Contenido

[Introducción 2](#_Toc7083847)

[Historia de los editores de niveles para juegos 2](#_Toc7083848)

[Editores Oficiales 2](#_Toc7083849)

[Editores para desarrollo 5](#_Toc7083850)

[Editores no Oficiales 5](#_Toc7083851)

[Estado actual de los editores de niveles 5](#_Toc7083852)

[Clasificación de Características 5](#_Toc7083853)

[Análisis de los tipos de editores 5](#_Toc7083854)

[Juegos 2D históricamente y Perspectiva Actual 5](#_Toc7083855)

[Objetivos del juego 5](#_Toc7083856)

[Herramientas e Infraestructura 5](#_Toc7083857)

[Unity3D 6](#_Toc7083858)

[C# 6](#_Toc7083859)

[Visual Studio 7](#_Toc7083860)

[Photoshop CS6 7](#_Toc7083861)

[GitHub 7](#_Toc7083862)

[Trello 8](#_Toc7083863)

[Marco Teórico 9](#_Toc7083864)

[Juego de Plataformas 9](#_Toc7083865)

[Leyes de UX 9](#_Toc7083866)

[Mecánicas 9](#_Toc7083867)

[Implementación 9](#_Toc7083868)

[Serialización 9](#_Toc7083869)

[Validación 10](#_Toc7083870)

[Conclusiones 10](#_Toc7083871)

[Conclusiones Personales 10](#_Toc7083872)

[Lecciones aprendidas 10](#_Toc7083873)

[Trabajo futuro 10](#_Toc7083874)

[Bibliografía 10](#_Toc7083875)

[Referencias 11](#_Toc7083876)

# Introducción

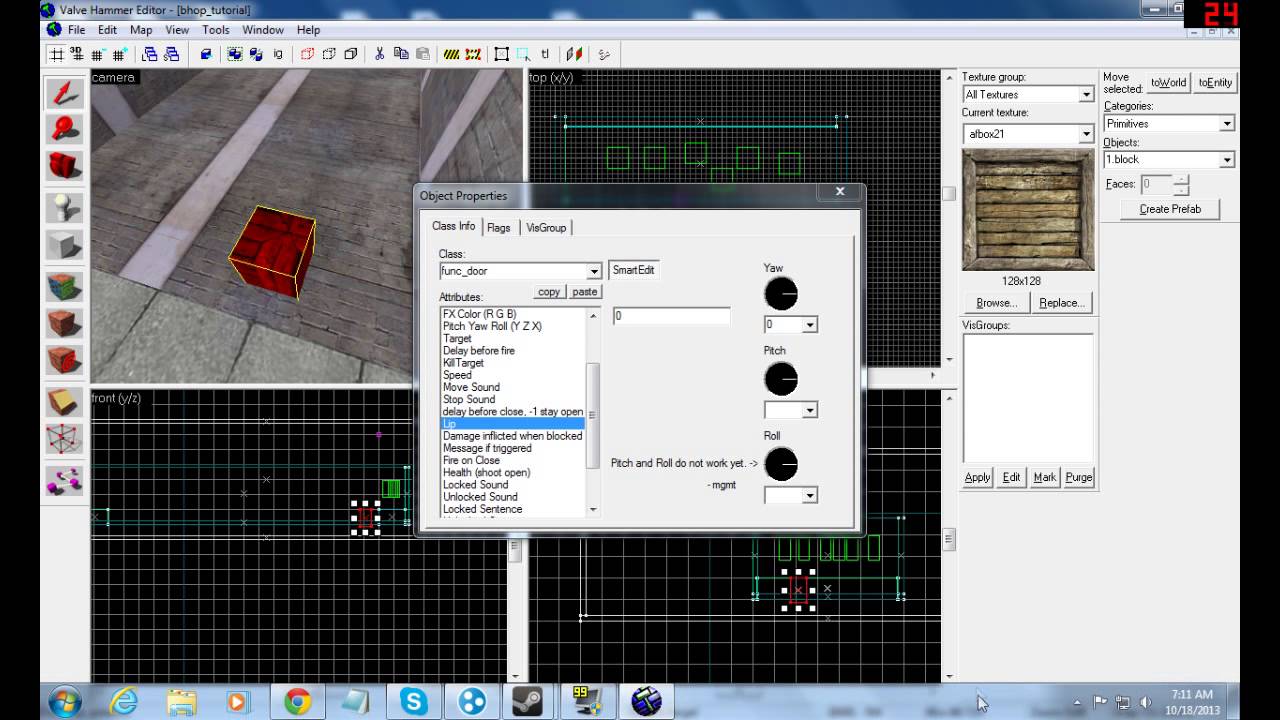
## Historia de los editores de niveles para juegos

Introducción y clasificación personal de los editores de niveles.

