Universidad Zaragoza
1542

Proyecto Fin de Carrera

Videojuego educativo para la enseñanza de polinomios a alumnos de 3º de la E.S.O.

Autor

Adam Barreiro Costa

Director y ponente

Director: José María Falcó Boudet
Ponente: José Merseguer Hernáiz

Escuela de Ingeniería y Arquitectura
2013/2014

A mi madre y a mi padre, por todo el cariño y la comprensión que me han dado. Sin ellos no podría haber llegado tan lejos.

A mis hermanos, por todos esos momentos juntos, y por haber sido una fuente de apoyo e inspiración durante todos estos años.

A mis amigos, por las incontables alegrías y todos los momentos inolvidables que he podido vivir a su lado.

Videojuego educativo para la enseñanza de polinomios a alumnos de 3º de la E.S.O.

Resumen

Pese a que en la actualidad el sector del ocio digital goza de una amplia gama de videojuegos educativos, la mayoría de software de este tipo se focaliza exclusivamente en un rango muy limitado de edades, ya que generalmente ha sido diseñado para niños pequeños, con contenidos didácticos que no suelen adentrarse en el temario de niveles más avanzados como la educación secundaria o el bachillerato.

Por otra parte, nos encontramos en un punto en el que el concepto de gamificación (o ludificación) ha cobrado una relevante importancia en el mundo educativo. Cada vez son más estudios los que demuestran que aprender jugando ayuda a asimilar mejor e interesar más al alumno por el aprendizaje.

Con el presente proyecto se pretende acercar este concepto a las aulas de educación secundaria, desarrollando un videojuego educativo accesible vía navegador web, que cubra todo el temario sobre polinomios correspondiente al tercer curso del programa educativo actual.

Para su elaboración se han usado tecnologías web modernas como HTML5 y JavaScript, con el motor para el desarrollo de videojuegos CraftyJS y la librería jQuery que lo complementa en aspectos como eventos de entrada o enriquecimiento de la interfaz de usuario. El back-end se ha construido con node.js para dar servicio a los usuarios y MongoDB para el almacenamiento de información.

El resultado es un videojuego que sumergirá al alumno en un mundo 2D de ciencia ficción repleto de enemigos a los que deberá derrotar mediante la resolución de operaciones con polinomios, desde sumas a cocientes pasando por restas, productos y productos notables. Los enfrentamientos pueden ser contrarreloj o sin límite de tiempo, dependiendo de la pericia del jugador. Si el enemigo no descubre al personaje o éste está escondido, el alumno podrá introducir las soluciones sin la presión que supone tener una cuenta atrás. Si por el contrario el jugador falla una operación en este modo o simplemente es descubierto, las operaciones deberán resolverse antes de que el reloj llegue a cero, o el jugador perderá vida.

El alumno contará con la ayuda de bonificaciones que debe obtener respondiendo correctamente a preguntas teóricas de “verdadero” o ”falso” que le aparecerán al abrir los cofres que hay disponibles a lo largo de los niveles.

Se incluye también un modo multijugador cooperativo, donde un jugador puede jugar con su partida junto a otro alumno que le ayudará a derrotar enemigos y avanzar, con el objetivo de fomentar la interacción entre ellos.

El profesor administrador dispone de un editor de niveles que permite modificar el juego de forma muy flexible, añadiendo más niveles o modificando los existentes en base a las necesidades de la clase, y un panel de gestión de alumnos para controlar a los alumnos que jueguen y ver sus progresos.

Índice general

I

Memoria

1. Introducción

En este capítulo del documento se va a explicar qué es la gamificación, la relación de este concepto con el proyecto y se hará un estudio de la tecnología que existe actualmente para desarrollar videojuegos web. Todos estos elementos servirán para situar al lector en el contexto en el que se ha realizado este proyecto fin de carrera.

1.1. El concepto de gamificación

La gamificación se define como la aplicación de mecánicas propias de juegos y entretenimiento en otros entornos o actividades que no tienen nada que ver con el ocio, con la finalidad de crear un ambiente de esfuerzo, concentración y superación[1]. Estas mecánicas pueden ser, por ejemplo:

- Asignar puntos a una acción cualquiera, es decir, un valor cuantitativo acumulable.
- Asignar un nivel superior al superar cierto umbral de puntos que nos diferencie de otros niveles más bajos.
- Recompensar el hecho de superar ciertos niveles.
- Crear clasificaciones en base a la puntuación.
- Realizar desafíos o retos que den más puntos.

En la actualidad, con el auge de los dispositivos móviles, redes sociales, videoconsolas y ordenadores, este concepto ha cobrado un gran interés para muchos expertos en todo tipo de sectores, desde el tecnológico hasta el educativo, debido a las características beneficiosas que presenta jugar con videojuegos:

- Según un estudio elaborado por el neurocientífico Paul Howard-Jones de la Universidad de Bristol, la producción de dopamina (responsable directa de las conexiones neuronales) se incrementa al jugar con videojuegos[2].

[1] http://www.gamificacion.com/que-es-la-gamificacion
[2] http://www.onlineschools.com/in-focus/gaming-in-classroom

- Al igual que se ha demostrado que leer un libro o practicar deporte afecta a ciertas partes del cerebro, los videojuegos también tienen un impacto significativo en el mismo. Por ejemplo, un estudio elaborado por la Universidad de Rochester concluyó que los aficionados a los juegos tenían la capacidad de tomar decisiones un 25% más rápido que los que no lo hacían[3].

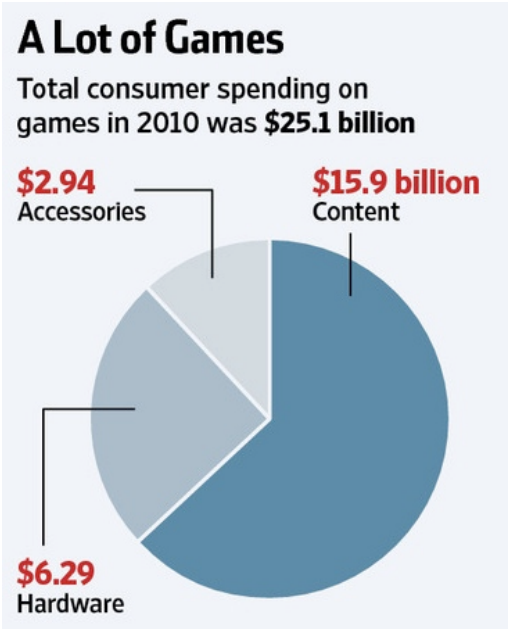


Figura 1: Gráfico que muestra el gasto de los consumidores en juegos

- La Universidad de Michigan realizó el Test de Creatividad de Torrance a 491 alumnos, que consiste hacer un dibujo sobre una hoja de papel con una figura curva, ponerle nombre y escribir una historia sobre él. Los que jugaban a videojuegos sacaron más nota, independientemente de raza, sexo o género al que jugaban[3].

- Alumnos que jugaban a juegos educativos en clase lograron notas más altas en un examen nacional de conocimientos con respecto a los que no lo hicieron[4].

En el Anexo A se presentan diversos ejemplos reales de gamificación en diferentes entornos como empresas o organismos educativos.

[3] http://online.wsj.com/news/articles/SB10001424052970203458604577263273943183932
[4] http://www.onlineschools.com/in-focus/gaming-in-classroom

1.2. El problema

El objetivo que persigue este proyecto fin de carrera es acercar los videojuegos a una parte del sector educativo concreta, como es la educación secundaria, en la que el concepto de la gamificación no está tan extendido en la actualidad debido a que prácticamente el número de juegos diseñados para el temario que se estudia es inexistente.

Además, el gran error de muchos videojuegos educativos es que dan tanta prioridad y protagonismo a los elementos pedagógicos que muchas veces el hecho de que sea un videojuego queda reducido a un segundo plano, y por lo tanto no divierten tanto como deberían. Con este proyecto también se busca crear un mundo más complejo donde aprender y divertirse sean dos conceptos que convivan de forma equilibrada, para lograr así crear una experiencia de juego enriquecedora en la que se aprenda sin ser consciente de que se está aprendiendo.

El reto que supone acercarse a unas mentes más desarrolladas como las de los adolescentes de la E.S.O. es que van a requerir un videojuego que diste de los juegos educativos infantiles que existen actualmente y que debe asemejarse a los que juegan en sus casas.

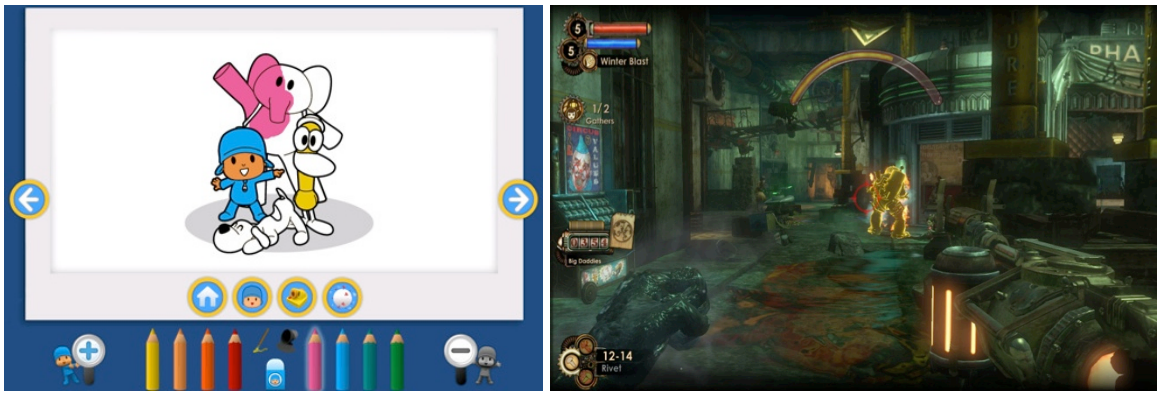


Figura 2: El reto es encontrar el equilibrio entre aprendizaje y diversión

1.3. Contexto tecnológico

Desde que en 2012 fuera propuesto como nuevo estándar para la web por el W3C, HTML5 ha ganado muchísima popularidad a la hora de desarrollar aplicaciones web enriquecidas con elementos multimedia como vídeo, música y gráficos, sustituyendo en algunos casos a las clásicas aplicaciones de escritorio. Esto ha propiciado que tecnologías asociadas

hayan cobrado mayor relevancia, como JavaScript, y otras completamente nuevas hayan surgido, como WebSockets.

JavaScript es un lenguaje de programación que desde su invención se ha usado para facilitar la navegación al usuario de la web, transformando elementos de la misma con propósitos informativos, mostrando mensajes, saneando entradas en formularios, etc. Con la llegada de HTML5, JavaScript ha ganado en popularidad debido a que ahora puede ser usado para hacer aplicaciones completamente funcionales.

En el mundo de los videojuegos, HTML5 ha supuesto una nueva vía de desarrollo. Para calmar el escepticismo que supuso la entrada de esta tecnología empresas como Mozilla decidieron rehacer videojuegos antiguos usando únicamente HTML5 y JavaScript para mostrar todo su potencial, como es el caso del remake de Doom de la figura 3.

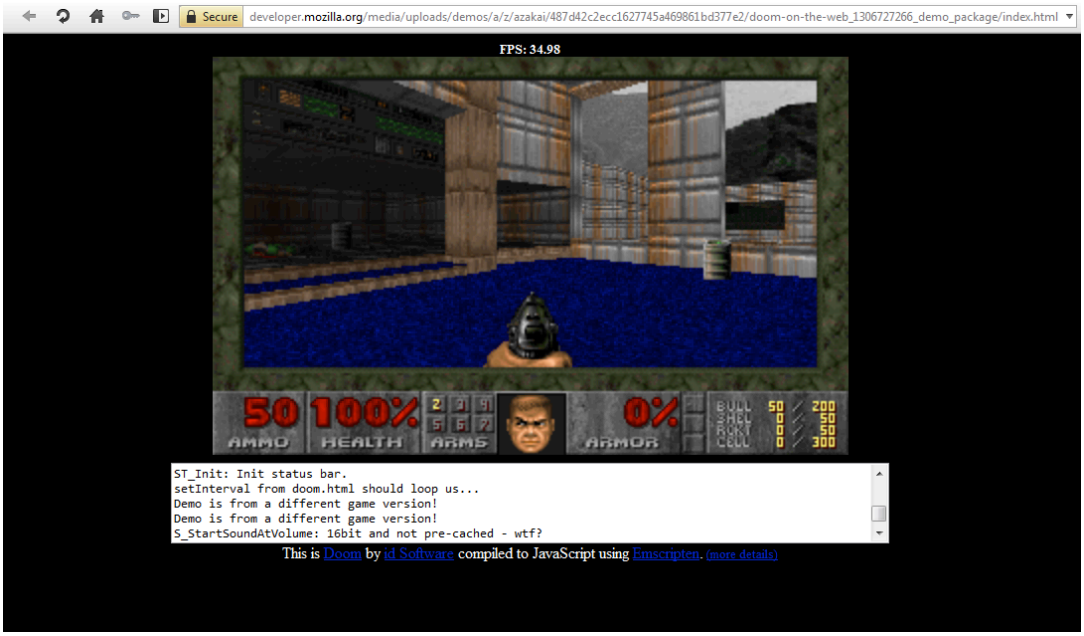


Figura 3: Remake de Doom, de id Software, creado con HTML5

WebSockets es una tecnología completamente nueva. Básicamente se trata del uso de sockets TCP cuya característica principal es que se usan en un navegador web, permitiendo comunicar información con otros navegadores independientemente del contenido que estemos viendo, pudiendo enviar datos en segundo plano en formato JSON (JSON es la notación que usa JavaScript a la hora de crear sus objetos, que consiste en una clave y su valor asociado) y así comunicarnos con otro usuario. Como se verá en esta memoria, esto tiene implicaciones interesantes a la hora de crear juegos multijugador.

2. Estado del arte

A la hora de comenzar un proyecto de este tipo, es importante analizar las herramientas y diferentes tecnologías disponibles enfocadas al desarrollo de videojuegos. Como se verá en el apartado dedicado al análisis de los requisitos (apartado 3.1), al ser un juego para navegador se exploraron únicamente las opciones relacionadas con desarrollo web.

2.1. Tecnologías web

2.1.1. Adobe Flash vs HTML5

Adobe Flash es una herramienta de desarrollo de aplicaciones multimedia que combina el uso de animaciones, audio y vídeo con el lenguaje de programación ActionScript 3.0. Desde que se lanzara la primera versión de Flash en 1996 y la suite de desarrollo en 2008, Flash ha experimentado mejoras y cambios muy significativos.

Las características más importantes de Adobe Flash son:

- Llena el vacío que tienen los navegadores no compatibles con HTML5 para reproducir vídeo, música y audio.
- Compatibilidad con todos los dispositivos, independientemente de su sistema operativo.
- Capacidad de uso de gráficos vectoriales.

Pese a al atractivo que tiene el hecho de permitir usar animaciones y música, se descartó muy rápidamente como opción por bastantes motivos:

Precio

Adobe Flash es una tecnología de una empresa privada. Como tal, el software para desarrollar en esta plataforma es de pago, lo que supone un impedimento para elegirlo como plataforma para el proyecto teniendo en cuenta que una licencia estándar ronda los 250€[5].

[5] http://www.adobe.com/es/products/flash-builder.html

Seguridad

Un aspecto importante al diseñar una aplicación para menores de edad es la seguridad. Adobe Flash ha sufrido varias vulnerabilidades de tipo *0-day*[6] y se ha distribuido bastante malware falseando su instalador.

Compatibilidad

Por otra parte, Adobe Flash requiere instalar un plugin en el ordenador del usuario. A priori no parece un problema grave, pero lo habitual en los centros educativos es que el usuario no tenga derechos de instalación y por ello no pueda instalar Flash. Por ejemplo, en los laboratorios de la Universidad de Zaragoza no se tienen permisos de instalación. Es por esto que se busca una opción que no obligue al usuario a instalar nada en su equipo.

Software condenado al olvido

Tampoco hay que olvidar que muchas empresas como Apple y Google están comenzando a dar la espalda a Adobe Flash por ser una tecnología antigua y obsoleta, para sustituirla por otras tecnologías en auge como HTML5.

Para desarrollar en Adobe Flash es necesario programar en ActionScript, que como todo lenguaje de programación requiere superar una curva de aprendizaje, por tanto tampoco se estimó adecuado comenzar a aprender una tecnología totalmente desconocida para el proyectando justamente en el momento en el que se su uso está empezando a decaer.

Conclusión

Adobe Flash fue una buena opción en el pasado, cuando los navegadores no soportaban audio ni vídeo. Con la llegada de HTML5 esto ha cambiado, y dado que está en plena expansión se consideró una inversión de futuro realizar el videojuego usando esta tecnología junto a JavaScript, del que se hablará en el apartado 2.1.2.

Además, HTML5 tiene bastantes ventajas. Entre ellas destacar que tiene el soporte de toda una comunidad internacional que lo prueba y estandariza como es el W3C, y que para su acceso sólo hay que disponer de un navegador moderno como Google Chrome, Mozilla Firefox, Opera o Safari.

[6] http://arstechnica.com/security/2014/02/adobe-releases-emergency-flash-update-amid-new-zero-day-drive-by-attacks/

2.1.2. Librerías JavaScript

JavaScript ha cobrado mucha importancia en la actualidad, debido a que es el único lenguaje de programación que interpretan los navegadores. Su potencia y la cantidad inmensa de librerías de desarrollo que tiene lo convierten en una opción ideal para crear un videojuego web. A continuación se explican las que se han usado en este proyecto:

jQuery

jQuery facilita enormemente el desarrollo de aplicaciones en JavaScript, al generar un código compacto, limpio y entendible, sobre todo en la parte que concierne a la gestión y control de eventos (por ejemplo, el desplazamiento del ratón o las pulsaciones de teclado).

RequireJS

La naturaleza de JavaScript hace que crear módulos independientes como hacen otros lenguajes como Java sea muy complicado. RequireJS sirve para solventar este problema, haciendo que cada fichero de código sea un módulo independiente que se carga en otros sucesivos módulos bajo demanda.

2.1.3. Motores JavaScript para videojuegos

Tras escoger las librerías a utilizar, se realizó un estudio de los diferentes motores JavaScript para la realización de videojuegos. Estos motores se caracterizan por tener una serie de características definidas y de las que el programador se puede abstraer. Un ejemplo podría ser la inclusión de las físicas, o del manejo de imágenes, vídeo o música. Los motores estudiados son:

ImpactJS

Aunque a priori parece el más potente y fácil de usar de todos los motores analizados, el principal problema que tiene es que es de pago (cuesta $99), por lo que se descartó automáticamente. Como ventajas cabe destacar que incluye un editor de niveles y un framework de código abierto llamado Ejecta que permite incluir animaciones y otros elementos multimedia usando características propias de HTML5 y WebGL, logrando compatibilidad con todo tipo de dispositivos.

Akihabara y Effect Engine

Akihabara es otro motor con licencia GPL2. El inconveniente que tiene es que está orientado a juegos arcade y no tiene tanta potencia como los otros mencionados en este análisis.

Por su parte, Effect Engine es propiedad de una asociación que dejó de dar servicio en 2011. No se vio acertado escoger ninguno de estos dos motores.

CraftyJS

De todos los motores analizados, se escogió CraftyJS porque propone un paradigma de programación que el proyectando nunca había visto, denominado "Orientado a Entidades y Componentes".

Para explicarlo de manera breve y sencilla, uno puede imaginar que un componente es una pegatina que se quita o se pone, como una ficha de Lego. Esta ficha se puede poner sobre otras fichas o directamente sobre una entidad, dotando a esta de todas las propiedades de las fichas que le estamos colocando.

La diferencia con respecto a otros paradigmas como "Orientado a Objetos" es que la herencia está mucho mejor definida y más controlada. En un lenguaje orientado a objetos, si el juego es extenso o tiene gran variedad de clases, el árbol de herencia puede crecer descontroladamente y ser muy complejo de mantener y expandir en un futuro.

Destacar también que CraftyJS gestiona el transcurso del juego mediante escenas, como si fuera un teatro, lo que permite abstraer al programador del típico bucle de juego. Se puede crear una escena de carga, una escena donde esté el juego principal, o una escena final.

