón oscuro |
| 2 | Obstáculo removible | Bloque de color amarillo |
| 3 | Bosque | Bloque de color verde |
| 4 | Agua | Relieve de color azul, de menor altura que un bloque. |

Cada tipo de bloque difiere en color y en representación. No obstante, dichos bloques se pueden agrupar en dos clases:

1. **Obstáculos**: esto incluye los bloques 1, 3 y 4. Bajo ningún concepto el robot podrá pasar sobre ellos, es decir, los detectará como colisiones.
2. **Obstáculos removibles:** constituidos únicamente por el tipo de bloque 2. Estos obstáculos pueden ser destruidos por el robot, dejando en su lugar el camino libre.

Ilustración 3: Ejemplo de mapa

## Cronograma del proyecto

En este apartado, veremos, ordenadamente, las tareas seguidas para la realización de este proyecto.

1. **Gráfico**
   1. Definir Medidas
   2. Crear Matriz de Pruebas
   3. Crear Robot
   4. Programación de Colisiones
   5. *Generación de Mapas*
      1. Generación de los Ficheros de Mapas
      2. Mapear Ficheros a Gráficos
2. **Programación IA**
   1. *Creación de Estructuras de Datos*
      1. Representación del Mapa
      2. Representación del Agente
      3. Acciones y sensores
   2. *Programación de Búsqueda*
      1. Programar el Task Manager
      2. Programación de un Agente
      3. Comunicación entre Agentes
   3. *Programación Caminos Mínimos*
      1. Programación de la Estrategia de Búsqueda
      2. Programar Asignación de Caminos

Durante todo este desarrollo, se han ido puliendo aspectos gráficos y relacionados con la interfaz web. Así mismo, la documentación aquí presente así como el testeo y posterior corrección de errores han sido efectuados a lo largo de todo el proyecto.

# Funcionamiento

## Generador de mapas

El generador de mapas, proporciona una interfaz gráfica sobre la cual crear los mapas que los robots explorarán y cartografiarán. Dichos mapas, se representarán por ficheros.

Tal como podemos observar en la ilustración número 4, la interfaz intuitiva facilita el diseño de mapas. Gracias al menú de edición, situado a la izquierda, se puede crear un mapa de dimensiones variables, cargar un mapa ya previamente creado para su modificación o bien guardar el mapa que actualmente se esté editando.

Para editar un mapa, tenemos distintas opciones que aparecen, también, en la barra lateral izquierda. El método de modificación por casillas consiste en escoger el material que vayamos a colocar y, posteriormente, seleccionar la posición, o casilla, donde queremos que aparezca dicho material.

También, el generador de mapas cuenta con opciones de edición rápida como ‘Bordes’ y ‘Rellenar’. La primera opción permite bordear el mapa de paredes y la segunda, rellena el mapa en su totalidad con ellas.