### Editores Oficiales

#### Valve Hammer Editor

Valve Hammer Editor es un editor de mapas desarrollado por Valve para el motor de videojuegos **Source**. Este editor es utilizado para el desarrollo de diversos juegos de la compañía como Half-Life 2, Portal y Left 4 Dead, así como por empresas independientes para la creación de mods.



En el editor, al igual que en algunos programas de edición 3D, nos muestra 4 imágenes desde diferentes perspectivas (isométrica, planta, alzado y perfil), lo que nos permite editar los elementos desde todas las perspectivas, y a su vez comprobar los resultados en tiempo real tal y como se verán en el resultado final.

Cuenta con varios botones clave para su rápido acceso, ya que son las herramientas más utilizadas a la hora de crear diversos mapas. Entre ellas se encuentra:

* **Herramienta Selección**: Nos permite seleccionar los diversos elementos de la escena que estamos creando. El problema puede ser el tener que acceder a dicha herramienta cada vez que deseamos seleccionar un objeto, y cambiar de herramienta de nuevo para modificar el elemento.
* **Herramienta de Cámara**: Aunque podamos movernos con las letras AWSD, este movimiento solo se hace en los ejes X e Y, por lo que la herramienta nos ayuda a movernos con mayor libertad con el ratón.
* **Herramienta de Entidades**: Con ella, colocamos diferentes personajes en la escena.
* **Aplicación de Texturas**: Este botón nos muestra una ventana desde la que podemos editar diferentes texturas para nuestros objetos, las cuales se aplican utilizando las dos herramientas siguientes ya sea para aplicarlas a todo el objeto, o a ciertas áreas que el diseñador elija.
* **Herramienta de Clipping**: Con ella, podemos deformar los elementos para crear diferentes formas.

Hay que destacar que los elementos de este editor son cubos, por lo que debemos desarrollar el mapa completo a partir de ellos, aplicándoles las texturas y deformaciones que se deseen por medio de las herramientas mencionadas anteriormente.

#### Super Mario Maker

Mario Maker es un videojuego diseñado por la compañía japonesa de videojuegos Nintendo que ofrece la posibilidad de crear escenarios de la saga Super Mario Bros, publicarlos, y permitir que otros usuarios prueben los mapas creados por los jugadores.

Para ello, el videojuego ofrece un editor donde podemos elegir la versión del videojuego que deseamos utilizar, y una barra de herramientas con 12 elementos que podemos colocar. Esta barra de herramientas ofrece la posibilidad de cambiar la combinación de elementos, la cual se puede editar, liberando así el espacio ocupado en la pantalla.



La parte “jugable” del nivel, se divide por medio de una rejilla, limitando así dónde se pueden colocar los diferentes elementos en el mapa. Además, para ciertos objetos, como contenedores, lo que hace es colocar el contenedor en cuestión en el escenario, y arrastrando dentro de él otro de los elementos, automáticamente se introduce en él, permitiendo que este objeto salga cuando el usuario golpee el cubo.

También, debido a la similitud entre objetos, para evitar tener un número excesivo de elementos en la barra de objetos, permite el cambio de uno por otro simplemente agitando el objeto en cuestión tras ser arrastrado al mapa.

#### Transformice

Transformice es un juego web plataformas donde controlamos a un ratón. Con un tiempo determinado por mapa, debemos conseguir una porción del queso que se encuentra en el mapa y llegar a la ratonera.

Cada mapa es diferente, y prácticamente todos los niveles han sido desarrollados por la comunidad gracias al editor que tiene el juego. A rasgos generales, este editor cuenta con un número finito de objetos (considerados como elementos con físicas utilizados para estorbar, ayudar o decorar), y un número finito de plataformas.

Estas plataformas son escalables, pero no tileables, ya que se trata de una textura cuadrada de alta resolución, por lo que la calidad no varía cuando se escala en ambos ejes, pero al hacerlo solo en uno, se puede observar cómo se distorsiona la textura.



En lo referente al modo de edición de mapas, como se puede apreciar en la imagen, unos radio button nos permiten seleccionar en función de las características, no solo de los elementos, si no también del mapa. Una vez seleccionada una de estas características, se nos mostrará un segundo menú con los elementos cuyas características se corresponden con las seleccionadas. Estos elementos se eligen por su nombre en alguna de las características y por imágenes en otras.