Por último, CraftyJS es de código abierto y goza de una extensa comunidad distribuida en grupos de Google, StackOverflow y otros sitios de consulta, de forma que al escoger esta opción se supo que habría un gran apoyo por parte de otros desarrolladores a la hora de necesitar soporte o ejemplos.

2.2. Desarrollo del back-end

Para que el mantenimiento y aplicación del servidor sean sencillos su programación requiere que sea ligero y fácil de configurar, por lo que desde un primer momento implementar un servidor en Java se descartó porque no cumple estas dos cualidades.

Para desarrollar un servidor web en Java, generalmente es necesario desplegar un contenedor de aplicaciones como JBoss o Tomcat y además instalar el servidor Apache para gestionar las peticiones entrantes. Todo esto es un proceso que requiere configurar entornos y variables de forma compleja, y dado que se busca algo sencillo para que un profesor sin conocimientos en informática lo pueda usar y configurar, se descartó.

Finalmente se optó por node.js por los siguientes motivos:

- Se programa en JavaScript, con las implicaciones que tiene: Al utilizar el mismo lenguaje en el lado del servidor y en el lado del cliente el código se puede reutilizar. Por ejemplo, si saneo la entrada en un formulario de login no es necesario volver a codificar lo mismo en otro lenguaje en el lado del servidor, como sí habría que hacer con Java o PHP.
- Es una tecnología moderna, que rompe con el modo de pensar síncrono para pasar al asíncrono. node.js por tanto es concurrente, él se encarga de administrar sus peticiones.
- Tiene una cantidad enorme de módulos instalables de forma muy sencilla que potencian y facilitan la creación de un servidor. Prácticamente con muy pocas líneas de código se puede tener un servidor completamente funcional.

2.3. Almacenamiento de la información

Se estudió la opción de usar una base de datos relacional o una no relacional. En sistemas que requieren una alta escalabilidad horizontal las relacionales son menos aptas porque hay una alta probabilidad de que haya dependencias entre tablas debido a que la mayoría de implementaciones en SQL aceptan el operador JOIN, haciendo que no se puedan separar entre varios nodos de ninguna forma.

Por el contrario las no relacionales son altamente escalables porque no tienen este operador. A cambio la consistencia de la información suele recaer más en el programador que en la propia base de datos.

Dado que el volumen de datos a almacenar y el número de esquemas de datos es demasiado pequeño como para notar diferencias entre una opción y otra, se eligió MongoDB como gestor de bases de datos por otros motivos diferentes:

- La información se guarda en documentos escritos en formato JSON, formato que usa el resto de la aplicación en su totalidad.
- MongoDB tiene soporte en node.js con varios módulos.
- Tiene una amplia comunidad, con foros, cursos formativos y libros.
- Después de haber trabajado exclusivamente con bases de datos relacionales, se vio adecuado escoger esta opción para ampliar conocimientos y porque supone una interesante puerta al futuro.

3. Desarrollo del proyecto

En el siguiente apartado se va a explicar el trabajo realizado a lo largo de todo el proyecto, desde el análisis de los requisitos hasta el plan de pruebas.

3.1. Análisis de los requisitos

A continuación se detallan los requisitos del videojuego desarrollado, extraídos de las reuniones con el director del proyecto:

Funcionales:

- Se centrará en el bloque temático de polinomios correspondiente al tercer curso de la Educación Secundaria Obligatoria.
- Debe enseñar de alguna forma, a elección del proyectando, todo el bloque temático, incluyendo sumas, restas, productos, productos notables y cocientes.
- Se preguntarán partes teóricas.
- Debe ser multiplataforma.
- El juego tendrá modo de un jugador y modo para dos jugadores.
- El juego será para navegador.

No funcionales:

- Es necesario instalar el juego en una máquina para dar servicio o subir el juego a un servicio de Internet.
- El idioma será el castellano.

3.2. Gestión del proyecto

Debido a las numerosas partes de las que consta el proyecto se descartó un modelo de desarrollo en cascada por ser imposible de llevar a cabo sin generar algún problema grave. Se optó por un desarrollo iterativo e incremental donde se diferenciaron varias partes:

- Menú de administración de alumnos
- Editor de niveles
- Motor general del juego: Movimiento, combates, IA, generación de polinomios…
- Expansión del juego con el modo multijugador.

Todas estas iteraciones constan de subtareas como análisis, diseño, implementación, pruebas, mantenimiento y corrección de errores.

Para el control del tiempo se usó un Diagrama de Gantt (figura 4) que muestra el tiempo invertido en cada parte del proyecto. Aproximadamente se dedicó un mes a cada fase incremental del desarrollo, dedicando un total de 1000 horas al proyecto.

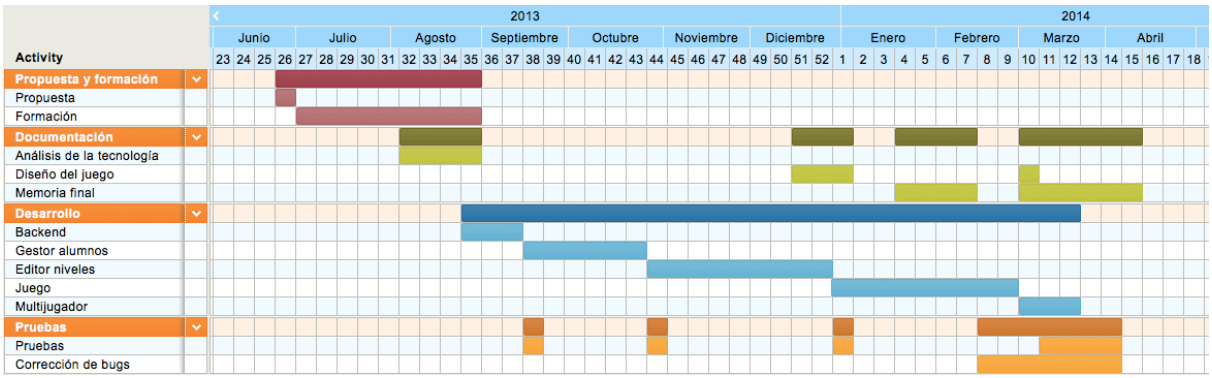


Figura 4: Diagrama de Gantt que muestra el esfuerzo invertido en cada parte del proyecto

En el Anexo C se puede encontrar más información sobre la gestión del proyecto.

3.3. Diseño del juego

En esta sección el documento se van a especificar los elementos del juego y su mecánica, además de explicar el editor de niveles y el gestor de alumnos, herramientas que controlará el profesor.

3.3.1. Descripción breve

El videojuego se llama *Polynomial* (“polinomio” en inglés) debido a la unidad didáctica que comprende. Su estilo es parecido a franquicias conocidas como Metroid de Nintendo o Megaman de Capcom, es decir, es una mezcla entre un juego de plataformas y uno de acción, ambientado en un entorno de ciencia ficción con monstruos que retarán al jugador a realizar operaciones con polinomios, como sumar, restar, multiplicar, resolver productos notables y realizar multiplicaciones o divisiones con fracciones.

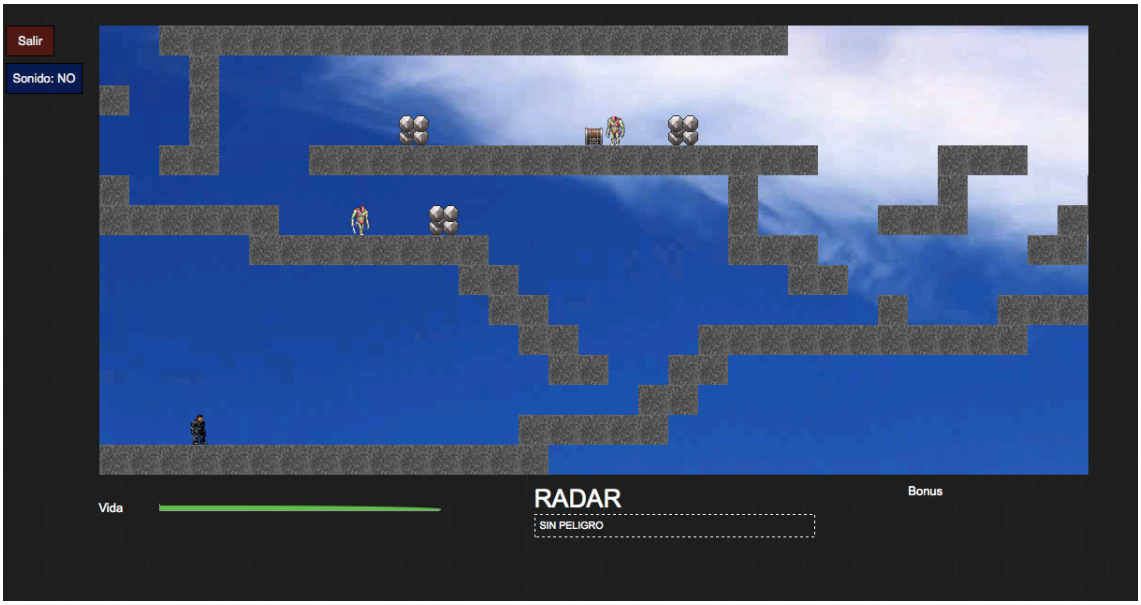


Figura 5: Captura del primer nivel del juego

3.3.2. Argumento

El juego nos pondrá en el papel de un soldado superviviente de una expedición bélica realizada en la luna Europa. Los planes consistían en estudiar los materiales y su posible uso en nuevas armas, pero lo único que encontraron fue un portal del que salieron unas extrañas criaturas que exterminaron a todos los integrantes del grupo.

Estas criaturas son inmunes a cualquier arma normal, salvo una: El poder de los polinomios. El soldado consigue infiltrarse en el portal y llegar al mundo de las criaturas. Ahora el soldado tiene la obligación de llegar hasta el final y destruir a las criaturas para poder salvar la galaxia y a sí mismo.

3.3.3. Gráficos

Los gráficos utilizados en este videojuego han sido obtenidos a través de fuentes Creative Commons y posteriormente editados al gusto con una herramienta de edición fotográfica para cambiarles el tamaño, color y otras propiedades.

Estos gráficos son idénticos tanto para la parte del editor de niveles como la del juego, con un par de importantes diferencias. Dado que es el profesor el que utiliza el editor de niveles, en vez de gráficos de monstruos se han usado los símbolos de las operaciones aritméticas que representan.

Figura 6: Conversión de gráficos en el editor

Además, en el editor de niveles no se usan las hojas de sprites o "*spritesheets*" usadas en el juego, sino imágenes simples. Una *spritesheet* es una imagen dividida en casillas imaginarias de x·y píxeles. Cada casilla representa un fotograma de la animación del personaje que usa esa *spritesheet*. Por ejemplo:

Figura 7: Spritesheet del protagonista

En la figura 7 vemos que la imagen está dividida en 5x8 casillas. La primera fila es el personaje fijo en una posición, la segunda fila se corresponde con el personaje corriendo, etc. De esto se encarga CraftyJS mediante sus funciones internas para animar entidades. Todas estas spritesheets se pueden ver en el Anexo B.

Es importante destacar que todos los gráficos, incluidos los fondos (que se muestran de forma aleatoria en cada nivel) se alojan en el contenido público del servidor (del que se hablará más tarde en el apartado 3.4) de modo que son fácilmente sustituibles si el administrador así lo desea, es decir, puede cambiar la lava roja por agua azul o modificar los enemigos y el terreno.

3.3.4. Música y sonido

El juego cuenta con tres canciones que se reproducen de forma aleatoria a lo largo de los niveles y otras dos para las batallas, así como efectos de sonido como golpes y gritos. Todos estos archivos se han extraído de fuentes Creative Commons y han sido editados con Audacity para cortar ciertas partes y comprimirlos en formato OGG Vorbis.

El juego en todo momento permite desactivar el audio para evitar molestar al usuario si necesita concentrarse o está en un ambiente en el que no procede reproducir ningún sonido, como un aula o una biblioteca.

3.3.5. Diseño de los niveles

Para finalizar el juego se deben superar 10 niveles. Cada dos niveles se introduce una parte del temario relacionado con la unidad de los polinomios, tal como se indica a continuación:

- Nivel 1: Sumas.
- Nivel 3: Restas.
- Nivel 5: Productos.
- Nivel 7: Productos notables.
- Nivel 9: Operaciones con fracciones.

No obstante, dado que no es óptimo para el alumno que un nivel tenga únicamente la parte específica que se ha introducido, se han diseñado los niveles de forma que por ejemplo el nivel 5 introduzca productos pero a su vez siga habiendo enemigos correspondientes a sumas y restas. Así al avanzar por el juego no olvidará lo que ha aprendido.

En cuanto al tamaño, cada nivel tiene 5376x3200 píxeles. Dado que cada *tile* o casilla correspondiente a una entidad de terreno tiene un tamaño de 32x32 píxeles, tenemos que los niveles tienen un tamaño de 168x100.

Para pasar de un nivel a otro, el personaje tiene que alcanzar la zona del sprite “salida”, como se ve en la figura 8.

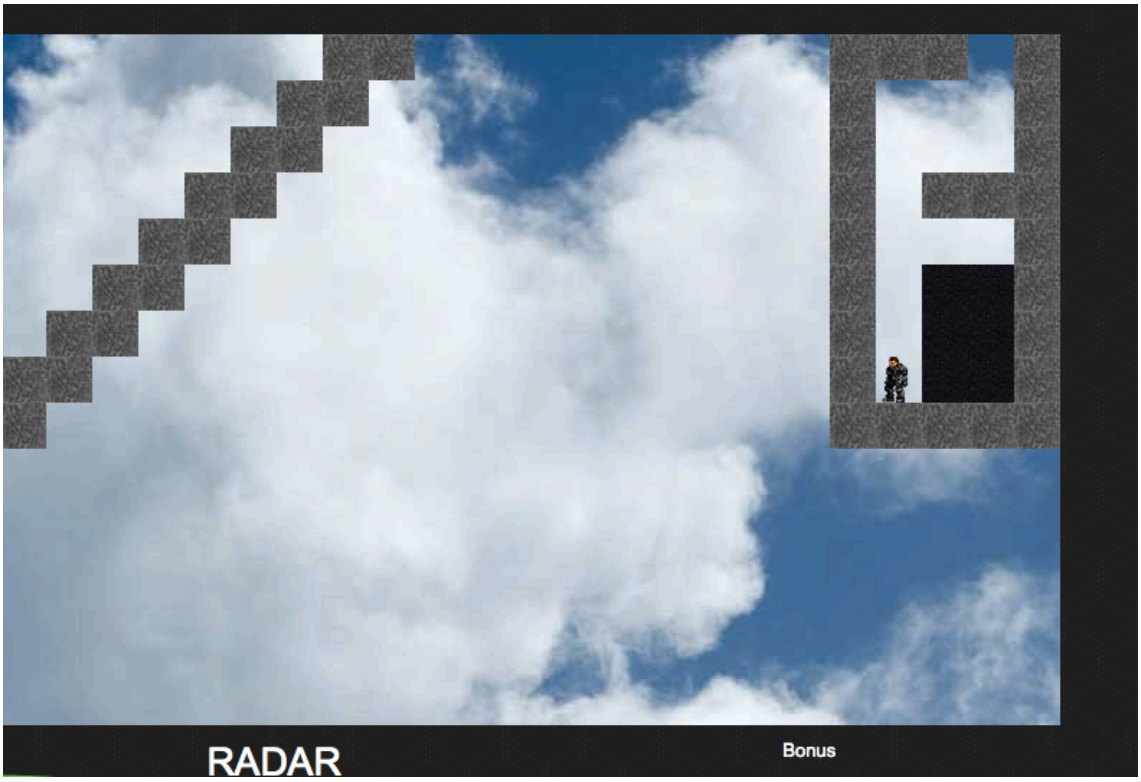


Figura 8: Personaje junto a la salida del nivel

3.1. Tipos de enemigos

El juego dispone de cinco tipos diferentes de enemigos asociados a los cinco tipos de operaciones de los que consta el temario, cada uno de los cuales evaluará únicamente su operación asociada al entablar combate.

Cada enemigo retará al jugador a unas tres o cuatro operaciones. El número es variable debido a que el daño que provoca el personaje es aleatorio. Dado que un enemigo puede evitarse mediante los escondites, se ha forzado que en cada nivel haya un mínimo de tres enemigos con los que es obligatorio entablar un combate al no haber forma posible de esconderse. Esto hace que cada alumno deba realizar de 10 a 15 operaciones como mínimo por nivel.

Como el juego ofrece un editor de niveles, el profesor puede estimar oportuno incrementar el número de enemigos si cree que en su clase hay una gran soltura con las operaciones, o por el contrario disminuirlo si ve que sus alumnos tienen dificultades.

3.2. Mecánica de las batallas

Los enemigos patrullan de manera bastante sencilla por el terreno. Prosiguen su camino hasta encontrar un obstáculo o un precipicio. En ese momento dan la vuelta y patrullan en la otra dirección.

Los combates con los enemigos podrán ser desencadenados por el jugador o por los propios enemigos si le detectan a partir de un radio de distancia prefijado (cuatro casillas de distancia), el enemigo está patrullando en su dirección y además el jugador no está dentro de un escondite o delante de un obstáculo como una plataforma.

Esto hace que haya dos tipos de combates: con y sin cuenta atrás. Si es el jugador el que los provoca (y el enemigo no le detecta en todo momento) la lucha tendrá lugar sin ningún tipo de tiempo límite. El jugador podrá realizar las operaciones una a una hasta la última, que hará que el enemigo sea destruido completamente. En el caso de que el jugador falle una operación, el personaje perderá vida y se pasará al modo de combate con límite de tiempo.

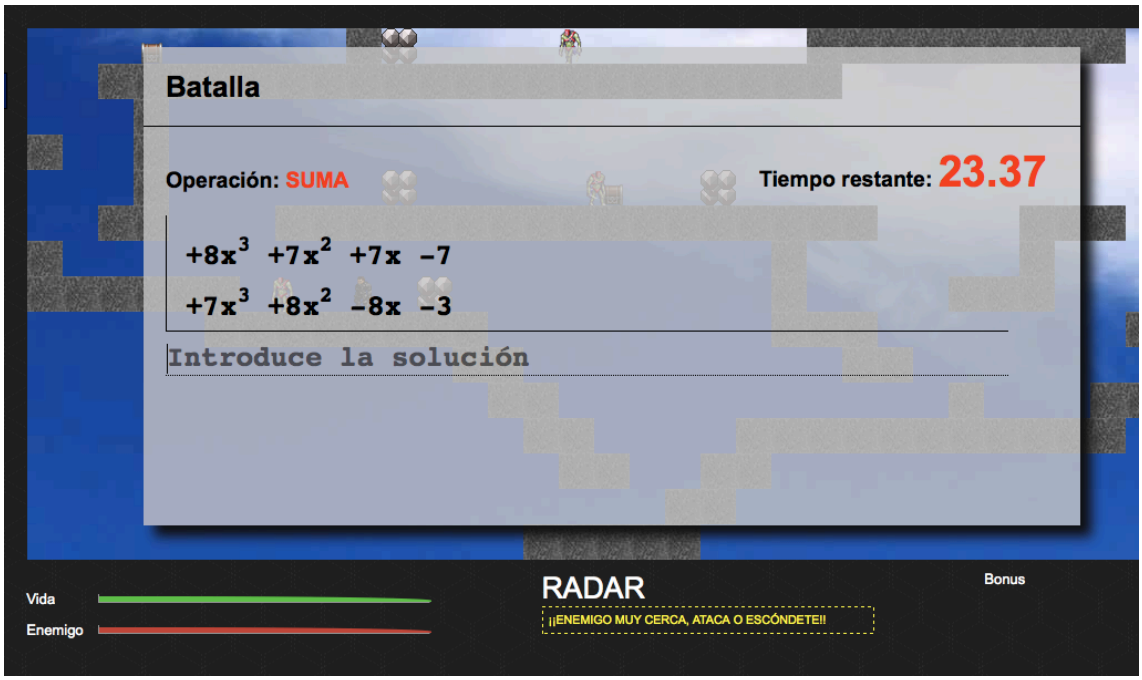


Figura 9: Combate con tiempo

El modo de combate con tiempo se desencadena al fallar una operación o directamente si el enemigo detecta al personaje. Este modo es idéntico, pero el jugador dispondrá de tiempo límite entre operación y operación. Si el tiempo se agota o la operación es incorrecta, el jugador perderá vida.