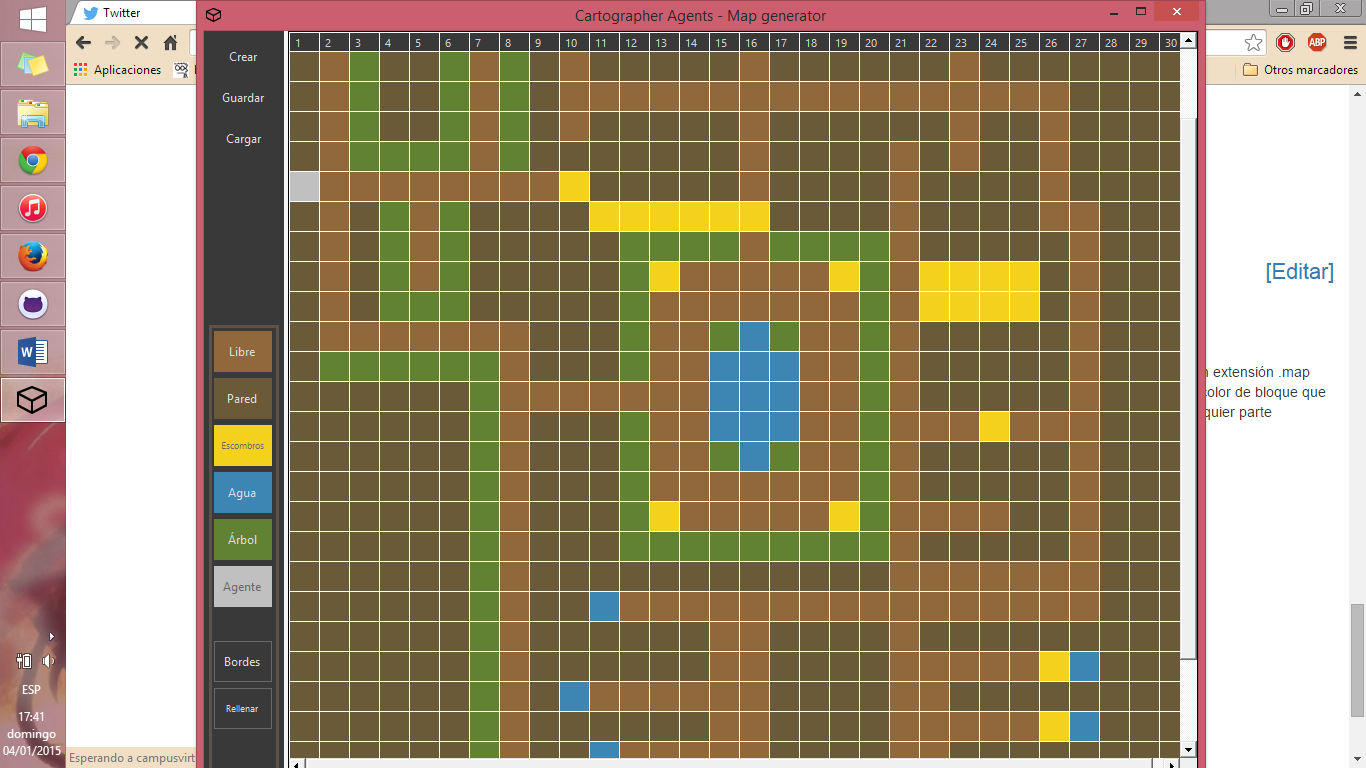


Ilustración 4: Generador de Mapas

## 

## Aplicación Web

### Interfaz

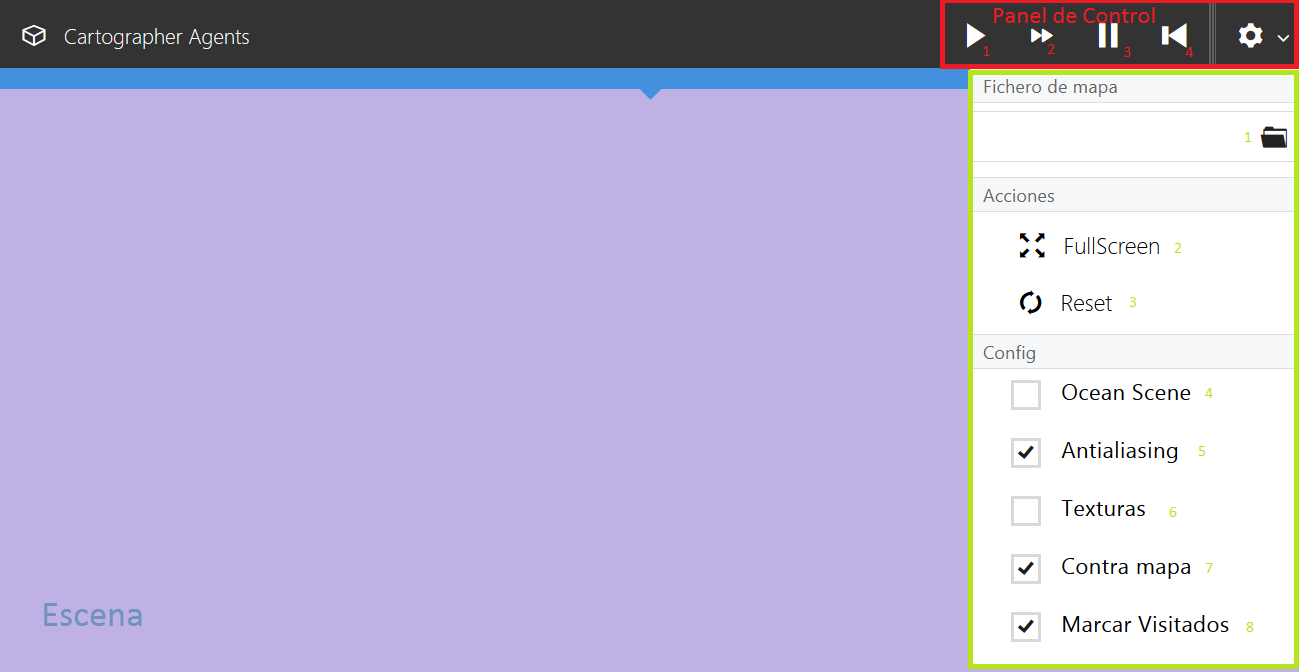
La aplicación web consta de una barra de navegación identificativa con el nombre del proyecto, compuesta por el panel de control y el menú desplegable. Esto se encuentra sobre la escena donde se mostrará la ejecución del programa.