Esto, desde mi punto de vista, reduce la facilidad de editar niveles para los usuarios, ya que teniendo en cuenta que, como ya se ha destacado antes, gran parte de los niveles han sido creados por los usuarios, esto generaliza el uso de algunos elementos, lo que provoca un desuso constante de objetos no reconocibles por los usuarios.

Además, en muchos casos, la mayoría de los usuarios no conocen los nombres concretos de dichos objetos, por lo que encontrar la entidad suele ser tedioso en muchas ocasiones.

A pesar de ello, uno de sus puntos a favor es la colocación de elementos. Una vez seleccionado el objeto en cuestión, aparecerá junto al puntero del ratón, y manteniendo pulsado el botón izquierdo, se podrá colocar y escalar. Cuando soltemos el botón, podremos acceder a sus valores (posición, rotación características específicas), pudiendo editar los valores numéricos introduciendo nuestros propios valores, y permitiendo modificar algunas de las características normales de la entidad.

#### BroForce

#### Battle Block Theater

### Editores para desarrollo

* UbiArt Framework (<https://www.youtube.com/watch?v=WFu1utKAZ18>)
* Tiled Map Editor
* RPG Maker

### Editores no Oficiales

* Minecraft Editor
* The Binding of Isaac Editor
* Bomberman

## Estado actual de los editores de niveles

* Pocos juegos lo ofrecen.
* Juegos AAA lo hacen público para alargar la vida de sus videojuegos.
* Juegos Indie de plataformas 2D con modo multijugador los ofrecen sin problemas.
* Introducción sobre los editores (comunidades de modders), de forma histórica...
* Moddb.com
* Como acceder a nuevos niveles (descargar, directamente…)

### Clasificación de Características

* Clasificar los editores( puedo modificar scripts, texturas? Solo uso los objetos que me dan?).

## Análisis de los tipos de editores

* Justificar por qué elegir estos.

## Juegos 2D históricamente y Perspectiva Actual

Cuando hablamos actualmente de los videojuegos, cualquiera podría pensar en juegos hiperrealistas, con unos gráficos y físicas muy cercanos a la realidad, y por supuesto, en tres dimensiones. Ni esto es así actualmente, ni lo fue cuando los videojuegos comenzaron a desarrollarse.

Partiendo de los primeros videojuegos, es comprensible entender que la potencia tecnológica de la época no permitía el desarrollo de elaborados videojuegos, por lo que estos debían ser, en su mayoría, elementos geométricos planos con capacidad de movimiento, como lo fueron PONG, Asteroids, OXO o Tetris.

# Objetivos del juego

Descripción de los objetivos.

# Herramientas e Infraestructura

* EXPLICAR POR QUÉ ESTAS HERRAMIENTAS SON BUENAS PARA MI PROYECTO.

A continuación, se procede con la descripción de las herramientas software utilizadas para el desarrollo, así como los lenguajes de programación empleados, describiendo la forma en que se ha trabajado con el software y los resultados obtenidos. Las diferentes herramientas son:

## Unity3D

Unity es un motor de videojuegos multiplataformas desarrollado por Unity Technologies. Puede utilizarse en tres plataformas, Windows, OS X y Linux. Recibe la calificación de multiplataforma debido a que es posible crear una aplicación y exportarla a diversas plataformas (WebGL, Android, IOS, Windows…).

Unity surgió tras el desarrollo de un juego denominado GooBall, el cual no obtuvo beneficios. A pesar de ello, sus desarrolladores decidieron potenciar el motor empleado para la creación del juego de forma que cualquiera pudiera utilizarlo. De esta forma, los desarrolladores independientes sin capacidad para crear su propio motor de juego u obtener diversas licencias comenzaron a utilizarlo, otorgando el éxito que hoy día tiene el motor.

Ofrece a los programadores utilizar el lenguaje C#, un lenguaje orientado a objetos fácil de utilizar para aquellos con pocos conocimientos, y además lo hace por medio de scripting basado en Mono, una implementación de código abierto de .NET Framework.

Anteriormente, los lenguajes de programación podrían ser JAVASCRIPT y C#, pero tras el lanzamiento de una de sus versiones, quedó únicamente C#. Además, Unity incluía MonoDevelop como entorno de desarrollo, pero junto con la eliminación de JAVASCRIPT, se eliminó, dando paso a Visual Studio como entorno predeterminado. De esta forma, se facilita el modo Debug, conectando el programa con el motor, de forma que la depuración es mucho más sencilla, el uso de puntos de parada, la obtención de datos durante la ejecución, etc.