Los tiempos se han asignado de forma heterogénea para cada operación, ya que como es comprensible es necesario más tiempo para realizar una multiplicación que una suma. Estos tiempos pueden ser modificables por el profesor gracias a un fichero de texto alojado en el servidor, que tiene la siguiente sintaxis:

```
SUMA: T
RESTA: T
MULTIPLICACION: T
NOTABLE: T
DIVISION: T
```

Donde T es el tiempo en milisegundos a asignar a la operación concreta. Por defecto estos tiempos son:

Suma: 29,99 segundos
Resta: 29,99 segundos
Multiplicación: 159,99 segundos
Productos notables: 29,99 segundos
Fracciones: 149,99 segundos

La cantidad de vida perdida por cada golpe que recibe el personaje al fallar una operación varía entre un valor aleatorio situado entre el 10% y el 15% de la vida total del personaje al iniciar la partida.

Si el jugador es derrotado volverá a empezar el mismo nivel en el que fue derrotado desde el principio.

3.3.6. Fosos

A lo largo de los niveles también se han introducido trampas como fosos con lava que dañará al jugador al tocarla, concretamente un 1% cada décima de segundo.

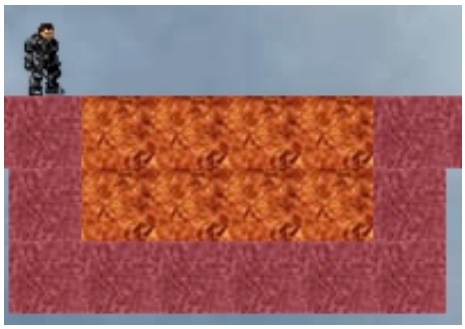


Figura 10: Foso con lava

3.3.7. Mejoras de personaje

A lo largo del juego habrá disponibles unos cofres dispersos con diversas mejoras:

- **Escudo:** Permite fallar operaciones sin perder vida. Dura hasta que se agota, teniendo en cuenta que cada golpe recibido con el escudo hace el doble de daño.
- **Poder:** Las tres próximas operaciones acertadas hacen el doble de daño.
- **Tiempo extra:** Permite tener más tiempo si el jugador es descubierto.
- **Recuperar vida:** Recupera energía de la barra de vida. Este es el objeto más común ya que si no el juego podría ser excesivamente difícil. El montón de vida recuperado es un 70% del total.

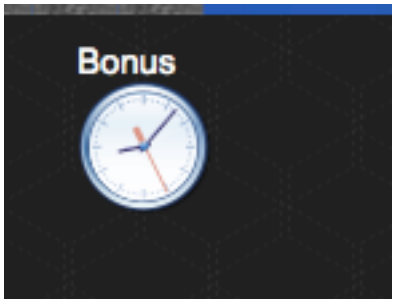


Figura 11: Se ha conseguido el bonus de tiempo extra

Todos los bonus son acumulables entre ellos, es decir, si se obtienen dos bonus de "poder" tendremos seis operaciones con doble daño y no tres. También se pueden combinar varios bonus, llegando a poder tener el escudo, el poder y el tiempo extra a la vez.

Para obtener estas recompensas será necesario responder a varias preguntas teóricas del tema de polinomios, cuya respuesta será verdadero o falso. Las preguntas y sus

correspondientes respuestas estarán escritas en un fichero de texto a razón de pregunta y respuesta por línea, con el formato siguiente:

```
¿(x-3)(x+3) es x^2-9?#V#
¿El literal de 2x es "2"?#F#
```

Es decir, la respuesta vendrá delimitada por los caracteres "#". De este modo el profesor puede editar a su antojo este fichero para añadir o modificar las preguntas existentes. A modo de muestra se han incluido 10 preguntas.

3.3.8. Controles

El jugador puede moverse por el escenario con las flechas del teclado. La de arriba se usa para saltar. Para atacar se usará la tecla A, siempre y cuando el jugador esté al alcance del enemigo. Para abrir tesoros se usará la tecla S.

Todo esto es explicado al jugador al principio del juego con las imágenes que se pueden ver en la figura 12.

Figura 12: Pequeño tutorial antes de comenzar una partida

3.3.9. Multijugador

El multijugador es un modo cooperativo entre dos jugadores, lo que significa que uno de ellos debe comenzar con su partida y el otro se unirá, para apoyarle y ayudarle. Este último jugador no guardará los avances de ningún modo ya que está jugando con la partida de su compañero. Al desconectar, la partida únicamente termina si es el jugador que la comenzó el que cierra el juego.

Los elementos compartidos son los siguientes:

- Enemigos: Tanto el daño que recibe un enemigo, como si el enemigo está muerto.
- Movimiento: Un jugador ve el movimiento de su compañero.
- Pase al siguiente nivel: El primer jugador que llegue a la meta hará que el nivel avance, independientemente de la posición del jugador que haya quedado rezagado.

Los cofres tendrán su instancia en cada partida y no serán comunes.

3.3.10. Parte administrativa

El profesor (o administrador) también tiene un rol importante dentro del juego. A través de un menú al que sólo puede acceder él, puede administrar alumnos, sus grupos de clase y editar sus datos, así como editar los diferentes niveles creados para agregar o quitar elementos como enemigos, terreno, lava o cofres.

Administrador de alumnos

El juego funciona mediante registro y admisión. Esto significa que el profesor no crea a los alumnos, son ellos los que se registran y son asignados a un grupo por un profesor.

Los grupos también pueden ser editados, eliminados y desasignados a un alumno. También permite ver los datos personales de un alumno (correo electrónico, nombre, apellidos y grupo) y editar la contraseña de administrador.

Todos los campos están diseñados de tal forma que no se puedan introducir datos con símbolos extraños, y en el caso de las contraseñas se exige una longitud mínima de 6 caracteres.

Editor de niveles

El editor de niveles se ha diseñado para que sea sencillo de utilizar. Permite crear todos los niveles que uno quiera, permitiendo de esta forma expandir el juego.

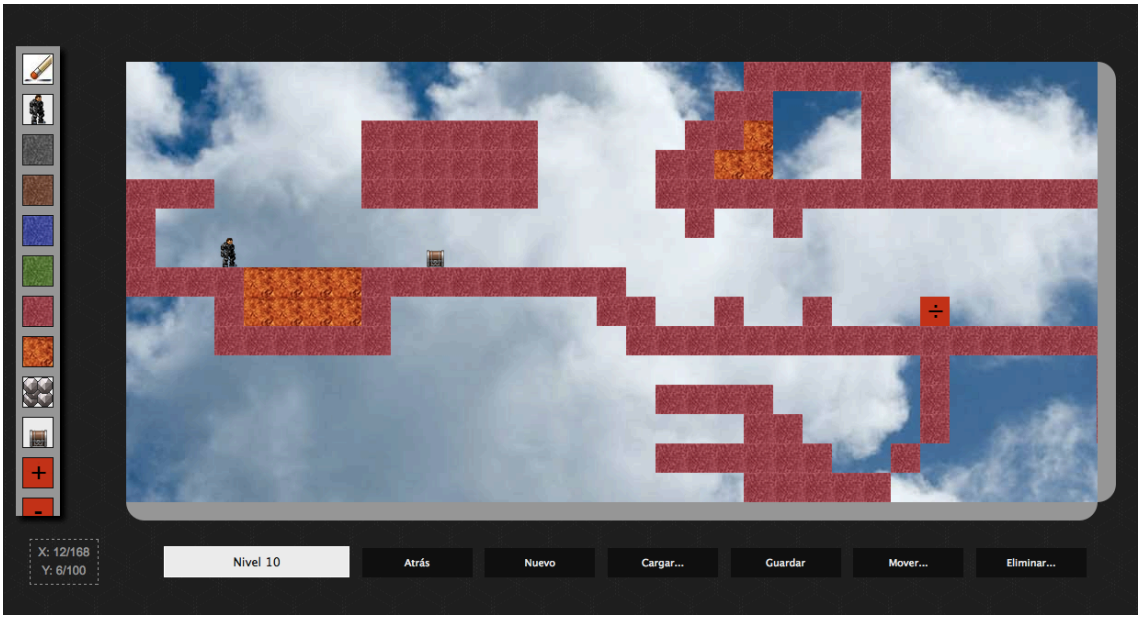


Figura 13: Editor de niveles

Cuando se accede por primera vez, aparece el editor "apagado". Al pulsar "Nuevo nivel" o al cargar uno existente, se activará la paleta de gráficos a la izquierda y aparecerá el nivel en la pantalla central tal como aparece en la figura 13.

Seleccionando un elemento de la paleta de la izquierda el administrador podrá pintar con ese elemento en el escenario. Si escribe encima de algo que ya existe, como un cofre u otro elemento, será sobreescrito. También dispone de una herramienta de borrado.

Para guardar los cambios en ese nivel hay que pulsar "Guardar", y los cambios serán transmitidos al servidor para que actualice el nivel. La opción de "Mover" permite intercambiar la posición de dos niveles diferentes.

El editor también permite eliminar un nivel. Esta opción requiere que el administrador entienda que cuando borra un nivel intermedio, por ejemplo el 5, el resto de niveles se reajustan para que no haya un hueco vacío. Esto es, el 6 pasa a ser el 5, el 7 pasa a ser el 6, etc. Conviene tener esto en cuenta ya que al eliminar un nivel, automáticamente todo el juego se reajusta, por

lo que a ojos del administrador seguirá habiendo un nivel 5, pero no será el que ha borrado, será el 6 que se ha reajustado. Esto se hace para que no haya huecos entre niveles. Para no generar inconsistencias en la base de datos, el nivel que se está editando se cierra.

Las pestañas situadas en los bordes de la pantalla sirven de controles para el desplazamiento del editor. Se puede hacer click o mantener pulsado el ratón. Para más comodidad se añadió la misma funcionalidad pero pulsando W para ir arriba, S para ir abajo, A para ir a la izquierda o D para ir a la derecha. Se escogieron estas teclas para poder pulsarlas con una mano y con la otra usar el ratón.

También se dispone de un pequeño cuadro de información para saber en qué posición del mapa se está dibujando actualmente, así como indicar si el editor está en movimiento.

3.4. Arquitectura del servidor

El sistema diseñado es el que permitirá tanto al alumnado como al profesor acceder a los recursos que hay en él. En el caso de los primeros, al videojuego desarrollado, y en el caso del profesor, al gestor de alumnos y editor de niveles. Para ello se ha seguido una arquitectura cliente-servidor como la que muestra la figura 14.

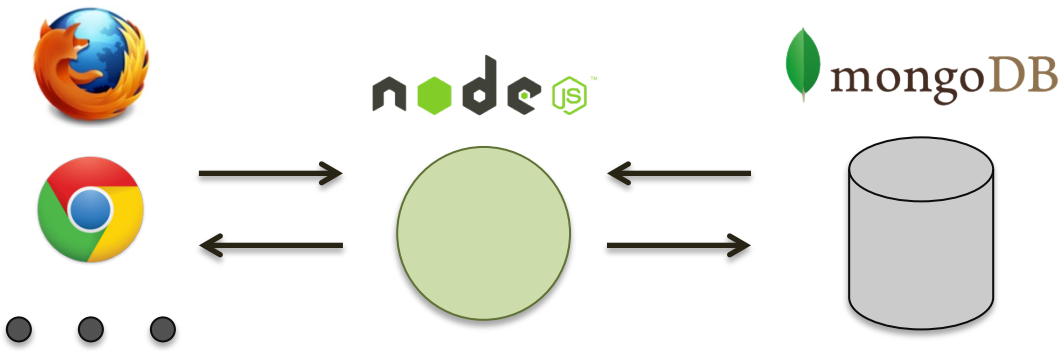


Figura 14: Esquema simplificado de la arquitectura

El esquema completo del servidor se muestra en la figura 15. En él se detallan los componentes y flujos de información que generan entre ellos.

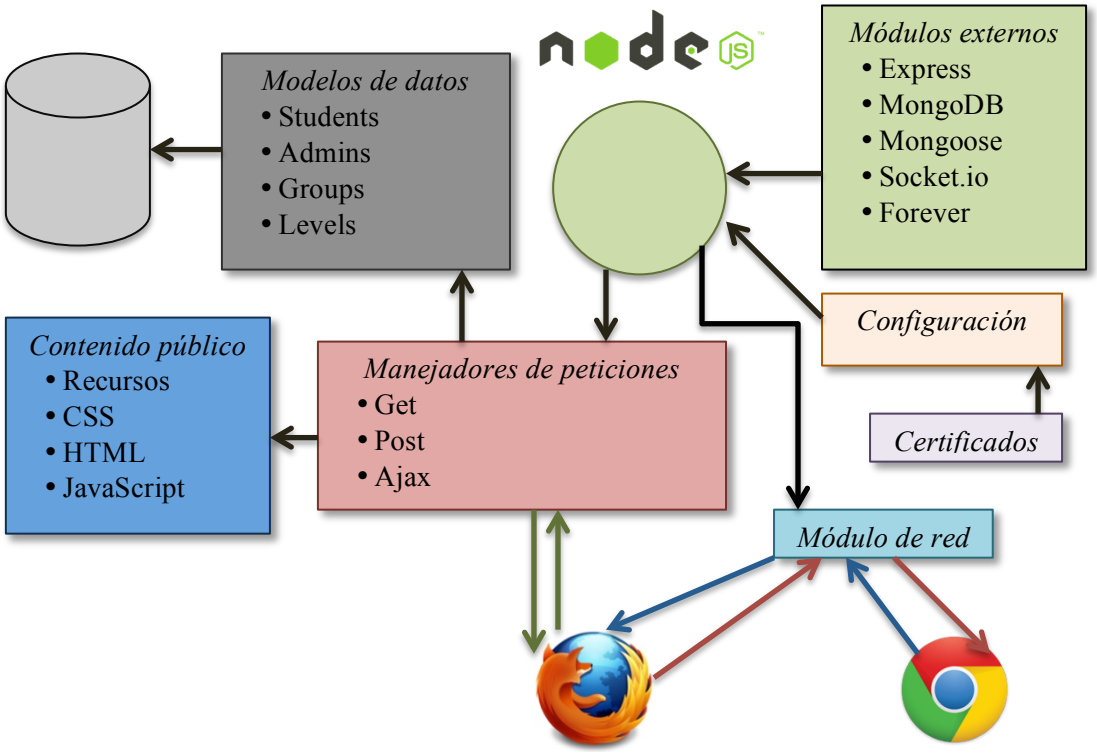


Figura 15: Arquitectura del servidor

3.4.1. Módulos externos

Para la ejecución del servidor son necesarios varios módulos de terceros:

- **Express**: Este módulo permite crear un servidor web de forma concisa y potente, así como establecer las rutas para directorios públicos, doctrinas para gestionar errores y opciones de seguridad (por ejemplo, tokens para evitar ataques de tipo Cross Site Request Forgery), así como un gestor de cookies y sesiones.

- **MongoDB**: Es el driver oficial para una base de datos que usa Mongo. Con este módulo se puede conectar el servidor con la fuente de datos que aloja toda la información mediante la API oficial.

- **Mongoose**: Permite modelar los datos de una forma más sencilla que con las funciones de Mongo. Es decir, permite crear esquemas de datos y manipular los datos conforme a estos esquemas con métodos que son más potentes y eficaces.

- **Socket.io**: Framework que simplifica el uso de WebSockets en el navegador. Es una tecnología del 2011 que permite comunicar dos sistemas como lo haría un socket TCP pero mediante navegador. Socket.io permite la comunicación entre clientes de forma muy sencilla, de modo que se usó para establecer las comunicaciones en el modo multijugador.

- **Forever**: Permite ejecutar indefinidamente el servidor. Su utilidad principal es continuar la ejecución si ocurre una excepción grave, ya que volverá a arrancar y dar servicio.

3.4.2. Modelos de datos

Comprendido por los módulos "Students", "Admins", "Groups" y "Levels", son los encargados de proporcionar funciones para obtener, editar o eliminar la información alojada en la base de datos y posteriormente usarlas en otros módulos como los manejadores de peticiones GET o los manejadores POST.

Los esquemas para cada modelo se definen en notación JSON y son los siguientes:

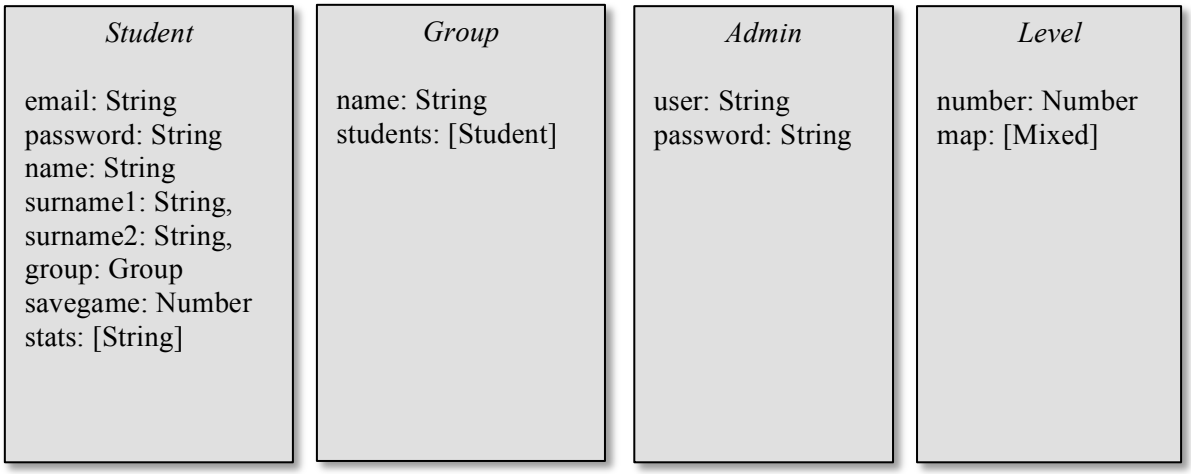


Figura 16: Esquemas del modelo de datos

El tipo Mixed que aparece en el esquema Level es una forma de indicar a Mongoose que el tipo de dato puede ser prácticamente cualquier cosa. Concretamente, "map" es un hashmap clave-valor, donde cada clave es una casilla del mapa y su valor el tipo de entidad.

3.4.3. Manejadores de peticiones

Como su nombre indica, son módulos que incluyen toda la lógica para responder a las peticiones GET, POST y peticiones GET y POST de tipo AJAX. Esto incluye las respuestas a las peticiones de registro, a las operaciones del gestor de alumnos, del editor de niveles y a algunas llamadas AJAX que se hacen durante el juego.

3.4.4. Configuración

Este módulo aloja rutas a ficheros de texto que sirven para modificar parámetros que afectan al comportamiento del servidor, únicamente modificables por el administrador. La lista de ficheros y sus implicaciones son las siguientes:

- **http_port.txt:** Especifica el puerto por el que el servidor va a escuchar las peticiones del protocolo http. En entornos locales no suele ser el 80 como en entornos de producción ya que no se suelen tener permisos para escuchar en ese puerto. Por defecto se ha asignado el 8080.

- **https_port.txt:** Especifica el puerto por el que el servidor va a escuchar las peticiones del protocolo http. Al igual que antes, el puerto en entornos de producción suele ser el 443, pero por defecto se le ha asignado el 8443.

- **database.txt:** Especifica la URL de la base de datos, incluyendo el nombre de usuario, contraseña, nombre de la base de datos a la que se intenta acceder y puerto para escuchar. Por defecto la base de datos se llama “Polynomial”.

- **groups_needed.txt:** Este fichero contiene “true” o “false” y es un elemento usado básicamente para la etapa de desarrollo. Cuando se activa a “true” permite que cualquier usuario que se registre en la aplicación pueda entrar sin tener asignado ningún grupo.

3.4.5. Certificados

Los **certificados** sirven para usar el protocolo HTTPS. Han sido autofirmados, de modo que los navegadores lo catalogarán de no seguro.