Ilustración 5: Vista de la Interfaz de usuario

El panel de control, parecido al control estándar en reproductores multimedia, está compuesto por:

1. **Play**: inicia/reanuda la ejecución
2. **Adelantar:** modificar la velocidad de ejecución
3. **Pausa:** pausa la ejecución
4. **Reset:** devuelve la ejecución al punto antes de que se iniciara

El menú de opciones, se encuentra bajo el engranaje identificativo de ajustes:

* **Fichero de Mapa**
  1. *Cargar Mapa:* pulsando sobre el botón, nos permitirá cargar un fichero mapa desde nuestro ordenador.
* **Acciones**

1. *Fullscreen:* botón de pantalla completa.
2. *Reset:* botón para aplicar los cambios efectuados en el entorno.

* **Config**

1. *Ocean Scene:* carga una escena oceánica al atardecer de fondo.
2. *Antialiasing:* modo para difuminar los bordes.
3. *Texturas:* posibilidad de añadir texturas al mapa, sustituyendo los bloques de color liso.
4. *Contramapa:* muestra el mapa generado por los robots
5. *Marcar Visitados:* sombrea en el mapa original las casillas visitadas.



Para controlar la escena en 3D, dispondremos del ratón, el cual nos permitirá realizar distintas acciones que nos dotarán de control total sobre la visualización de la escena.

* *Botón izquierdo:* rotar escena.
  + *Botón central:* ajustar el zoom. También, si dejamos pulsado, controlaremos el zoom deslizando el ratón.
  + *Botón derecho*: desplazar escena.

### 

### Mapeo

Al cargar un mapa, podremos observar la aparición de dos elementos en pantalla. El primero de ellos es un mapa de simulación para los robots, tal como se había visto anteriormente, y, a su lado, aparece el mapa generado en el instante actual.