Además, por parte de los diseñadores, les ofrece la posibilidad de desarrollar diversos niveles de forma sencilla gracias a sus herramientas, así como a los animadores y artistas, ya que ofrece herramientas de animación para crear las mismas directamente en el software, o exportando animaciones, crear a través de una máquina de estados los diferentes cambios de animación de uno de los personajes. También ofrece múltiples elementos de Interfaz de Usuario para poder utilizarse en diversos aspectos (Menús, Opciones, Registros, Páginas Web…).

Este motor puede también trabajar de forma conjunta con diversos programas utilizados por los artistas, tales como Blender, 3DS Max o Maya entre otros, permitiendo la modificación de objetos editados en alguno de estos programas, actualizándose los cambios automáticamente en el proyecto de Unity que los contiene, y de esta forma, se evita la importación manual tras cada modificación de dichos objetos.

La elección principal de el uso de este programa en el desarrollo de este trabajo, no es solo por las facilidades que ofrece de cara a un desarrollo individual, sino también porque es una de las herramientas más utilizadas en el ámbito del desarrollo de videojuegos a nivel profesional, lo que ofrece una oportunidad de plantear un proyecto, no solo como reto educativo, sino también como reto profesional.

Por otra parte, dado que Unity es una buena herramienta para el desarrollo de ciertas aplicaciones 2D, el hecho de plantear un editor de niveles con este programa ha sido la mejor opción tras descartar diversas plataformas de desarrollo.

## C#

Se trata de un lenguaje de programación orientado a objetos desarrollado por Microsoft como parte de la plataforma .NET similar a JAVA. Deriva de C/C++ incluyendo mejoras derivadas de otros lenguajes.

Es el lenguaje de programación base de las últimas versiones de Unity, lo que hace que su uso sea obligatorio en el desarrollo del proyecto. Además, ofrece múltiples posibilidades a la hora de programar, como el uso de Threads que permitan la ejecución simultanea de bloques de instrucciones, o la gran cantidad de funciones predefinidas que facilitan la interacción con clases objeto, estructuras de datos, y eventos.

Dado que Unity funciona por medio de Scripts, no es necesaria la organización del código por paquetes, sino que cada clase actúa por separado. Claro está, que, en el caso de necesitar una relación entre diversas clases, es posible la creación de scripts que funcionen como una clase objeto, de forma que no sea necesaria la interacción directa con el motor para acceder a ellas.

También debemos destacar la presencia de corrutinas, funciones que nos permiten detener su ejecución, devolviendo el control a Unity. Esto es útil cuando por ejemplo tenemos que esperar un tiempo determinado para continuar la ejecución, o en algunos casos, necesitamos un tiempo de espera para realizar comprobaciones o conexiones.

## Visual Studio

Es un entorno de desarrollo integrado para Windows que soporta lenguajes como C#, C++, JAVA o PHP. Visual Studio permite crear sitios y aplicaciones web y servicios web en cualquier entorno con soporte .NET. De esta forma, se pueden desarrollar aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, móviles… Además, ofrece asistencia en tiempo real según se escribe el código, otorgando la función de autocompletar que muestra posibles códigos a escribir. También, ofrece información acerca de funciones o variables para inspeccionar las definiciones, o ver los errores cometidos a la hora de programar.

Actualmente, se instala por defecto con Unity, por lo que la configuración para el modo *Debug* viene establecida por defecto, lo que facilita la depuración de código activando esta opción. También contiene asistencia de autocompletado de código enlazado a las librerías de Unity, no solo con nombres de funciones, sino con una extensa API que nos informa de la función del método, los parámetros posibles, y en caso de errores por tipo de datos, ofrece la posibilidad de aplicar una solución automática, tanto de conversión de datos, como de modificación de variables.

## Photoshop CS6

Se trata de un editor de gráficos rasterizados desarrollado por Adobe, utilizado principalmente para el retoque de fotografías y gráficos. No solo es el programa más utilizado en la edición de imagen, sino que también ofrece múltiples posibilidades para un desarrollo experto, y facilidad de uso para un nivel usuario.

En el desarrollo de la aplicación ha sido utilizado principalmente para el retoque de diversas texturas tileables, partes de los personajes principales, y creación de textos, ventanas e iconos.