3.4.6. Red

El módulo de **red** es el que gestiona toda la lógica de comunicaciones entre un jugador y otro para el modo multijugador. El servidor hace de "router" redirigiendo los datos enviados de un jugador a otro. Esto es así básicamente porque el único elemento que conoce a todos los jugadores es el servidor, es decir, un cliente por sí solo no puede comunicarse con otro cliente porque no sabe quién es ni dónde está. De esta forma, uno crea una partida, notifica al servidor y ésta queda registrada para que otro cliente pueda conectarse a ella.

3.4.7. Contenido público

Consiste en contenido web con sus respectivas hojas de estilos CSS, HTML y contenido JavaScript. Se trata por tanto del videojuego desarrollado, del editor de niveles y del gestor de alumnos, que el servidor se encarga de enviar al usuario mediante la unión de todos los elementos anteriores. Como es otra parte compleja del proyecto, se explicará en las siguientes subsecciones.

Gestor de alumnos

El gestor de alumnos fue el primer elemento del videojuego que se creó, ya que se encarga de gestionar las cuentas que los estudiantes necesitan para jugar y es necesario para el acceso al juego.

Para su desarrollo se utilizó el patrón de diseño MVC (Modelo-Vista-Controlador) que permite separar la lógica y los datos de la visualización que se ofrece al usuario (Figura 17). Por ello toda esta parte del proyecto se construyó con tres módulos en JavaScript, uno que construye la UI (*User Interface*) con jQuery y HTML, el módulo controlador que se encarga de enviar y gestionar todos los eventos y cambios, realizar peticiones AJAX, GET y POST y el módulo restante que es el modelo de datos, del que se ha hablado en la sección anterior, y es el que se encarga de manipular la base de datos y devolver los resultados al usuario.

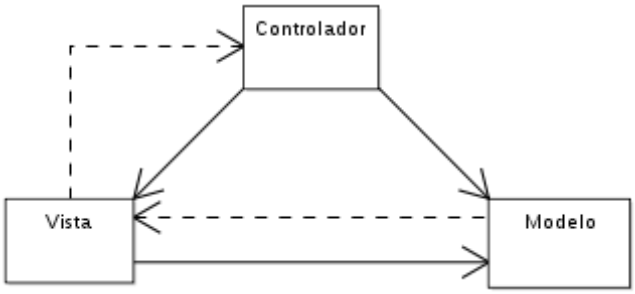


Figura 17: Esquema del patrón MVC

Pese a que existen en la actualidad numerosos frameworks para implementar este patrón, como Jade o AngularJS, no se usó ninguno debido al desconocimiento de ambos. Para cuando el proyectando descubrió de su existencia, esta parte del proyecto ya estaba implementada y probada, de modo que se optó por dejarla así por ser completamente funcional pese a haber "reinventado la rueda".

Editor de niveles

El editor de niveles, a diferencia del gestor de alumnos, no sigue ningún patrón de diseño. Para su desarrollo se mezcló el uso de jQuery y CraftyJS en forma de capas, como muestra la figura 18.

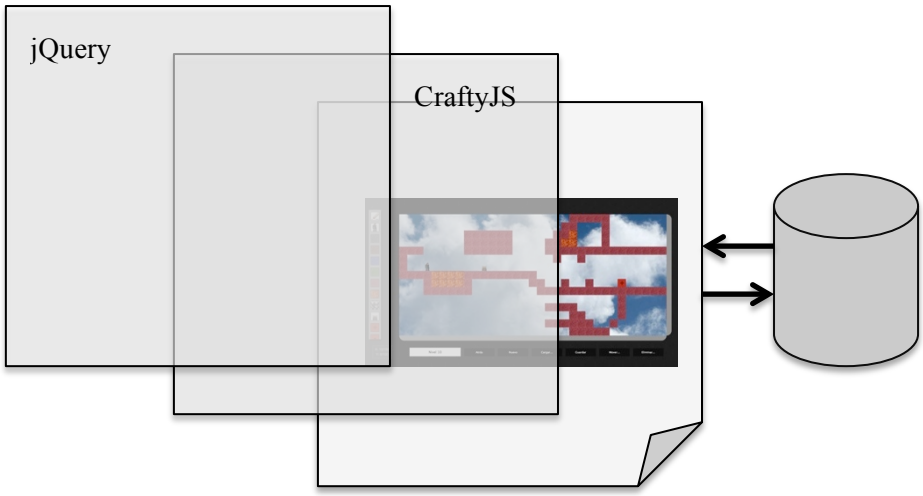


Figura 18: Arquitectura del editor

- La capa que usa jQuery crea la interfaz de usuario y gestiona todos los eventos de ratón y teclado, proporcionando a la capa elaborada con CraftyJS diversos elementos como posiciones de clicks. También gestiona las llamadas AJAX para crear, cargar, mover y eliminar niveles.

- La capa que usa CraftyJS carga todos los gráficos de los componentes y convierte los elementos que recoge jQuery en acciones. Así, convierte los clicks y el desplazamiento del ratón en un "pincel" de entidades, dibuja los niveles completos al cargar o transforma la posición del ratón en coordenadas para alojar una entidad.

3.5. Arquitectura del juego

El juego se compone de varios módulos principales.

3.5.1. Audio

Este módulo, como su nombre indica, se encarga de cargar los sonidos y la música en el motor y de gestionar su reproducción y parada.

3.5.2. Menu

Gestiona la interfaz de usuario y todas las opciones de juego disponibles, dibujando y ocultando menús mediante jQuery. Al iniciar una partida, se llama al módulo Scenes.

3.5.3. Scenes

Este módulo se encarga de generar todas las escenas que componen el juego:

- Escena de carga de juego, donde aparece el tutorial.
- Escena de juego principal, donde se dibuja todo el nivel cargado mediante AJAX de la base de datos.
- Escena para pasar de nivel.
- Escena para cuando muere el personaje y debe volver a empezar.
- Escena de fin de juego.

3.5.4. Multi

Este módulo auxiliar gestiona varias variables requeridas para el multijugador, que se explicaran en el desglose de componentes (página 32).

3.5.5. Constants

Este módulo gestiona constantes como el tamaño de la pantalla en píxeles o el tamaño de pantalla en casillas de 32x32.

3.5.6. Connector y Creator

Contienen la lógica propia del modo multijugador. Como sus nombres indican, uno es el que crea la partida y otro el que se conecta para ayudar al jugador. Se explicarán con más detalle en el desglose de componentes (página 35).

3.5.7. Components

El módulo Components es el que se encarga de llamar y cargar los elementos necesarios para crear todos los componentes del juego. Estos componentes se han organizado de forma jerárquica y se pueden ver en la figura 19.

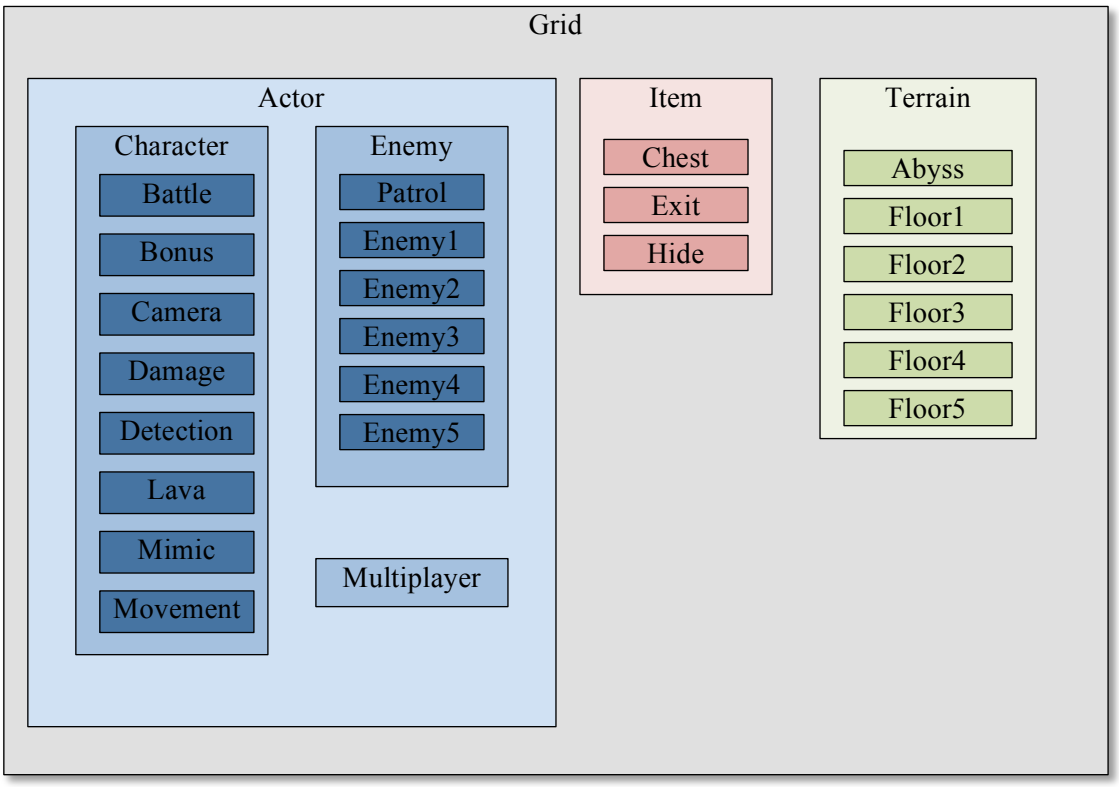


Figura 19: Esquema jerarquizado de componentes

El componente "Grid" es el componente padre y básicamente proporciona un método para dar coordenadas a cada entidad y asignar los componentes incluidos por defecto en CraftyJS, como por ejemplo "Canvas" que permite dibujar sobre elementos canvas, o "2D" que permite asignar a los componentes propiedades de elementos en dos dimensiones, como rotación, movimiento, etc.

Los componentes más significativos del esquema, además de "Grid", son:

Abyss

Este componente representa el terreno que daña al jugador cuando se sitúa encima.

Movement

Asigna eventos de teclado a la entidad asociada para lograr que se mueva. También implementa la habilidad de saltar y un algoritmo de detección de colisiones tanto con entidades inmediatamente encima del personaje como al chocar con cualquier terreno. Para lograr esto hubo que adaptar el algoritmo que usa CraftyJS para generar el efecto de la gravedad y

modificarlo para hacer una "gravedad inversa" que permitiera detectar una colisión con una entidad inmediatamente superior en vez de una colisión con el suelo.

Detection

Implementa el algoritmo de detección que dispara las batallas. Aunque parece que la idea más intuitiva es asignarlo a los enemigos, no es una buena opción por el gran número que hay en cada nivel y por el hecho de que no tiene sentido que un enemigo que está lejos busque al personaje. Lo más eficiente es que sea el personaje quien detecta a los enemigos y dispara el proceso de batalla.

El algoritmo busca una entidad de tipo "Enemy" en un radio de 10 casillas por ambos lados. Si el enemigo está cara al jugador y está a 4 casillas de distancia, se dispara una batalla con tiempo. Si está mirando en dirección contraria, se muestra un aviso en el radar y se permite al usuario desencadenar una batalla. Si el enemigo está más lejos que 4 casillas simplemente se emite un aviso de que hay un enemigo cerca.

También se tiene en cuenta que si el jugador está escondido simplemente hay que comprobar que la distancia al enemigo es menor a 4 casillas.

La figura 20 muestra un resumen conceptual. Como se puede observar, el mensaje que muestra el radar cambia en función de lo lejos que esté el jugador del enemigo, o de si nos encontramos escondidos:

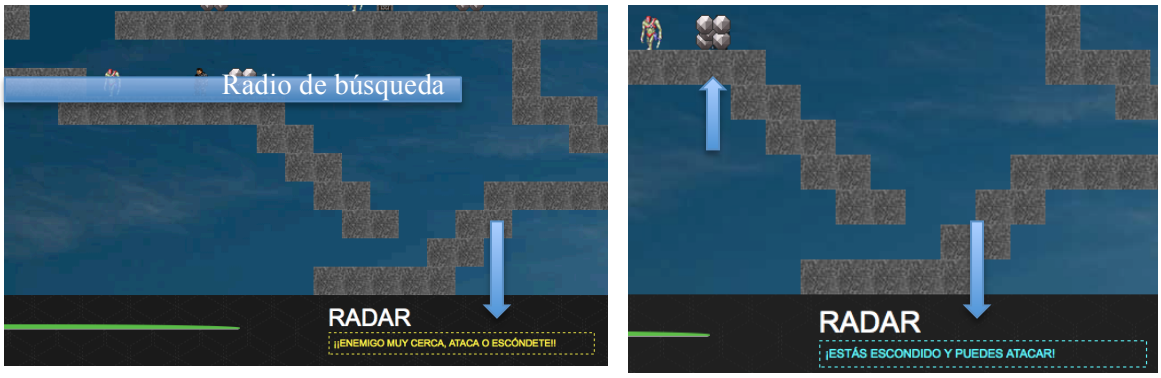


Figura 20: Sistema de detección

Battle

Componente que gestiona todo el sistema de batalla y el más complejo e importante de todos. Para trabajar con polinomios se han usado vectores, donde cada exponente es un índice y su coeficiente es su valor. Por ejemplo, el polinomio $4x^6+9x^4-2x^2+1$ se traduce a:

4	0	9	0	-2	0	1

Cada vector permite guardar un polinomio de grado máximo 6, aunque como se puede ver en el Anexo E se puede ampliar o reducir.

El flujo de trabajo es el siguiente:

- Se determina cuál es la operación correspondiente al enemigo que disparó la batalla.

- Se generan los vectores de polinomios aleatorios junto con la solución asociada a la operación. Estos polinomios tienen las siguientes propiedades:

    - Si es suma o resta, estos vectores tienen grado máximo 6, en los que como máximo aparecen 4 términos con coeficientes del 1 al 9.

    - Si es una multiplicación se genera un polinomio con los parámetros idénticos a los de una suma/resta. El polinomio factor se genera limitando el número de términos a 2 para que la operación no sea costosa de realizar para el alumno y así evitar que se aburra.

    - Si es un producto notable, el vector se genera con sólo dos elementos, ya que son necesarios únicamente dos términos, ax^b y cx^d, de la expresión

      $$(ax^b+cx^d)^2$$

      y su grado. Para la suma por diferencia el proceso es el mismo. Como el juego permite tanto compactar como desarrollar un producto notable, se guarda como solución el vector asociado a $(ax^b+cx^d)^2$ o el vector asociado a su desarrollo según el caso. El tipo de operación se decide aleatoriamente.

  - Para los enemigos de fracciones algebraicas hay tres posibles operaciones: Simplificar, multiplicar o dividir, también aleatorias. Para la operación de simplificación se genera un polinomio divisor con 2 términos al que se le multiplica la solución a introducir, generando un tercer polinomio que será el dividendo. Para multiplicar y dividir se generan cuatro polinomios con 2 términos como máximo.

- Una vez están los vectores creados y alojados en memoria, el flujo se bifurca según el tipo de operación, ya que cada una necesita mostrar un código html diferente. En cualquier caso se diseñaron funciones para convertir un vector a un polinomio en código html. Por ejemplo, el vector anterior generaría el código

```
4x<sup>7</sup>+9x<sup>4</sup>-2x<sup>2</sup>+1
```

- Se activa un temporizador con setInterval en el caso de que sea una batalla con tiempo y se asigna un manejador de evento de teclado para el campo de texto donde se introduce la solución.
- Al pulsar Enter, la solución se capta como una cadena de texto y se pasa por un saneador/parseador que convierte el string a vector, con una serie de llamadas a funciones creadas a tal efecto.
- Una vez convertida la entrada, se comprueba con la solución generada al crear los polinomios en el primer paso, generando daño al enemigo si es correcta o daño al jugador si es incorrecta, mostrando un parpadeo de color verde esmeralda o rojo azulado respectivamente.

Bonus

Este componente asocia un evento de teclado como es la pulsación de la tecla S a la generación de una popup con una pregunta teórica como la de la figura 21, siempre y cuando el jugador esté cerca de un cofre.

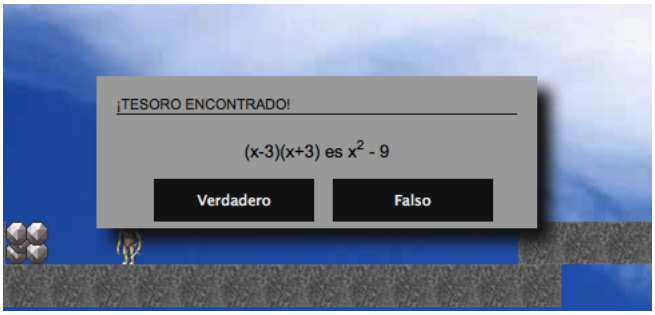


Figura 21: Encontramos un tesoro y aparece una pregunta

Las preguntas se cargan al principio del nivel, cuando se ensamblan todos los componentes, mediante una llamada AJAX que devuelve el contenido del fichero de preguntas especificado en el documento diseño.

Al acertar la pregunta, otra función nos asignará un bonus con una probabilidad del 50% de que sea vida y otro 50% para el resto de bonus (poder, tiempo extra y escudo, explicados en el apartado 3.3.7). Cada cofre tiene una propiedad para detectar si ha sido abierto o no, para que de este modo no se vuelva a abrir y muestre el gráfico de cofre abierto.

Damage y Lava

Estos dos componentes gestionan el daño que recibe el personaje, tanto si es por estar en contacto con una entidad con componente "Abyss" como por recibir un golpe de un enemigo en batalla.

En ambos casos se comprueba si el daño lo recibe el escudo o la vida del personaje, y si llega a cero el juego invoca al módulo de escenas para que muestre la de fin de juego y reinicie el nivel.

En el caso de la lava, se creó un temporizador con la función setInterval() de JavaScript para quitar vida cada diez milisegundos tal como se especificó en el apartado de diseño (página 22). También se crean partículas para dar un efecto visual más realista y así hacer saber al jugador que está perdiendo vida, además de reproducir el sonido asociado al daño.

Exit

Este componente se encarga de invocar a la escena de cambio de nivel, para avanzar al siguiente, en cuanto el personaje toca a la entidad asociada.

Mimic y Multiplayer

Multiplayer es la entidad que se crea en la partida al crear una sesión multijugador, y que reproduce exactamente los movimientos del otro usuario, que son gestionados con el componente "Mimic" asociado a la entidad "Character". Entre sus funciones destacan:

- Enviar datos de movimientos y saltos para que la entidad "Multiplayer" cambie de animación en esos dos casos y cambie la posición.
- Enviar datos de la vida de los enemigos cada vez que uno de los dos jugadores empieza el nivel y cada vez que se hace daño a un enemigo, independientemente de si estamos luchando a la vez con el mismo o no. Esto los mantiene sincronizados en todo momento.
- Enviar el paso de nivel: Cuando uno de los dos jugadores alcanza el final se envía un mensaje al otro jugador para que invoque a la escena de paso de nivel.

3.6. Pruebas, problemas encontrados, alternativas y soluciones

La última etapa del proyecto ha consistido en la ejecución de diversas pruebas que han sacado una serie de problemas que ha habido que resolver. En algunos casos se tuvo que replantear el diseño de algunos componentes por la imposibilidad de solucionar los fallos.

- Al principio se diseñó un teclado bastante restrictivo construido con eventos de teclado en jQuery. La intención era crear un teclado intuitivo para los alumnos que no están acostumbrados a los símbolos "^" y a la introducción de polinomios en un ordenador. El resultado fue un desastre por varios motivos. El primero es que el teclado era tan complejo que el código fuente era muy extenso y complicado, con muchísimos condicionales y variables que controlar. El segundo y más importante, que provocó su descarte, es que era tan sumamente restrictivo que no era en absoluto intuitivo. Se probó con diversas personas ajenas al proyecto y la mayoría tuvo problemas con su manejo. Al final se sustituyó por una caja de texto html.

- La conversión de polinomios a vectores, de vectores a html y la entrada de teclado es tan compleja que prácticamente todos los problemas del proyecto

recayeron en el componente que gestionaba las batallas. Tras numerosos rediseños se logró una solución óptima.