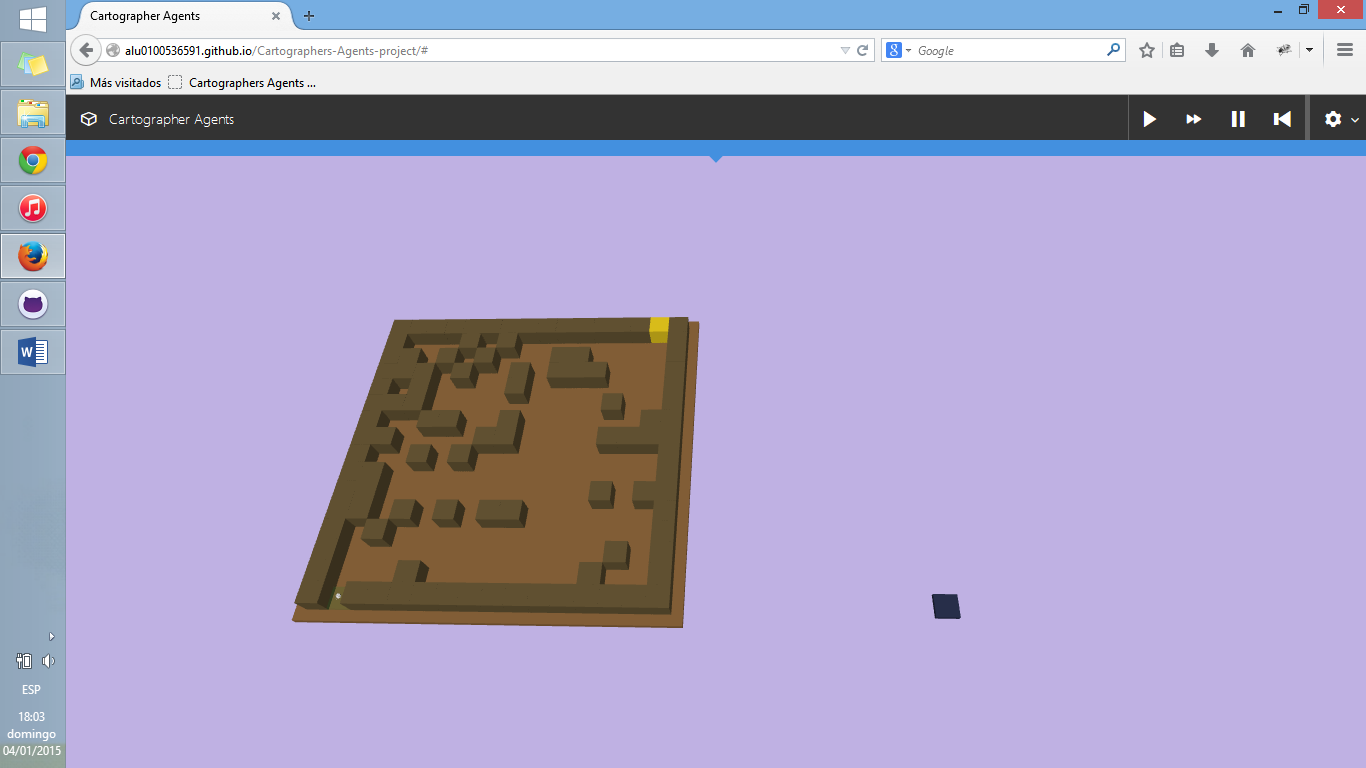


Ilustración 6: Mapeo en el instante 0

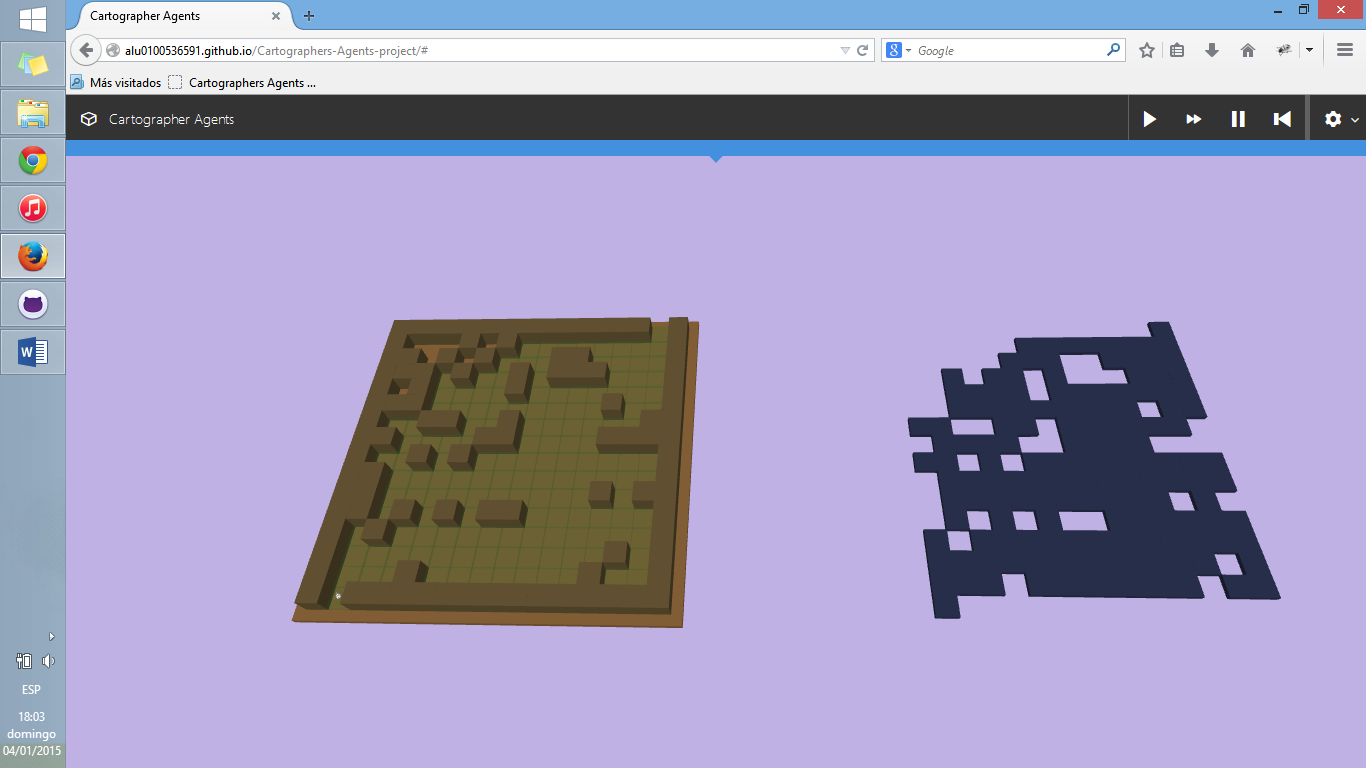
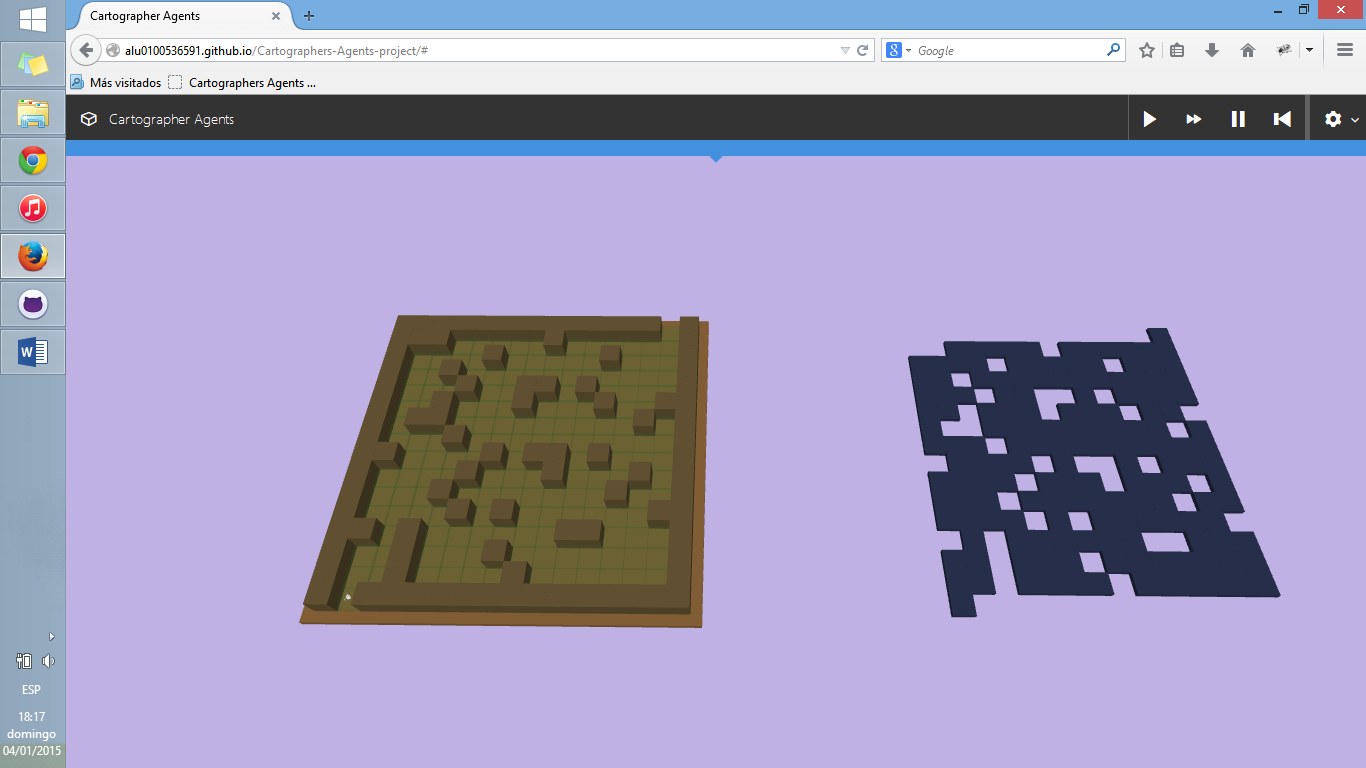
A medida que inciemos la ejecución y la controlemos con los controles descritos anteriormente, el mapa generado por los robots, irá cambiando. Así como las áreas marcadas como visitadas en el mapa original, esto es, sombreadas en color verde.