## GitHub

Github es una plataforma de desarrollo colaborativo que permite el alojamiento de proyectos de diversos tipos. Principalmente se utiliza para almacenar programas en desarrollo de diversos lenguajes de forma pública, aunque también ofrece un plan de pago para privatizar los proyectos.

Es una buena forma para compartir proyectos y realizar un desarrollo multiusuario con pocos problemas, ya que ofrece soluciones a la hora de la publicación simultanea de archivos modificados por varios usuarios, de forma que sea posible decidir qué cambios se aplican al fichero.

Además, Github ofrece una plataforma para proyectos web que permite publicar las versiones a modo de uso online, sin necesidad de probarlo en local.

También debemos destacar que GitHub almacena cada subida a la plataforma, lo cual permite que en caso de ser necesario un retroceso a una versión anterior a la que actualmente se encuentra en la nube, podremos hacerlo con una simple restauración, sin eliminar los cambios futuros a dicho punto.

Github trabaja con ramas, es decir, cuando realizamos la primera subida, todos los datos se enlazan a una rama (generalmente la denominada *master*), y si no realizamos ninguna modificación al respecto, cualquier subida permanecerá en ella. Pero es posible la creación de diversas ramas para el desarrollo, lo cual permite realizar modificaciones de fragmentos del proyecto a partir de un punto de la rama principal, de forma que cualquier modificación en cualquiera de las ramas no modifique ninguna de las otras.

Así pues, con esta herramienta, es posible realizar un trabajo paralelo sin verse afectado por los errores del desarrollo actual de otra rama, hasta que los desarrolladores crean conveniente y fusionen la rama secundaria con la principal, uniendo el proyecto de nuevo.

## Trello

Trello es una aplicación de administración de proyectos que utiliza el sistema Kanban para el registro de actividades con tarjetas. Además, permite organizar tareas, crear listas, adjuntar archivos, gestionar tiempos y usuarios dedicados a tareas…

Cuenta con una versión web y una aplicación móvil lo que facilita el acceso desde cualquier lugar a los tablones, así como la gestión de las posibilidades anteriormente mencionadas.

Ha sido utilizado en este proyecto con el fin de mantener un desarrollo organizado, un control de dificultad/tiempo en el avance de las diversas tareas, y un lugar donde poder almacenar las diferentes ideas surgidas a lo largo del desarrollo de la aplicación.

Trello se organiza en tableros. Un tablero puede contener listas con títulos, a las cuales añadimos tarjetas. Son estas tarjetas donde realmente se especifican las tareas, ya que en ellas se puede especificar un título, una descripción genérica de la tarea, y diversas opciones más, entre las que encontramos:

* **Etiquetas**: Permiten organizar tareas por la función que especifiquemos a dichas etiquetas (desarrollo UI, seriealización, diseño de assets…).
* **Miembros**: Con esta opción, es posible añadir diferentes usuarios autorizados en el tablero a las tareas, de forma que se controle quién se encarga de cada parte del desarrollo.
* **Checklist**: Es útil para especificar objetivos de la tarea, de forma que una vez completado dicho objetivo, se marque como terminado, y podamos observar el porcentaje desarrollado de la tarea.
* **Vencimiento**: Este campo nos permite establecer un deadline para la tarea, y así controlar el tiempo empleado en ella, comprobar si la planificación del tiempo ha sido correcta, demasiado extensa o por el contrario demasiado breve.

En este proyecto, la planificación se realiza en diversas listas. La primera de ellas es el denominado *cajón desastre*, en el cual se encuentran ideas para futuras implementaciones, errores a solucionar en un futuro, etc. Después encontramos las listas *por hacer*, orientadas a un objetivo concreto (desarrollo del editor, de las mecánicas, de assets…), que contienen tareas primordiales que aún no se han comenzado y que se deben realizar cuanto antes.

Tras estas listas, tenemos *haciendo*, una lista específica para las tareas que están en desarrollo. Es importante destacar que a ella deben destinarse las tareas en el momento en que comienza su desarrollo, tanto para controlar lo que se está haciendo, como para que en caso de la existencia de otros usuarios, sepan que esa tarea ya no puede tocarse.

Finalmente, las listas de *terminado*, que se dividen por mes, de forma que también se mantenga un control, no solo sobre el tiempo de desarrollo de una tarea, sino también del grueso de trabajo finalizado de forma mensual.

# Marco Teórico

Aquí va la descripción. Explicar qué es lo que quiero que haga, por qué poner las cosas donde están, y así posteriormente en el apartado de implementación, se explica cómo lo he solucionado. También hablar del tipo de assets.