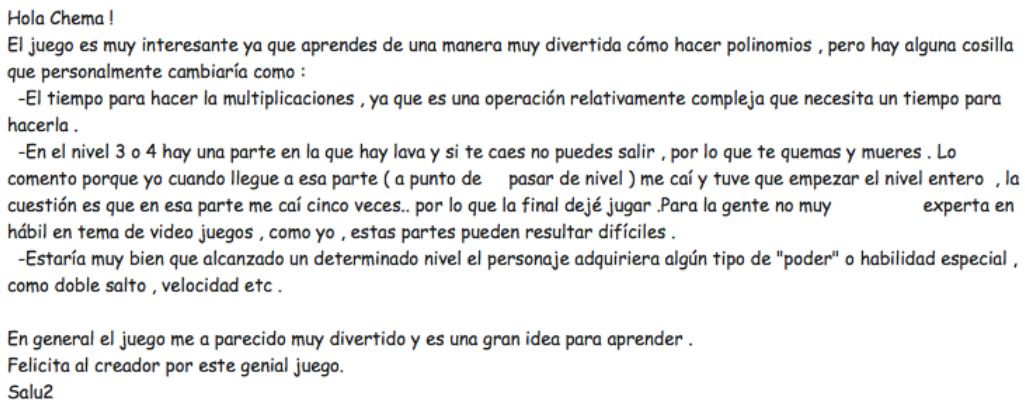
Para el resto de pruebas se envió el juego al director del presente proyecto y a varios compañeros de clase que jugaron varios niveles, reportando incidencias y opiniones. En la mayoría de casos se trataba de errores que el proyectando no había detectado y que fueron posteriormente corregidos. Esto es coherente con las experiencias de desarrollo que han demostrado la necesidad de contar con testers ajenos que desconozcan el funcionamiento interno de la aplicación para no verse influenciados por este hecho.

4. Resultados

4.1. Opiniones

Al someterlo a pruebas con diversas personas se extrajeron las siguientes opiniones:

- Compañeros: Polynomial es un juego muy vistoso y desafiante.
- Alumnos: En la figura 22 se puede ver la opinión de una alumna de 3º de la ESO que probó el juego durante unos días.



Hola Chema !
El juego es muy interesante ya que aprendes de una manera muy divertida cómo hacer polinomios , pero hay alguna cosilla que personalmente cambiaría como :
-El tiempo para hacer la multiplicaciones , ya que es una operación relativamente compleja que necesita un tiempo para hacerla .
-En el nivel 3 o 4 hay una parte en la que hay lava y si te caes no puedes salir , por lo que te quemas y mueres . Lo comento porque yo cuando llegue a esa parte (a punto de pasar de nivel) me caí y tuve que empezar el nivel entero , la cuestión es que en esa parte me caí cinco veces.. por lo que la final dejé jugar .Para la gente no muy experta en hábil en tema de video juegos , como yo , estas partes pueden resultar difíciles .
-Estaría muy bien que alcanzado un determinado nivel el personaje adquiriera algún tipo de "poder" o habilidad especial , como doble salto , velocidad etc .

En general el juego me a parecido muy divertido y es una gran idea para aprender .
Felicita al creador por este genial juego.
Salu2

Figura 22: Opinión de una alumna del director del proyecto

- Proyectando: Debido a la ingente cantidad de pruebas pasadas se ha terminado notando una notable mejoría en el cálculo mental, sobre todo en sumas y restas, y una mayor soltura haciendo operaciones.

4.2. Conclusión

Como resultado del proyecto se ha obtenido un videojuego educativo web completamente funcional, que se puede alojar en cualquier servidor online o en cualquier ordenador personal, con una configuración muy sencilla y con elementos editables como gráficos, música o niveles, convirtiéndolo en un juego versátil y adaptable a las necesidades del alumnado.

5. Líneas futuras

En este proyecto no se incluyeron varios aspectos interesantes que tendría que tener todo juego comercial. Dado que en este caso no lo es y dado que se orienta a un público realmente joven y no se estimó oportuno implementarlos, debido principalmente a que son varios aspectos que aumentarían la duración y complejidad del proyecto notablemente.

5.1. Corrección de trampas del jugador

El juego se sirve del servidor al cliente y éste se ejecuta exclusivamente en el navegador del usuario, lo que propicia esto es que si el usuario tiene conocimientos técnicos suficientes de JavaScript, conoce el motor y conoce cómo está implementado el juego puede hacer trampas manipulando la consola que incluyen los navegadores.

Lo ideal es que cada acción del jugador notificara al servidor de alguna manera, ya sea con WebSockets como se ha visto en anteriores apartados o de otra forma, y que fuera el servidor el que corrigiera este tipo de acciones si detecta que son anómalas, o incluso emitiera algún tipo de aviso o expulsara al usuario del juego.

5.2. Predicción de movimientos

La predicción de movimientos es algo que suelen hacer todos los juegos que tienen un modo para varios jugadores. Enviar cada movimiento de un jugador a otro acaba sobrecargando al servidor que tiene que reenviar cada mensaje con la posición a sus respectivos clientes. Es por esto que lo que se suele hacer es disminuir la carga prediciendo los movimientos del jugador.

5.3. Múltiples idiomas

Dado que el idioma requerido para el juego es el castellano, todos los textos y mensajes que aparecen en el juego están en castellano. Una línea futura interesante sería que el servidor alojara ficheros con múltiples idiomas y que el cliente eligiera qué idioma mostrar.

6. Valoración personal

El desarrollo de este proyecto ha supuesto adentrarme en un sector que me ha apasionado desde pequeño, como es el de los videojuegos. El hecho de que sea un juego educativo ha sido todo un reto, ya que a lo largo de los años he visto con mis propios ojos que los juegos educativos fallan en algo realmente básico: La mayoría no divierten o no tienen tanta profundidad como podría tener un videojuego totalmente comercial. Tuve por tanto que estudiar qué juego me gustaría jugar a mí si yo fuera un alumno real, y lo plasmé de la mejor forma que pude en el diseño presentado en esta memoria, para después desarrollarlo.

El resultado a mi juicio ha sido muy positivo, tanto por los resultados que he observado de toda la gente que ha podido probarlo como por la satisfacción personal de haber aprendido nuevas tecnologías para la elaboración de juegos, como jQuery, JavaScript, node.js y en general toda la tecnología y conceptos mencionados a lo largo de la memoria.

Para terminar, simplemente decir que espero que este juego ayude a todos los niños que lo prueben a aprender divirtiéndose, que siempre ha sido la máxima de este proyecto y mi máxima ilusión.

II

Anexos

A. Ejemplos de gamificación

En este anexo se van a presentar diversos ejemplos de gamificación que complementan los conceptos explicados en el apartado 1 de la memoria.

A.1. Ribbon Hero 2™

Ribbon Hero 2™ es un videojuego elaborado por Microsoft para enseñar a la gente a usar su suite ofimática (figura 23), y se ha convertido en uno de los proyectos más populares del equipo de desarrollo de Microsoft Office.

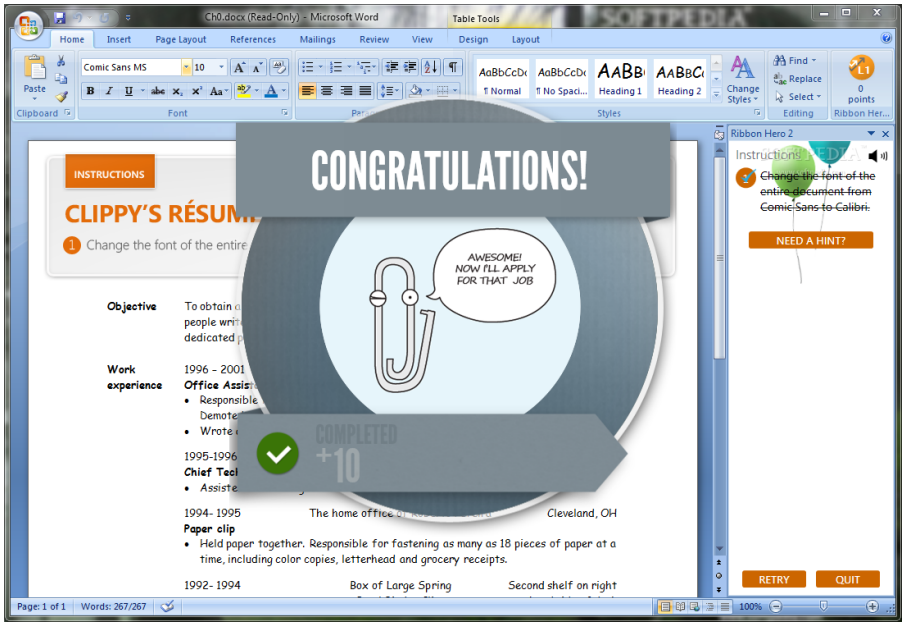


Figura 23: Ribbon Hero 2™ felicita al usuario al progresar correctamente

La historia nos pone en el papel de Clippy, el conocido clip que daba consejos a los usuarios en las versiones de Microsoft Office de 1997. A lo largo del juego deberemos superar diferentes tareas que relacionan períodos históricos con tutoriales de manejo de "Ribbon", la interfaz modular de las nuevas ediciones de Microsoft Office.

A.2. Foldit™

Foldit™ es un videojuego creado por el departamento de informática y el de bioquímica en la Universidad de Washington. En él, los estudiantes deben realizar un plegado de proteínas como si de un puzzle se tratase. Con los resultados, los investigadores de la universidad elaboran diferentes estudios que pueden contribuir a combatir diferentes enfermedades y crear nuevas innovaciones relacionadas con la biología.

Figura 24: Plegado de proteínas en Foldit™

A.3. ESP Game

El reconocimiento de imágenes es una tarea costosa computacionalmente y en muchos casos imposible, de modo que la idea de ESP Game es asignar esa idea a los humanos. Como puede ser una tarea poco interesante, se creó un videojuego que la convierte en algo lúdico.

En 2006 Google compró una licencia de este videojuego para crear su propia versión y así mejorar su servicio de búsqueda.

El juego consiste en etiquetar imágenes por parejas: Un jugador en su casa y el otro jugador, en otra parte del mundo, deben ponerse de acuerdo sin comunicarse en la etiqueta que debe tener la imagen. De este modo, deben etiquetar 15 imágenes en dos minutos y medio.

B. Hojas de sprites

En este anexo se muestran las hojas de sprites de todos los gráficos del juego que por su tamaño no se han podido introducir en la memoria.

Figura 25: Spritesheet de la lava

Figura 26: Spritesheet del personaje

Figura 27: Spritesheet de los cofres

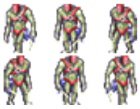


Figura 28: Spritesheet del enemigo sumador

Figura 29: Spritesheet del enemigo restador

Figura 30: Spritesheet del enemigo multiplicador

Figura 31: Spritesheet del enemigo de productos notables

Figura 32: Spritesheet del enemigo divisor

Figura 33: Spritesheet del multijugador

C. Gestión del proyecto

En este anexo se va a profundizar más sobre la planificación y metodología de trabajo aplicadas, así como las herramientas utilizadas durante el desarrollo del proyecto.

C.1. Ciclo de vida

Como se ha explicado en el apartado de gestión del proyecto (página 16), se ha seguido un ciclo de vida iterativo incremental debido fundamentalmente a que el proyecto presenta fases de desarrollo muy diferenciadas. Este sistema permite desarrollar cada parte de forma independiente, probarla y pasar a la siguiente, de forma flexible y adaptable a posibles cambios que sean requeridos por el cliente, en este caso el tutor.

A lo largo de la elaboración del proyecto se ha asignado una duración a cada parte de aproximadamente 30 días, otorgando un margen de maniobra a las partes más complejas, para poder asegurar su corrección y hacer posibles retoques.

C.2. Hitos

Se han fijado tres hitos a lo largo del proyecto:

- Parte administrativa: Gestión y editor de niveles: Diciembre de 2013
- Juego para un jugador: Febrero de 2013
- Juego completo: Abril de 2013

En cada hito el director probó cada una de las funcionalidades entregadas y redactó sus conclusiones, sugerencias y modificaciones a realizar.

C.3. Herramientas de desarrollo

C.3.1. IDEs

Para programar se usó el editor Sublime Text 2 con los siguientes plugins instalados:

- Alignment: Permite insertar y quitar márgenes a varias líneas de código.
- Bracket Highlighter: Señala la correspondencia entre paréntesis, para saber más fácilmente cuál está abierto/cerrado y así evitar errores.
- Docblockr: Permite escribir documentación en las funciones de forma sencilla.
- LiveReload: Permite visualizar los cambios de un fichero JavaScript en el navegador automáticamente al guardar.
- SublimeLinter: Indica errores en el código a medida que se va escribiendo.

C.3.2. Documentación

A lo largo del proyecto se usó tanto la suite ofimática Microsoft Office para redactar la documentación del mismo así como herramientas de diseño gráfico para introducir las diferentes figuras. También se usó Adobe Reader para leer documentación.

C.3.3. Control de versiones

Por ser gratuito y sencillo de utilizar, se usó GitHub como control de versiones, en un repositorio público accesible desde https://github.com/adambarreiro/Polynomial.

C.3.4. Copias de seguridad

Se utilizó Dropbox y Google Drive como servicios para alojar diversas copias de seguridad mensuales del proyecto. No se cuenta GitHub como sistema de copia de seguridad porque únicamente incluye el código del proyecto.

C.3.5. Diseño gráfico y audio

Se utilizó tanto Audacity como Adobe Photoshop en el diseño y modificación de audio e imágenes. El logotipo del juego fue realizado con Adobe Illustrator.

D. Pruebas

En esta sección de explican con más detalle todas las pruebas realizadas para garantizar que el proyecto no contiene errores.

Propósito	Acciones	Salida esperada
Registrarse en la aplicación	Rellenar el formulario de registro	Mensaje OK
Hacer login como administrador	Introducir las credenciales en el formulario de login	Aparece el menú de administrador en el navegador
Hacer login como alumno sin asignar	Introducir las credenciales en el formulario de login	Aparece el formulario de login y aparece un mensaje de error
Crear un grupo	Rellenar el formulario de creación de grupo	Mensaje OK
Editar un grupo	Rellenar el formulario de edición de grupo	Mensaje OK
Eliminar un grupo	Seleccionar el grupo a eliminar	Mensaje OK
Asignar un grupo	Seleccionar el grupo y el alumno en el formulario de asignación	Mensaje OK y aviso de “Alumnos sin asignar” desaparece
Desasignar un grupo	Seleccionar el alumno en el formulario de desasignación	Mensaje OK y vuelve a aparecer el aviso de “Alumnos sin asignar”
Ver alumno	Seleccionar la opción de Ver alumno, seleccionar uno del desplegable	Se visualizan sus datos
Borrar historial	De la prueba anterior, hacer click en el botón de “Borrar historial”	El historial en el desplegable desaparece
Eliminar alumno	Seleccionar el alumno en el formulario de borrado de alumno	Mensaje OK
Nuevo nivel	Seleccionar nuevo nivel en el editor de niveles	Aparece un nivel vacío, se activa la paleta de dibujo, los botones se activan
Guardar nivel	Seleccionar guardar nivel en el editor de niveles	Aparece una popup de aviso
Cargar nivel	Seleccionar cargar nivel en el editor de niveles	El nivel actual se cierra y aparece otro nuevo con el contenido del nivel cargado.
Mover nivel	Seleccionar mover nivel en el editor de niveles	El nivel actual se cierra y los dos niveles se intercambian (comprobar cargando)
Eliminar nivel	Seleccionar eliminar nivel en el editor de niveles	El nivel actual se cierra y el nivel borrado no aparece. La lista de niveles se reordena.
Tesoros	Nos dirigimos a un cofre y pulsamos S	Aparece una popup con una pregunta del fichero de preguntas. La respuesta es la del fichero.
Bonus	Contestamos correctamente la pregunta	Aparece el icono del bonus en la esquina inferior derecha. Si ya teníamos ese bonus no aparece otro icono igual.

D. Pruebas

Batallas con tiempo	Nos dirigimos a un enemigo y dejamos que nos descubra	Aparece el menú de batalla con el temporizador en marcha
Batallas sin tiempo	Nos escondemos y pulsamos A cuando el enemigo esté cerca	Aparece el menú de batalla sin el temporizador
Las sumas funcionan correctamente	Ganar un combate con sumas	El enemigo muere
Las restas funcionan correctamente	Ganar un combate con restas	El enemigo muere
Las multiplicaciones funcionan correctamente	Ganar un combate con multiplicaciones	El enemigo muere
Los productos notables funcionan correctamente	Ganar un combate con productos notables	El enemigo muere
Las divisiones funcionan correctamente	Ganar un combate con divisiones	El enemigo muere
La meta de los niveles es alcanzable	Recorrer los 10 niveles	Se alcanza el final del juego
La lava daña al personaje	Situarse encima de una casilla de lava	La vida comienza a descender
El personaje muere al quedarse sin vida	Perder combates hasta perder toda la vida	El nivel se reinicia
Al empezar el modo multijugador vemos al otro jugador	Seleccionar la opción de multijugador	Aparece un personaje idéntico al nuestro de color verde
La vida de los enemigos desciende si el otro jugador les daña	Entablar combate y esperar a que el otro jugador ataque a los enemigos	La vida del enemigo desciende sin que hagamos nada
Los enemigos muertos no vuelven a aparecer en el multijugador	Morir con un enemigo matado por el otor jugador.	Al reiniciar ese enemigo no está
Al tocar la salida el otro jugador, el nivel termina	Esperar a que el otro jugador alcance el final del nivel	El nivel termina y se pasa al siguiente

E. Guía de desarrollo

Una de las cosas que se tuvo en cuenta a la hora de desarrollar el juego fue estructurar todos sus componentes de forma clara para que en un futuro se pudieran expandir sus funcionalidades o agregar nuevos tipos de operaciones sin que surgieran dificultades. El presente anexo pretende ser una pequeña guía para orientar al desarrollador responsable de la expansión.

E.1. Añadir un nuevo tipo de enemigo

Lo primero que hay que hacer en este caso es crear un nuevo componente para el nuevo enemigo, y alojarlo en la carpeta donde se encuentran los otros componentes de enemigos. Para hacer esto basta con copiar y pegar uno de los existentes para después modificarlo:

Figura 34: Estructura del nuevo enemigo

Como se observa en la figura 34, creamos el componente con el nombre que queramos para el nuevo enemigo, para posteriormente crear una función init. La función init es la que toma CraftyJS por defecto cada vez que se crea una entidad de este tipo.

A esta función le indicamos que requerimos el componente Enemy y su hoja de sprites para que los cargue “spr_enemy1”. Para crear una animación, la definimos con this.reel(). Los parámetros indican en qué casilla empieza el primer sprite, la velocidad y el número de sprites (Casilla 0x0, 300ms de velocidad por los 3 sprites, 100ms cada sprite).

El componente “spr_enemy1” que contiene los gráficos del enemigo lo definimos en Components.js, que precarga todos los gráficos. En este archivo veremos dos vectores de imágenes. Simplemente añadimos el nombre del fichero dentro y modificamos el código de las funciones para incluirlo.

```
var EDITOR_ICONS = ['char.png', 'floor1.jpg', 'floor2.jpg','floor3.jpg','floor4.jpg',
                    'floor5.jpg','abyss.jpg','hide.png', 'chest.png','enemy1.png',
                    'enemy2.png','enemy3.png', 'enemy4.png','enemy5.png', 'exit.jpg'];
var GAME_ICONS = ['char.png', 'floor1.jpg', 'floor2.jpg','floor3.jpg','floor4.jpg',
                  'floor5.jpg','abyss.jpg','hide.png', 'chest.png','enemy1.png',
                  'enemy2.png','enemy3.png', 'enemy4.png','enemy5.png', 'exit.jpg',
                  'multiplayer.png'];
```

Figura 35: Vectores de sprites

Para incluir a este nuevo enemigo en el editor, tendremos que editar drawTile(x,y,type) en “engine.js” y “enemy.js” para incluir nuestro nuevo tipo de entidad en la paleta de pinceles y en el componente padre, respectivamente.

Por último, queda integrar la lógica de las batallas en el módulo “battle.js”, que queda a cargo del desarrollador. Si la operación tiene que ver con polinomios hay un amplio surtido de parsers de polinomios y términos, así como generadores de vectores y código html asociado a los mismos.