Ilustración 7: Aplicación web, instante final

# Testing en diversos mapas

## 16x16 – Un robot

Cartografiado: 100%

Mapa 1

## 30x30 – Cinco robots

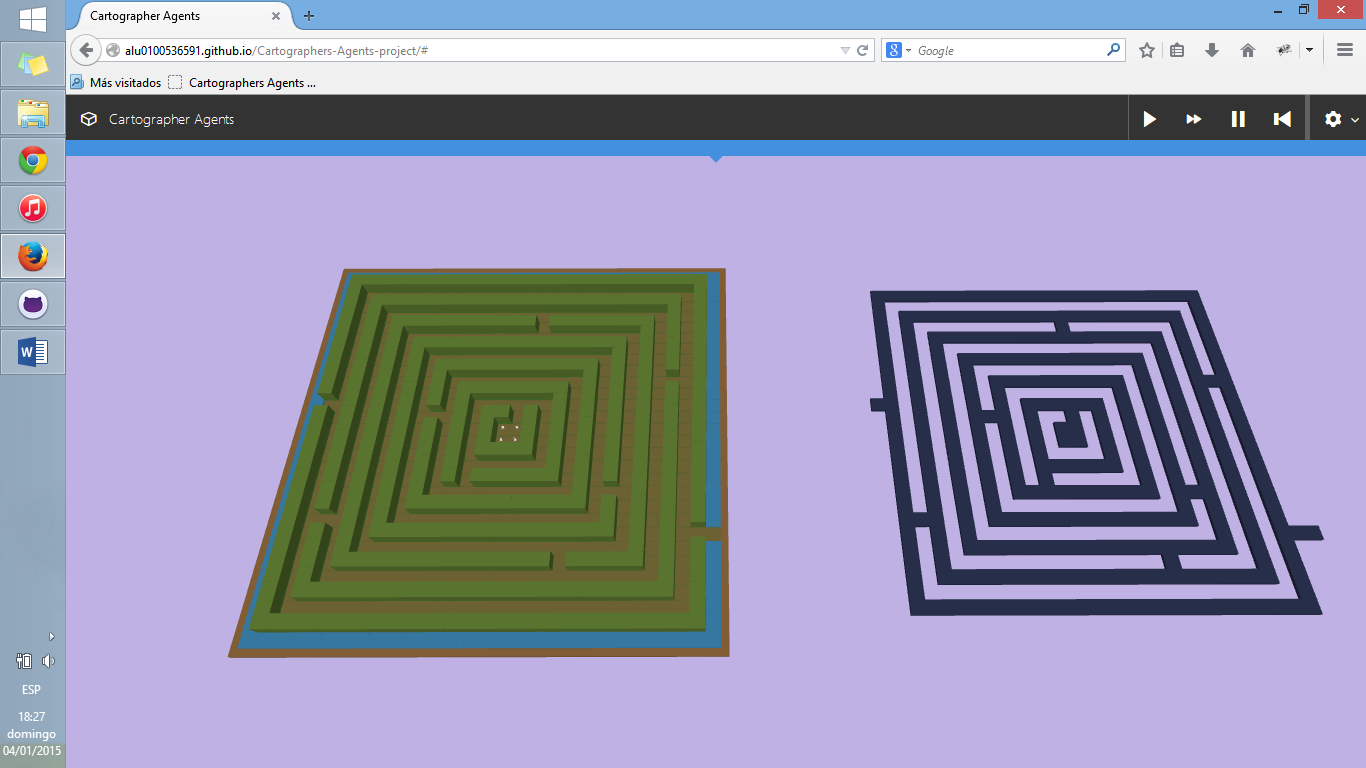
Cartografiado: 100%

Mapa 2

Tras el resultado de los dos primeros experimentos podemos afirmar que la comunicación entre robots funciona. El en caso del primer mapa, el único robot generó todo el mapa, pasando por todas las casillas.