La aplicación desarrollada como TFG, consta de dos partes principales: un juego simple con mecánicas básicas, y un editor de niveles acorde al juego.

La idea principal es el desarrollo de un editor de niveles para juegos, pero debido a las múltiples posibilidades que ofrece dicha idea, se enfocó dicho editor en un sencillo juego de plataformas con diversas mecánicas simples que permitan un extenso desarrollo de ambas partes.

Para ello, hablaremos a continuación de las dos partes:

## Juego de Plataformas

El videojuego elegido debía ser en 2D, ya que, a nivel de usuario, es más sencillo y entretenido la creación de niveles de este tipo que cualquiera 3D, ya que la idea de trabajar con tres ejes, múltiples puntos de vista y complejas configuraciones de mecánicas pueden hacer tan tediosa la edición de mapas hasta el punto de obviar la posibilidad de utilizar el editor.

En lo referente al tipo de videojuego, se eligió la temática de plataformas debido a las múltiples posibilidades que ofrece en cuanto a la diversidad de niveles, ya que al introducir las mecánicas (de las cuales se habla más adelante), el usuario puede ser capaz de crear un juego con objetivos completamente diferentes a los que en un principio se plantearon.

Concretamente, los objetivos de este juego principal son los siguientes. Se plantea un personaje principal con capacidad para utilizar diversas armas, lo cual amplía en gran medida la jugabilidad (ataque físico, a distancia...). Este personaje se mueve por un mundo repleto de enemigos de diversos tipos, y su objetivo es alcanzar la meta. Para ello, deberá desplazarse lateralmente por diferentes niveles, utilizando su capacidad de salto y sus habilidades para, o bien derrotar a los enemigos que encuentre por el camino, o simplemente llegar al final.

A lo largo del juego, el personaje se encontrará con diferentes puertas que le llevarán a otros niveles, de forma que el recorrido no sea únicamente lineal, sino que ese desplazamiento otorgue la sensación de profundidad.

## Leyes de UX

Explicación.

## Mecánicas

Explicación.

# Implementación

## Serialización

Para el desarrollo de la serialización, se decidió utilizar una mezcla entre las posibilidades que ofrece Unity, C# y código de uso público. Principalmente se basa en JSON Helper**1**, una clase desarrollada por *box head productions*, que permite la serialización de múltiples datos transformándolos a formato JSON.

Utilizando esta clase, es posible almacenar conjuntos de datos del mismo tipo en diversos archivos, los cuales se almacenan en un directorio elegido por el usuario, con nombres predefinidos en función del tipo de objeto que engloba.

Así pues, es necesario crear una clase objeto por cada conjunto que almacena los datos necesarios para la carga del nivel. De esta forma, al guardar los elementos del nivel, se insertan nuevos objetos en una lista del tipo de dicho objeto, y esta se escribe en un fichero en formato JSON.

Para la carga de datos, solo necesitaremos deserializar los datos, utilizando la clase mencionada al principio. Para ello, se recogen todos los ficheros almacenados en el directorio elegido por el usuario, y uno por uno, se instancian prefabs con las modificaciones pertinentes, es decir, los datos guardados.

## Máquina de Estados

# Validación

Imágenes del juego, y como se ha distribuido y se ha comprobado que funciona.

# Conclusiones

## Conclusiones Personales

Redacción.

## Lecciones aprendidas

Redacción.

## Trabajo futuro

Redacción.

# Bibliografía

<https://developer.valvesoftware.com/wiki/Valve_Hammer_Editor>

<https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjE-qL__-DfAhUpyYUKHU_dC7kQjRx6BAgBEAU&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3Dy03H7ktBocw&psig=AOvVaw3kg4T0azD_ldyc37ZQkQSs&ust=1547133614437812>

<https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwiu4p_EjuHfAhUxzoUKHTQNDmYQjRx6BAgBEAU&url=https%3A%2F%2Fwww.digitaltrends.com%2Fgame-reviews%2Fsuper-mario-maker-review%2F&psig=AOvVaw3cPVwMzijCNUn6frvSqzwE&ust=1547137519274921>

<https://es.wikipedia.org/wiki/Unity_(motor_de_juego)>

<https://visualstudio.microsoft.com/es/vs/features/ide/>

<https://es.wikipedia.org/wiki/Microsoft_Visual_Studio>

<https://hippogames.itch.io/fantasyheroes>

# Referencias

1. <http://www.boxheadproductions.com.au/deserializing-top-level-arrays-in-json-with-unity/>