E.2. Añadir efectos y música

Para realizar esto únicamente hay que modificar el módulo Audio. Introducimos el nombre del sonido en el vector de ficheros de audio y creamos una función para reproducirlo en el caso de ser un efecto o lo introducimos en la función playLevel() si es una canción.

E.3. Añadir terrenos

El proceso es prácticamente idéntico al de crear un enemigo. Hay que crear el componente en la carpeta de terrenos y hacer las modificaciones explicadas en el apartado E.1.

E.4. Modificación de parámetros

A continuación se detallan una serie de parámetros que afectan a la mecánica del juego y cómo modificarlos:

- Poder de curación de los bonus de vida, así como el número de bonus que se entregan por cofre: Editar "bonus.js" y cambiar las propiedades que aparecen en la zona privada del módulo.

- Cambiar el número de términos máximos de un polinomio, coeficiente máximo, probabilidad de que un término sea +/- y probabilidad de término nulo: Editar propiedades de "battle.js".

- Cambiar la cantidad de daño recibido por golpe tanto para escudo como para daño normal: Editar propiedades de "damage.js".

- Cambiar radio de detección: Editar la variable siguiente en "detection.js":

```
var detectionRange = Crafty.map.search({_x: this.x-320, _y: this.y, _w: 640, _h: 32}, true);
```

  Donde 640 son 10 casillas, ya que cada casilla son 32 píxeles (como se explicó en la página 19) y el radio busca en dos direcciones, luego: $10 \cdot 32 \cdot 2 = 640$. Al buscar en 2 direcciones tenemos que centrarlo, por eso a la propiedad de entrada _x es la mitad, 320.

- Cambiar el aspecto de las saplicaduras de lava: Cambiar los parámetros de la propiedad "drawLava" en "lava.js". No se recomienda incrementar el número de partículas por la sobrecarga de CPU que genera.

- Cambiar la gravedad o la altura del salto: Editar la constante de gravedad proporcionada a this.gravity().gravityConst() y la propiedad "JUMPY" en "movement.js".

- Cambiar la velocidad de patrulla de los enemigos: Pese a que no se recomienda ya que una velocidad excesiva puede hacer que el algoritmo de patrulla funcione

incorrectamente, basta con modificar la propiedad this._patrolSpeed de "patrol.js".

6.1.1. Modificación del gestor de alumnos

Para implementar nuevas opciones es suficiente con incluir la nueva lógica del gestor en la carpeta de contenido público, en los ficheros "view.js" y "controller.js". Puede ser necesario editar y ampliar el modelo de datos para adaptarse a las nuevas funcionalidades.

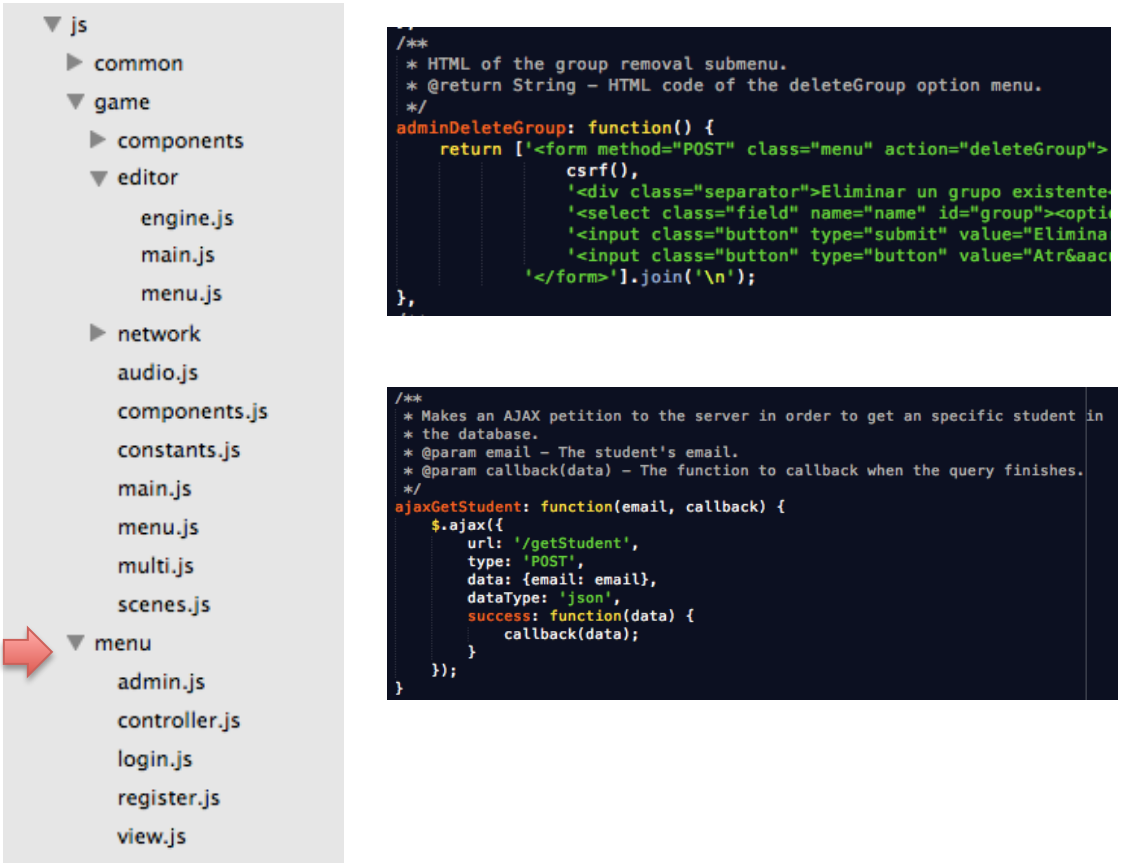


Figura 36: Rutas (Izquierda), Vista (Arriba), Controlador (Abajo)

F. Manual del administrador

Este manual pretende enseñar al profesor a instalar y gestionar el juego. Como se puede apreciar, a lo largo del mismo se muestran capturas de pantalla para guiar al lector. Pese a que los pasos mostrados han sido realizados en el sistema operativo Windows 7, todos ellos son realizables en cualquier sistema operativo de la misma forma que se explica en cada uno.

F.1. Descarga e instalación del software necesario

El primer paso a realizar por parte del administrador es escoger un ordenador en el que pueda alojar el juego. Para ello debe tener en cuenta que debe tener un espacio mínimo de 3,0GB libres en disco requeridos por la base de datos MongoDB y unos 25MB para node.js y el juego.

El sistema está pensado para que una sola máquina de servicio a todos los jugadores, de modo que el profesor deberá instalar el juego en un ordenador con conexión a red local, ya sea por cable o inalámbrica. No obstante, también se puede optar por instalar el juego en cada máquina, pero no es recomendable ya que los datos de alumnos y sus partidas no estarán centralizados en un mismo punto.

F.1.1. Instalación de MongoDB

Procedemos a descargar MongoDB de su página web oficial, http://www.mongodb.org/downloads. En el navegador nos aparecerán varias opciones en función del sistema operativo que tengamos, por lo que escogeremos la apropiada y esperaremos a que descargue.

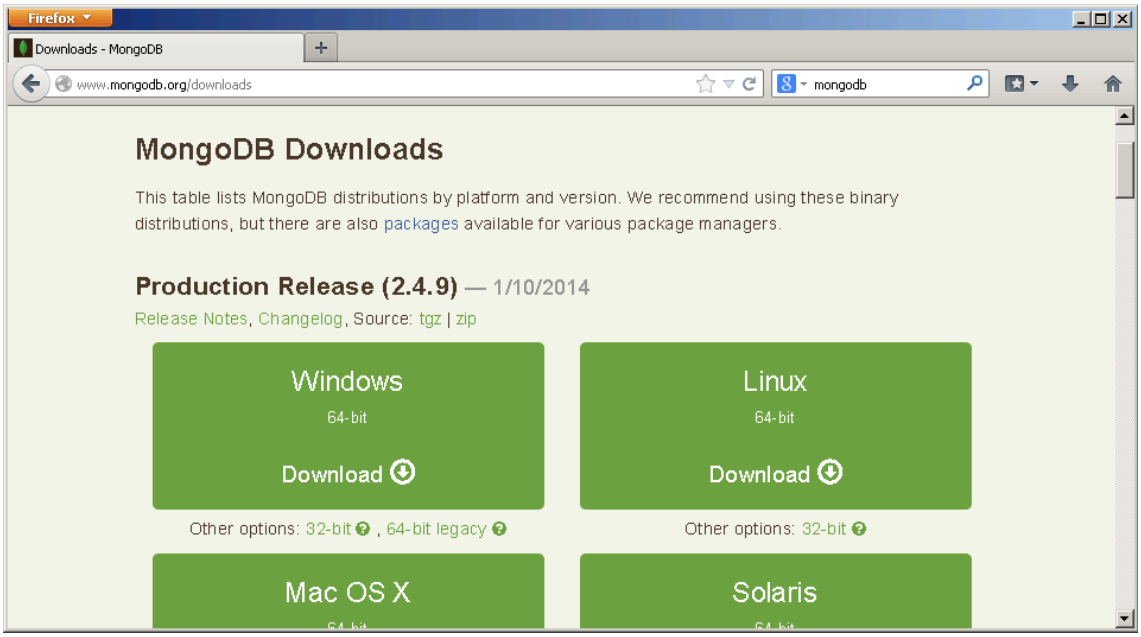


Figura 37: Sitio web de MongoDB

Una vez descargado, tendremos que crear dos directorios en las siguientes rutas:

- "*C:\mongodb*", donde alojaremos la carpeta "bin" que nos hemos descargado.
- "*C:\data\db*", donde se alojará la base de datos, aún sin crear.

A continuación, ejecutamos MongoDB para probar que todo funciona correctamente. Para ello debemos abrir un Símbolo del sistema en Menú inicio > Accesorios, o simplemente escribiendo "cmd" en la barra de búsqueda de este menú. Una vez abierta esta consola de comandos escribimos:

C:\mongodb\bin\mongod.exe

Debería aparecer una advertencia del cortafuegos de Windows preguntándonos si queremos dar privilegios a MongoDB. Aceptamos.

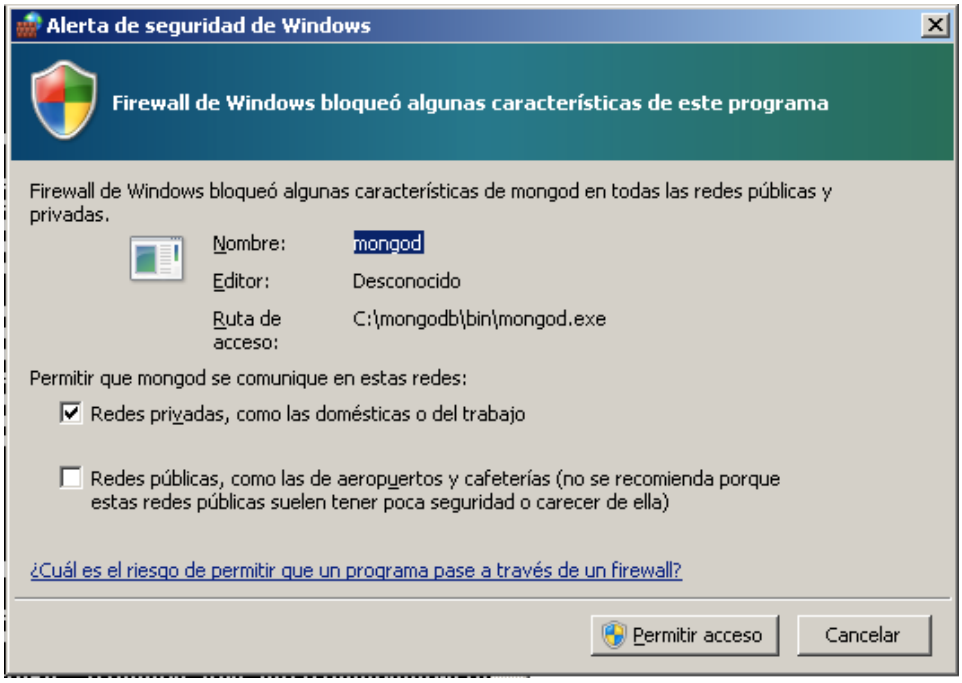


Figura 38: El cortafuegos de Windows pide permisos para ejecutar MongoDB

Una vez ejecutado MongoDB saldrán varios mensajes informativos, terminando con el de la figura 39, explicando que la base de datos está escuchando peticiones en el puerto 27017, de modo que la instalación es correcta y podemos continuar instalando.

Figura 39: Mensaje de la consola al iniciar MongoDB

F.1.2. Instalación de node.js

Procedemos a descargar node.js de su página web oficial, http://www.nodejs.org/download. En el navegador nos aparecerán varias opciones en función del sistema operativo que tengamos, por lo que escogeremos la apropiada y esperaremos a que descargue.

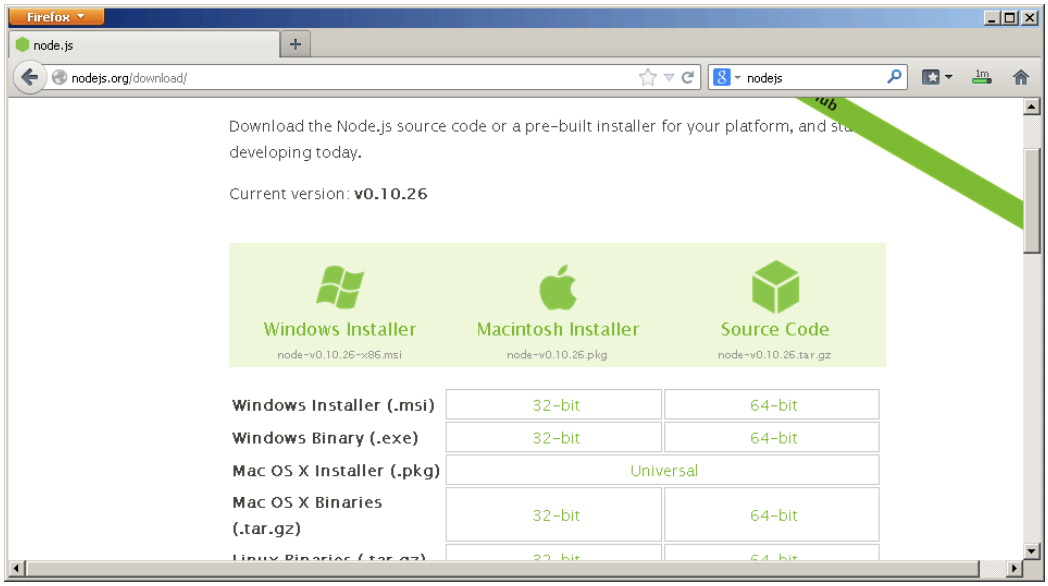


Figura 40: Sitio web de node.js

Una vez descargado, es suficiente con ejecutar el instalador y seguir los pasos.

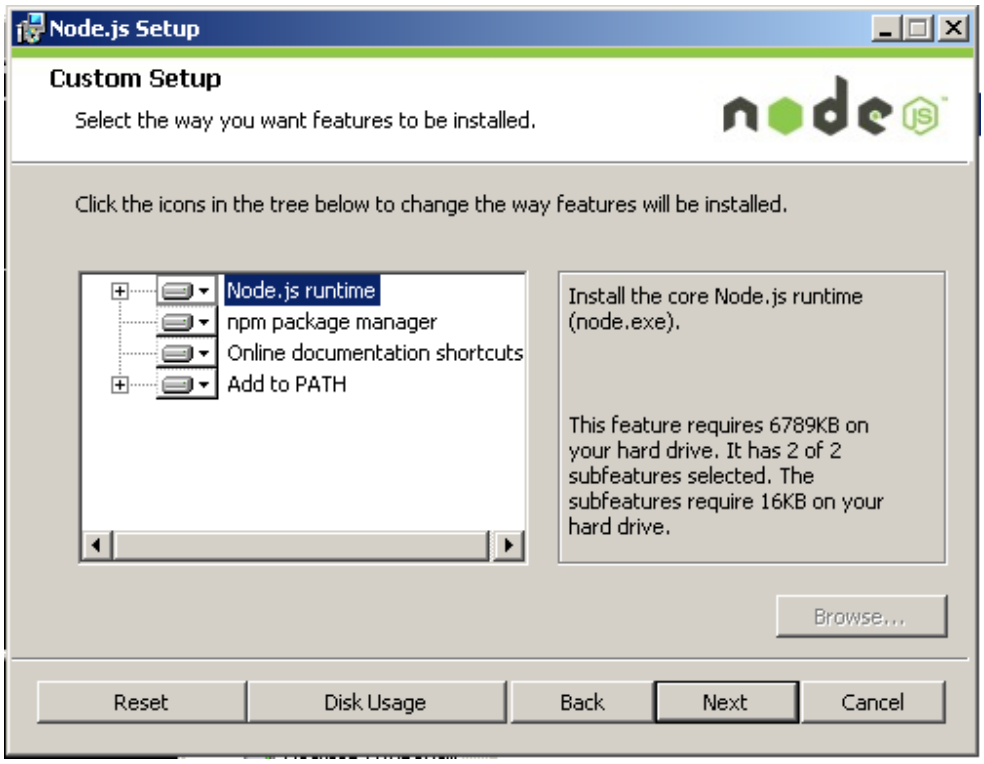


Figura 41: Paso del instalador de node.js

F.1.3. Instalación de Polynomial

Para instalar Polynomial sólo es necesario introducir los niveles y la cuenta de administrador en la base de datos. Para ello se ha incluido una carpeta llamada "database" dentro del proyecto con dos ficheros, "levels.js" y "admins.js". Para hacer más simple la operación, copiaremos estos dos ficheros a la carpeta C:\mongodb\bin donde instalamos MongoDB. Una vez hecho esto, abriremos otra consola de comandos y nos moveremos al directorio de MongoDB mediante la instrucción "cd C:\mongodb\bin", para después ejecutar los comandos de la figura (nótese que previamente debemos haber ejecutado MongoDB como se ha explicado en el último párrafo del apartado F.1.1):

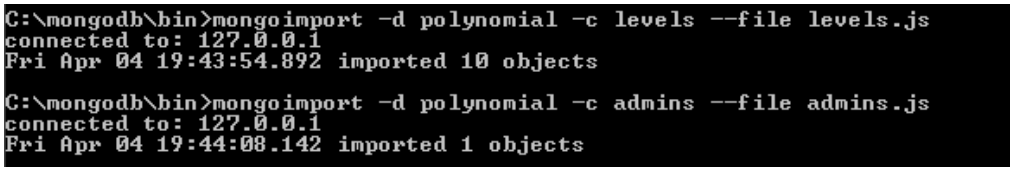


Figura 42: Instrucciones para instalar Polynomial

A continuación iremos al directorio donde se encuentra el proyecto con el comando "cd C:\ruta".

Cuando estemos en la carpeta deberemos ejecutar el comando "npm install". Nos saldrán los mensajes de la figura 43.