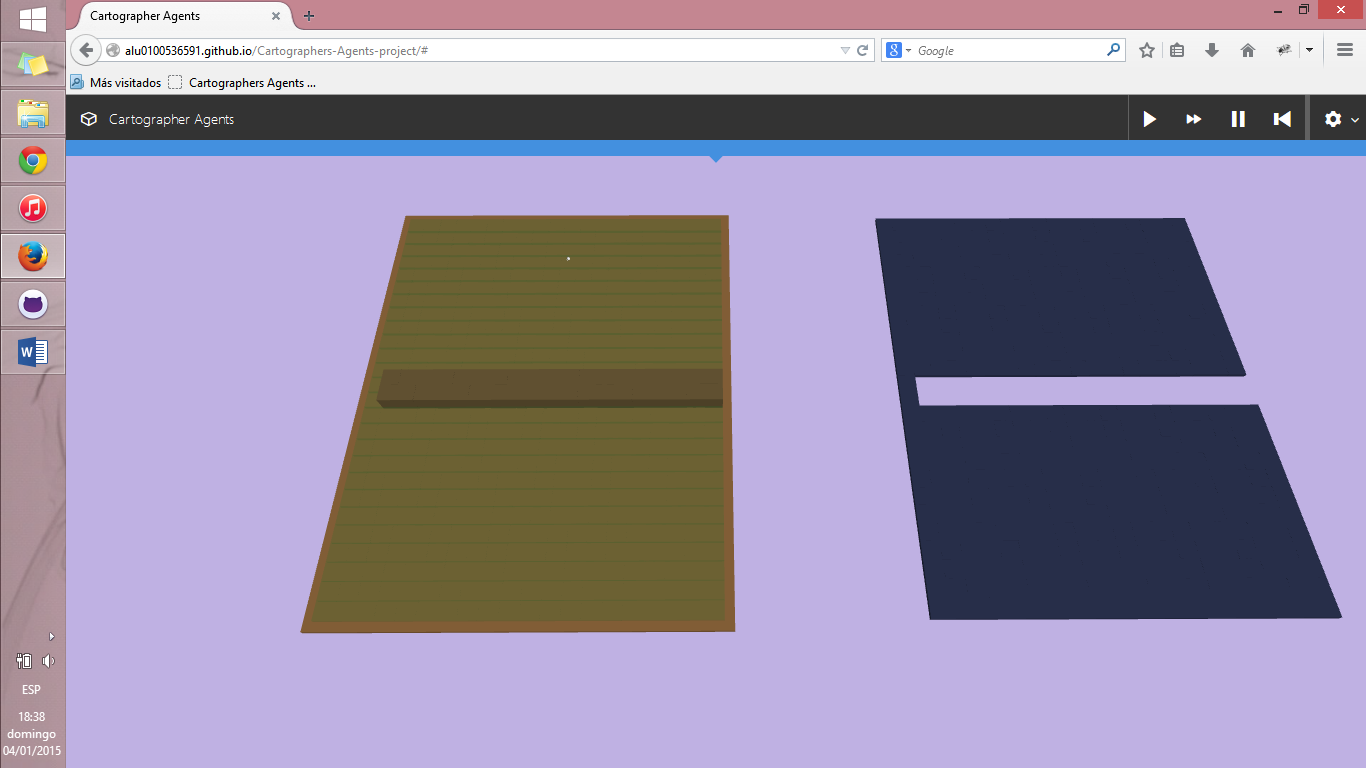
Sin embargo, en el segundo mapa, los robots se han repartido el mapa, lo cual quiere decir que ninguno de ellos ha recorrido todo el mapa, sino que el mapa generado ha sido el resultado de la colaboración del conocimiento individual.