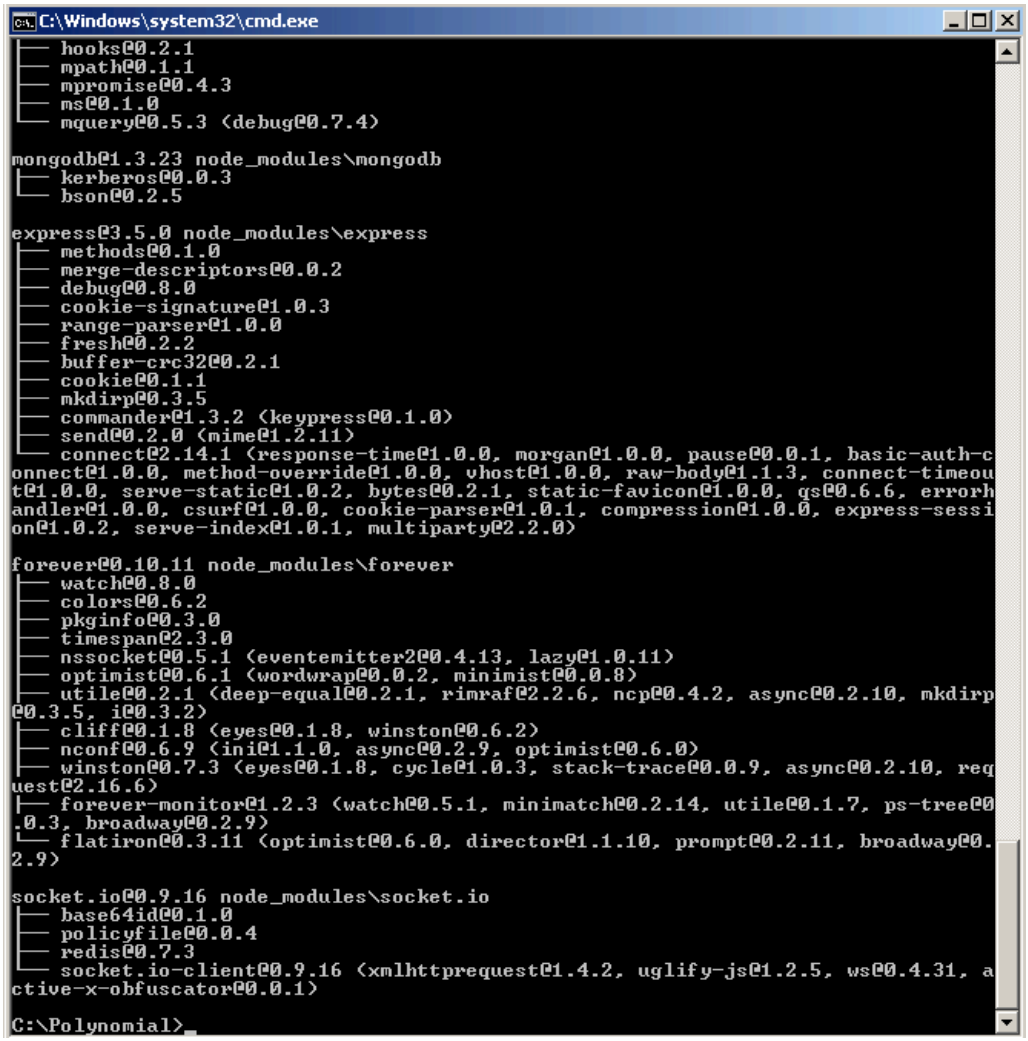


Figura 43: El comando npm install finaliza correctamente

Estos mensajes indican que la descarga de módulos se ha completado correctamente. Ahora tenemos que comprobar la configuración del servidor, para asegurar que es correcta. Para ello nos dirigimos a la carpeta del proyecto. Dentro veremos otra carpeta llamada "configuration" con varios ficheros de texto y dos carpetas dentro, "heroku" y "local". Si borramos los ficheros de texto y copiamos ahí mismo los que están dentro de "local" el juego debería funcionar. No obstante comprobamos su contenido:

- El contenido de "database.txt" debería ser "mongodb://localhost/polynomial".
- El contenido de "http_port.txt" debería ser "8080".
- El contenido de "https_port.txt" debería ser "8443".
- El contenido de "groups_needed.txt" debería ser "true".

F.1.4. Ejecución del servidor

Podemos ejecutar el servidor con la instrucción "node main.js" (en nuestra consola de comandos deberíamos estar en C:\ruta_del_proyecto como la habíamos dejado en el paso anterior). Debería aparecer otro aviso del cortafuegos de Windows, al que le daremos permiso de acceso:

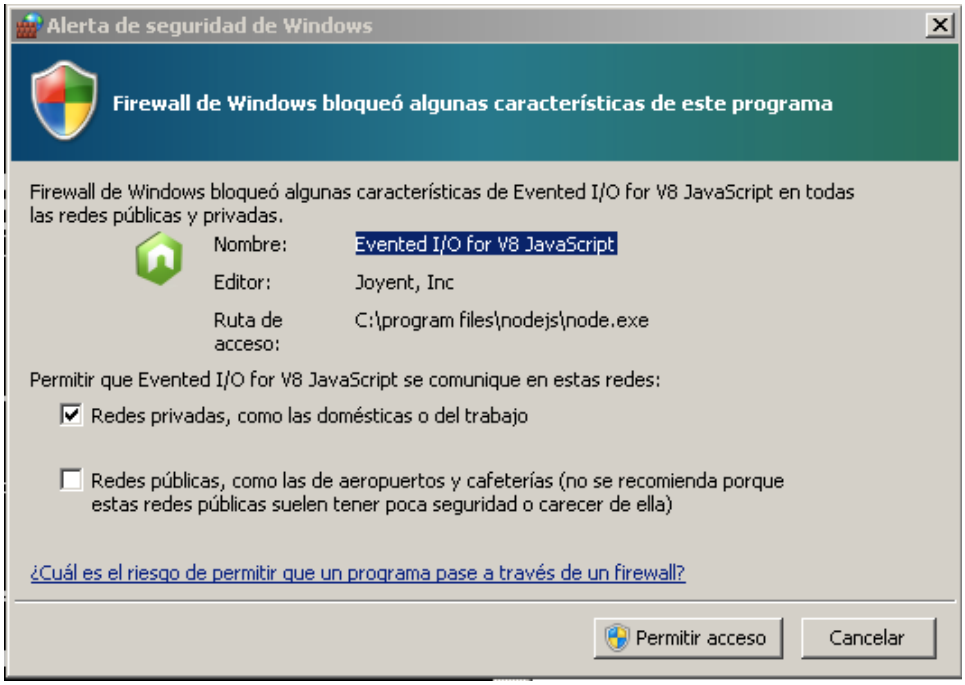


Figura 44: Permiso del cortafuegos a node.js

Una vez aceptado, aparecerá lo siguiente en la consola, confirmando que el juego está funcionando correctamente:

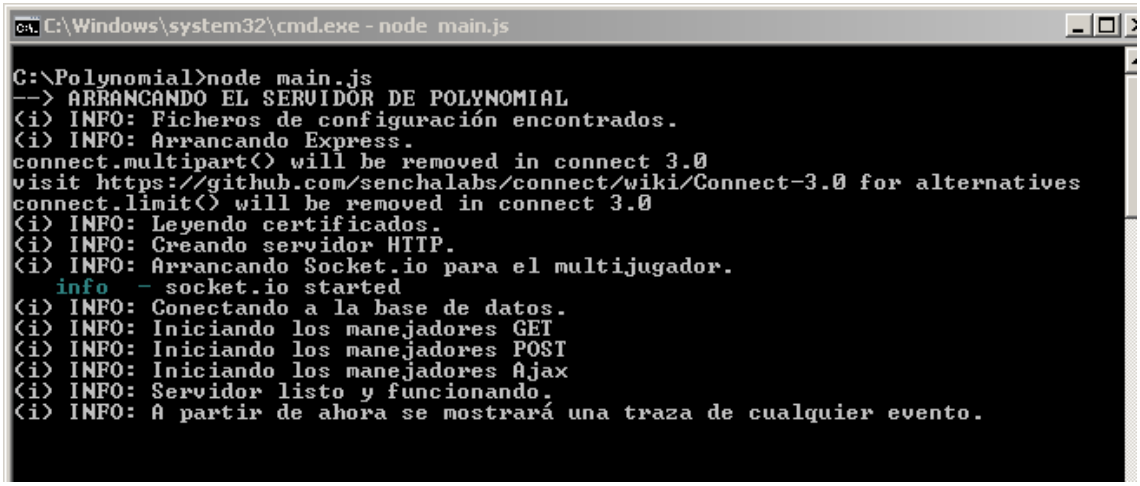


Figura 45: Arranque de Polynomial

Para que los jugadores se conecten a la máquina donde hemos instalado el juego necesitamos la IP de nuestro ordenador. Simplemente ejecutamos “ipconfig /all” como aparece en la figura y anotamos la dirección IPv4: “10.0.2.15”.

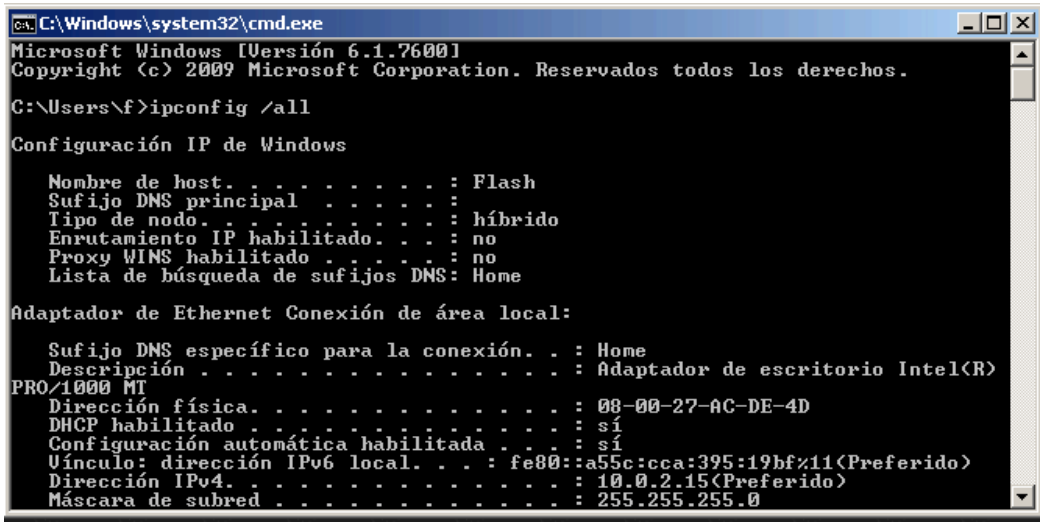


Figura 46: Comando ipconfig /all

De modo que los usuarios deberán introducir 10.0.2.15:8080 en su navegador para jugar. El administrador de la máquina puede entrar también con “localhost:8080”.

F.2. Administración

La cuenta del administrador es la siguiente (nótese que se podrá cambiar posteriormente):

- Usuario: “admin”
- Contraseña: “af35nad*”

Al introducir los datos en el formulario de login deberíamos acceder al panel de administración.

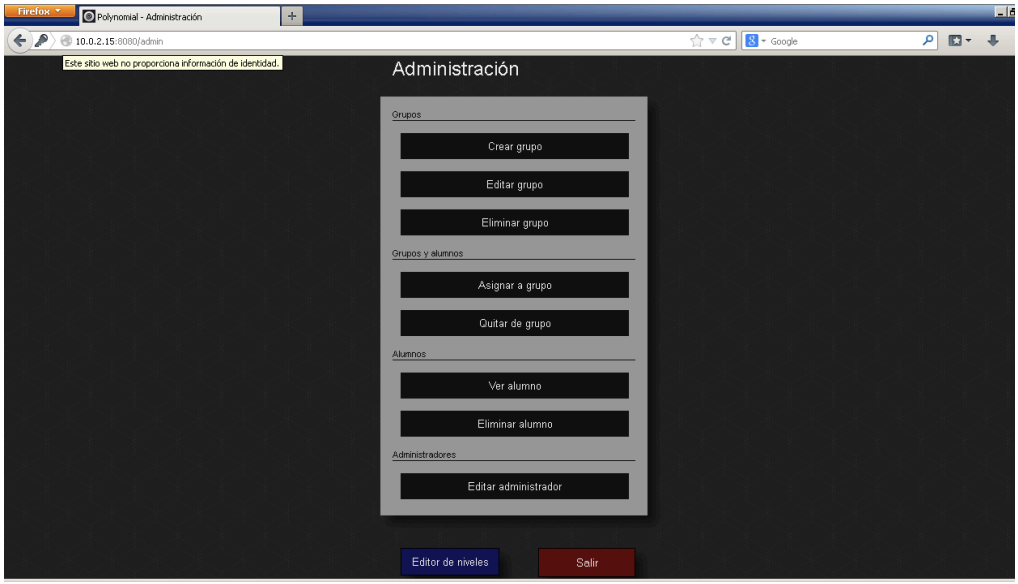


Figura 47: Panel de administración

A continuación se explica cada opción del panel.

F.2.1. Crear grupo

Como su nombre indica, permite crear un grupo completamente nuevo. Si acabamos de instalar el juego este paso será necesario si queremos que nuestros alumnos entren, ya que para ello necesitan pertenecer a uno.

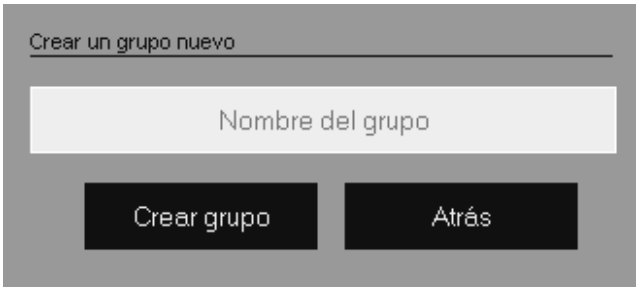


Figura 48: Crear un grupo nuevo

F.2.2. Editar grupo

Permite cambiarle el nombre a un grupo ya existente. Para ello debe existir un grupo que hayamos creado perviamente.

Figura 49: Editar un grupo existente

F.2.3. Eliminar grupo

Esta opción elimina un grupo existente. Hay que tener en cuenta que esta acción dejará sin grupo a todos los alumnos que lo tuvieran asignado.

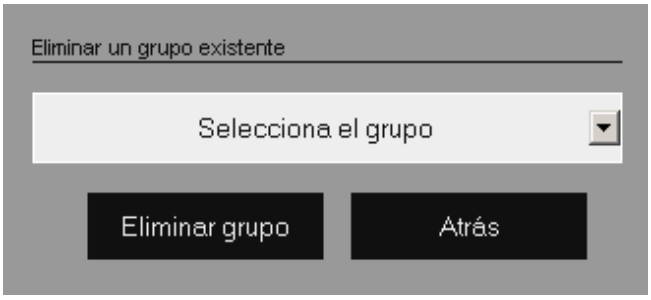


Figura 50: Eliminar un grupo existente

F.2.4. Asignar a grupo

Permite asignar un grupo ya existente a un alumno que se haya registrado y que aún no pertenezca a ninguno. Esta acción es esencial para permitir a un alumno entrar al juego.

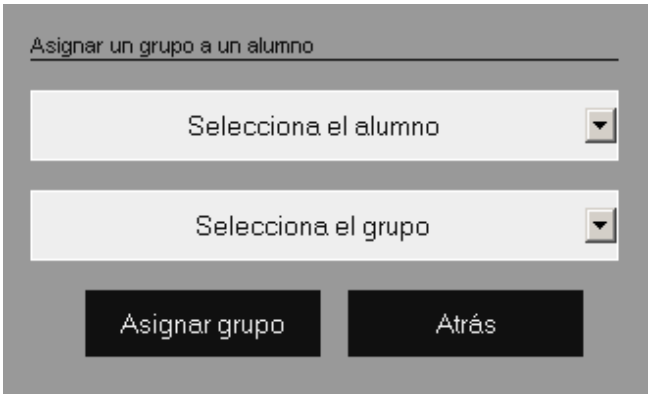


Figura 51: Asignar un grupo

F.2.5. Quitar de grupo

Esta acción quita a un alumno de su grupo, pero no lo borra.

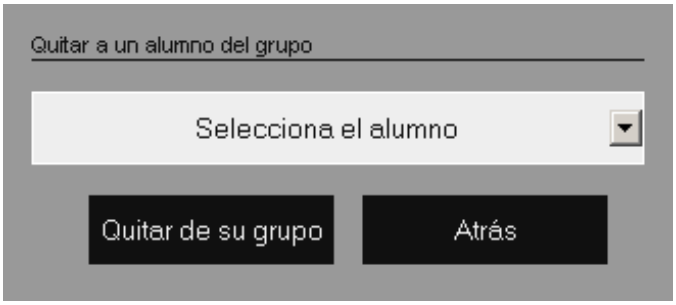


Figura 52: Quitar a un alumno del grupo

F.2.6. Ver alumno

Esta opción permite visualizar todos los datos de un alumno, desde su nombre y apellidos hasta las estadísticas de su partida, que incluyen el nivel por el que está jugando actualmente y un historial de los niveles superados con la fecha en la que lo logró.

Para borrar el historial de partidas, basta con pulsar el botón “Borrar historial”.

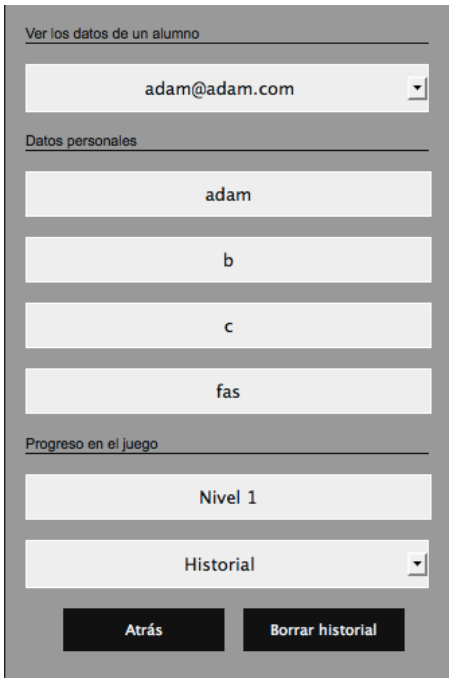


Figura 53: Ver un alumno

F.2.7. Eliminar alumno

Esta opción elimina un alumno completamente y lo quita del grupo en el que estaba. El borrado es irreversible, por tanto hay que usar esta opción con precaución.

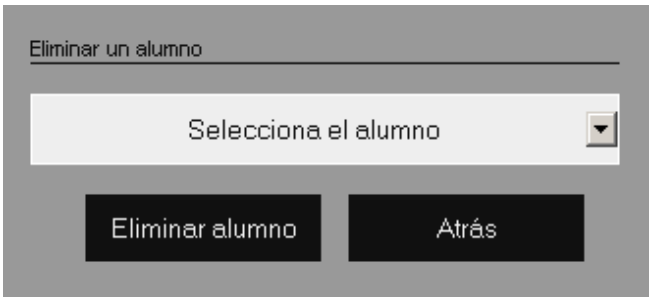


Figura 54: Eliminar un alumno

F.3. Edición de niveles

Para acceder al editor de niveles haremos click en el botón “Editor de niveles” de la parte inferior del menú de administración.

Figura 55: Botón del editor de niveles

En la parte inferior tenemos una serie de botones para crear, cargar, guardar, mover o eliminar niveles.

F.3.1. Nuevo nivel

Permite crear un nuevo nivel desde cero, colocándolo inmediatamente al final de todos los que hay disponibles. Esta opción activa la paleta de gráficos para poder poner al personaje, terreno, enemigos, etc.

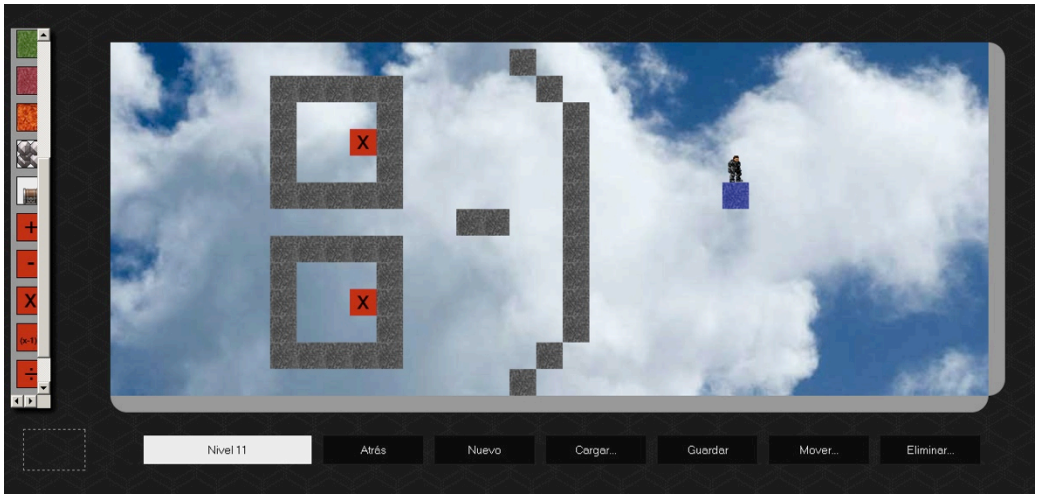


Figura 56: Al pulsar "Nuevo" crearemos un nuevo nivel

F.3.2. Cargar nivel

Permite cargar un nivel que hayamos guardado previamente.