## 30x30 – Cuatro robots

Cartografiado: 100%

Mapa 3

## 26x20 – Un robot

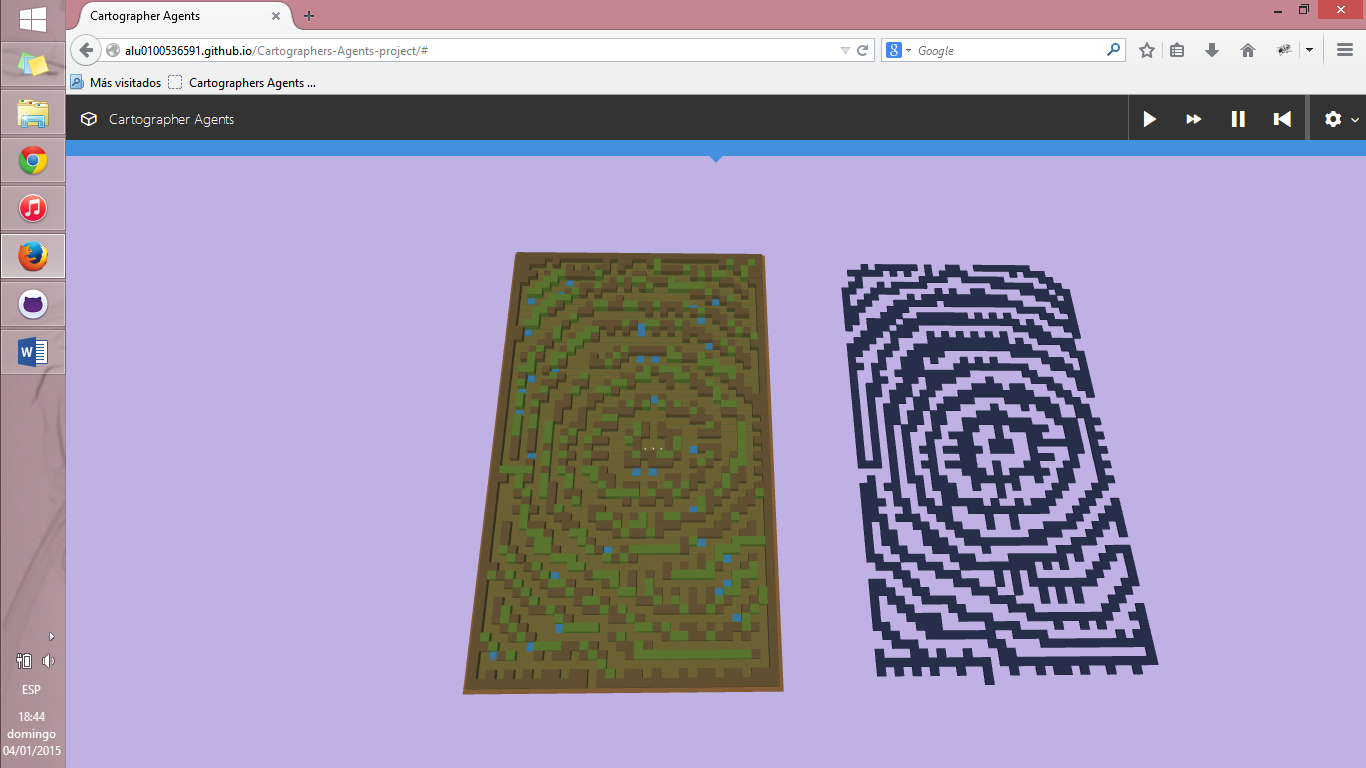
Cartografiado: 100%

Mapa 4

Tras los experimentos tres y cuatro, podemos afirmar que los robots implementados funcionan tanto en espacios laberínticos, con alto porcentaje de obstáculos, así como en espacios abiertos.

Además, para los espacios abiertos, podemos observar que el diseño del movimiento ocasiona que el robot haga un barrido horizontal, explorando de la manera más eficiente posible el mapa.

## 34x58 – Tres robots

Cartografiado: 100%

Mapa 5

## 40x50 – Seis robots

Cartografiado: 90% del mapa total, 100% del mapa accesible.

Mapa 6

Las conclusiones de este último experimento dejan en evidencia dos situaciones dispares. La primera de ellas, donde todo el mapa es accesible, como en el quinto mapa, y la segunda situación donde el agente no puede alcanzar todas las partes del mapa, como en el sexto mapa.

La segunda situación se puede dar debido a dos motivos, tal como podemos ver en el mapa número seis. El primero es que la parte libre sea accesible, es decir, esté rodeada de obstáculos, tal como se ve en las partes de bosque y de agua. La otra situación se hace patente en la esquina superior derecha, donde el robot es el que está acorralado.

Este experimento deja en evidencia que la aplicación funciona con mapas totalmente alcanzables y con mapas que dejan espacios inaccesibles.