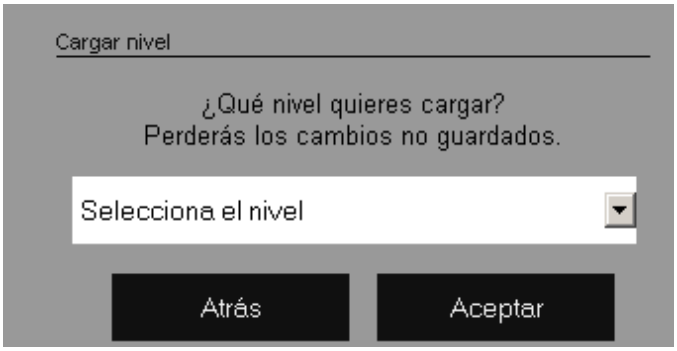


Figura 57: Cargar nivel

F.3.3. Mover nivel

En ocasiones puede resultar interesante intercambiar la posición de dos niveles. Esta opción permite hacerlo de forma sencilla. Hay que tener en cuenta de que si uno de los niveles afectados es el que tenemos actualmente abierto, se cerrará.

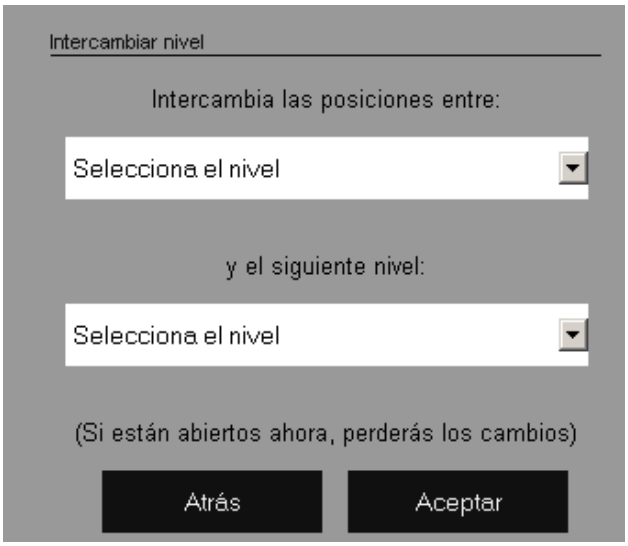


Figura 58: Mover nivel

F.3.4. Borrar nivel

Permite eliminar un nivel permanentemente. Esta acción reajusta todos los demás niveles automáticamente para que el juego permanezca consistente, es decir, si borramos el nivel 3, el 4 pasará a ser el 3, el 5 el 4 y así sucesivamente. Esta operación cierra el nivel abierto actual, y hay que tener en cuenta que el borrado es irreversible.

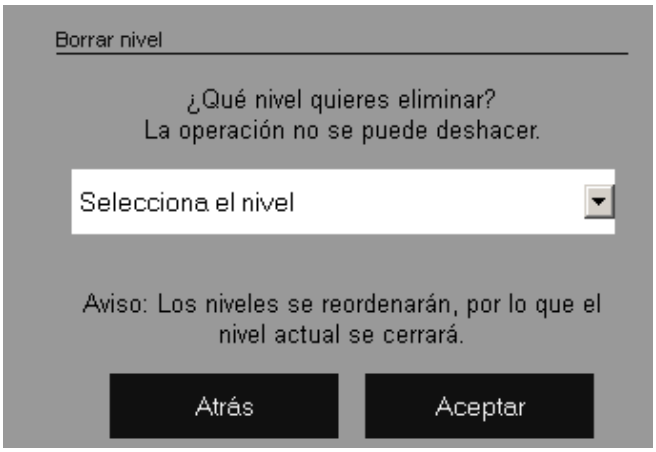


Figura 59: Borrar nivel

G. Guía de despliegue en Heroku

En algunos casos puede ser interesante la opción de subir el juego a Internet para jugar desde cualquier sitio y no sólo en las aulas. Heroku es un servicio en la nube que permite alojar aplicaciones creadas con node.js entre otros muchos tipos más. Se ha elegido este servicio por contar con un plan gratuito y por ser sencillo de configurar.

G.1. Creación de un repositorio en GitHub

El primer requisito para subir el juego a Heroku es tener alojado el código en este servicio gratuito de control de versiones. Para ello iremos a http://www.github.com y crearemos una cuenta rellenando los datos que nos solicita. Posteriormente nos descargaremos su programa de escritorio y lo instalaremos.

Cuando tengamos GitHub instalado, crearemos un repositorio nuevo con el botón de la figura 60, que aparece en la parte inferior de la aplicación con un símbolo "+":

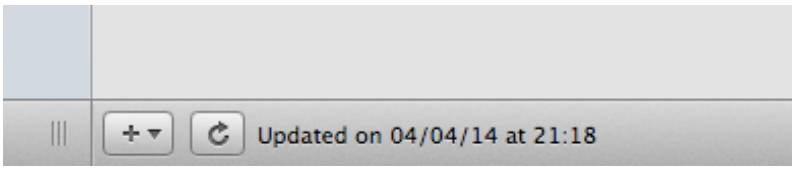


Figura 60: Creamos un repositorio con el botón "+"

Al hacer click nos saldrá un formulario como el de la figura 61. Aquí se nos pide el nombre del repositorio, una breve descripción (opcional) y la carpeta donde lo alojaremos localmente. Aquí es donde debemos introducir el proyecto una vez creemos el repositorio.

G. Guía de despliegue en Heroku

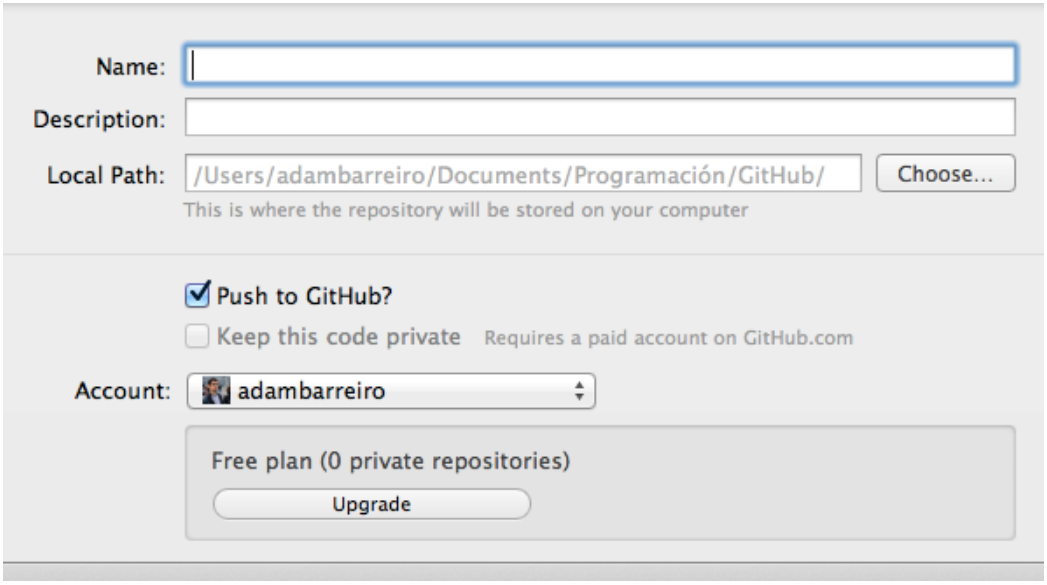


Figura 61: Creación de un repositorio en GitHub

Si lo hemos creado correctamente, en nuestra cuenta de GitHub aparecerá el repositorio (figura 62).

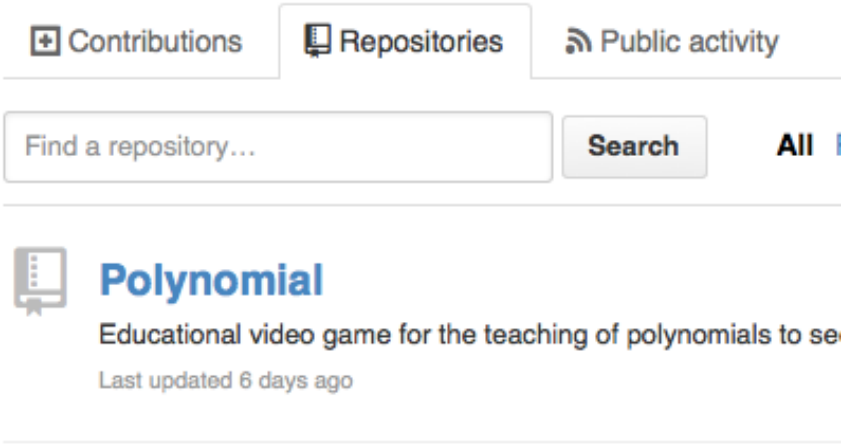


Figura 62: Repositorios en GitHub

Al introducir el proyecto en la carpeta local que hemos indicado al crear el repositorio, deberemos sincronizarlo para subir los cambios. Para esto activamos la opción Commit & Sync con el botón verde de la figura 63 y pulsamos el botón.

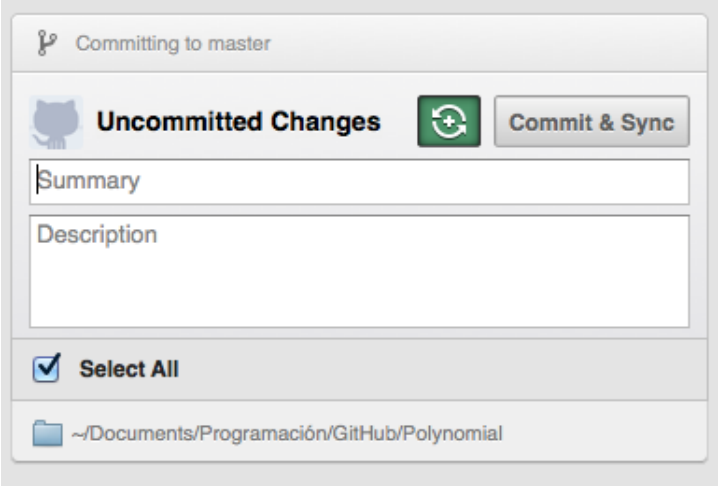


Figura 63: Cómo hacer un commit en GitHub

G.2. Creación de una base de datos remota

Para subir la aplicación a Internet también es necesario alojar la base de datos en un entorno online, de modo que necesitamos registrarnos en un servicio que ofrezca una base de datos MongoDB. Este sitio es http://www.mongolab.com, que se ha elegido por contar con un modo gratuito sencillo. Lo único que necesitamos es una cuenta y crear una nueva base de datos, teniendo en cuenta que debemos elegir las opciones de la figura 64.

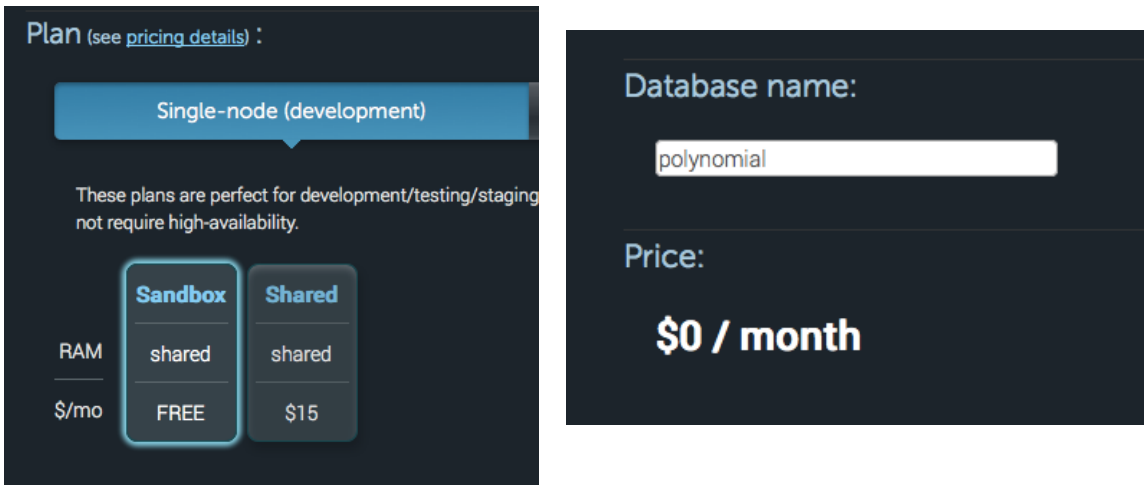


Figura 64: Configuración de MongoLab

Una vez creada, debemos asignarle un usuario y contraseña para evitar el acceso de personas extrañas. Vamos a la sección de usuarios y rellenamos el formulario de la figura 65 con el usuario y contraseña que queramos.

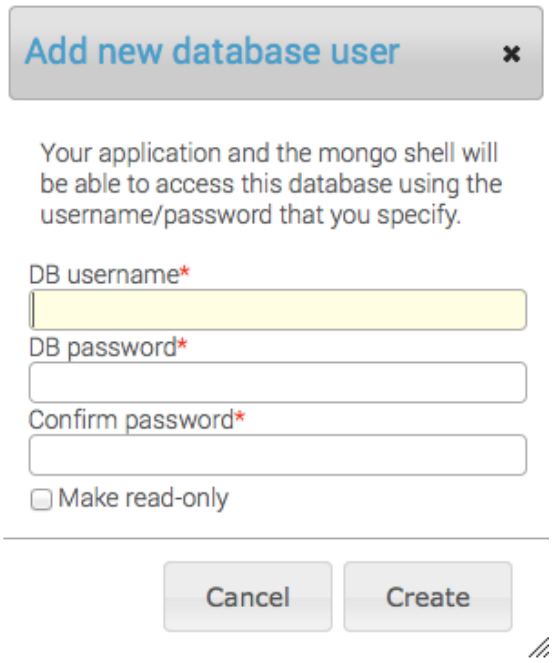


Figura 65: Creación de usuarios en MongoLab

Todos estos campos son los que pondremos en el fichero "database.txt" dentro de la carpeta "configuration/heroku" como se verá en el apartado G.4. En la página principal de MongoLab ya nos dan el contenido del fichero, que sigue esta sintaxis:

```
mongodb://<usuario>:<password>@*.mongolab.com:nnnnn/<nombrebbdd>
```

A continuación tenemos que abrir una consola de comandos y ejecutar "mongodump –d polynomial –o directorio". Esto nos dará una copia exacta de nuestra base de datos que podremos importar en MongoLab con otra instrucción que también tenemos que ejecutar y que MongoLab nos proporciona en el apartado de "Tools" y que aparece en la figura 66.

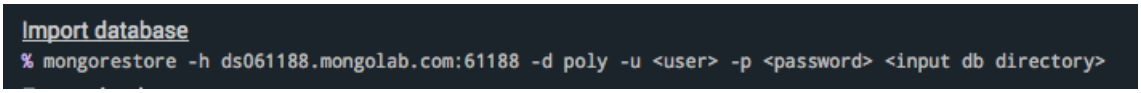


Figura 66: Instrucción de MongoLab para importar BBDD

G.3. Instalación del Heroku Toolbelt

Procedemos a registrarnos en Heroku para posteriormente descargarnos su software, que nos permitirá gestionar las aplicaciones que subamos mediante línea de comandos. Este software está accesible desde https://toolbelt.heroku.com y se llama Heroku Toolbelt.

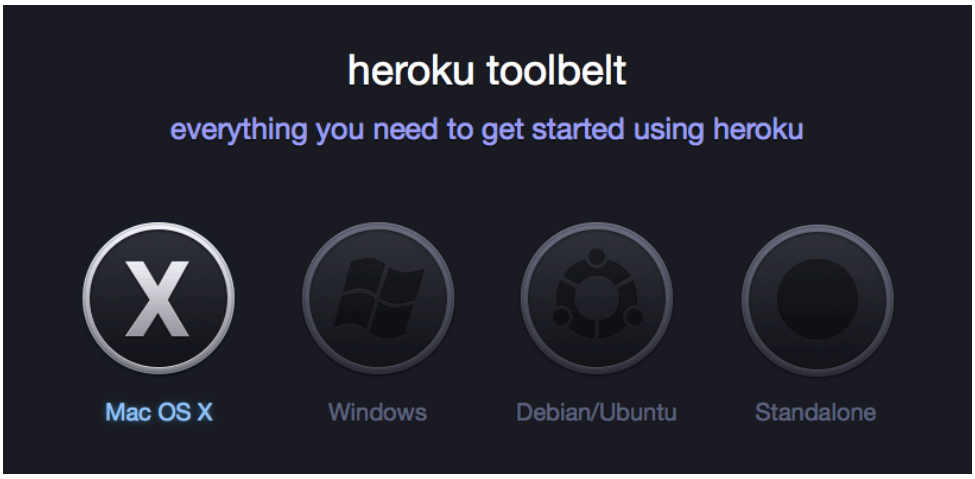


Figura 67: Pantalla de descarga de Heroku Toolbelt

Una vez instalado, abrimos una consola de comandos y hacemos login mediante la instrucción de la figura 68.

```
$ heroku login
Enter your Heroku credentials.
Email: zeke@example.com
Password:
Could not find an existing public key.
Would you like to generate one? [Yn]
Generating new SSH public key.
Uploading ssh public key /Users/adam/.ssh/id_rsa.pub
```

Figura 68: Login en Heroku

Para que los servidores de Heroku sepan qué fichero tienen que ejecutar, tenemos que introducir en la carpeta del proyecto un fichero llamado Procfile con este contenido:

```
web: main.js
```

Este fichero ya está incluido en la carpeta, de modo que no hace falta crearlo otra vez. Para crear una aplicación tenemos que abrir una consola de comandos y movernos a la carpeta donde tenemos el código en local, para posteriormente ejecutar el comando de la figura 69.

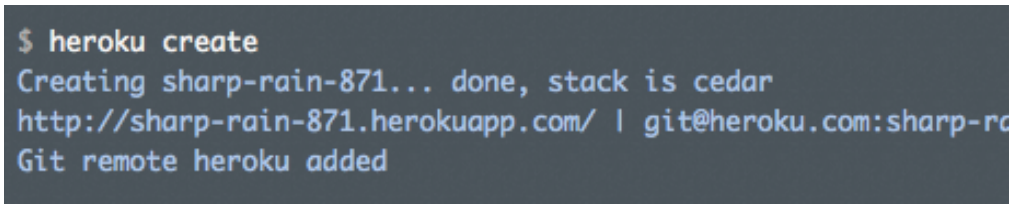


Figura 69: Crear una aplicación en Heroku

Opcionalmente podemos introducir un nombre detrás de "create" para ponerle un nombre a nuestra aplicación, si no Heroku le asignará un nombre aleatorio, como "sharp-rain-871":

```
$ heroku create pepito-app
```

G.4. Modificación de código y configuración

Antes de subir la aplicación debemos asegurar que la configuración es correcta. Para ello nos dirigimos a la carpeta del proyecto. Dentro veremos otra carpeta llamada "configuration" con varios ficheros de texto y dos carpetas dentro, "heroku" y "local". Si borramos los ficheros de texto y copiamos ahí mismo los que están dentro de "heroku" el juego debería funcionar. No obstante comprobamos su contenido:

- El contenido de "database.txt" debería ser del estilo mongodb://<usuario>:<password>@*.mongolab.com:nnnnn/<nombrebbdd> como se ha comentado en el apartado G.2.
- El contenido de "http_port.txt" debería ser "5000".
- El contenido de "https_port.txt" es indiferente porque Heroku no soporta HTTPS en su forma gratuita.
- El contenido de "groups_needed.txt" es indiferente, podemos poner "false" para no requerir la asignación del administrador si estamos en fase de pruebas de algún tipo.

Heroku implementa WebSockets de forma experimental, y de un tipo muy específico, por lo que también debemos modificar una línea de código concreta, que se detalla en la figura 70.

```
/**
 * Creates the socket.io structures and message protocol for the multiplayer.
 */
function createRouter(server) {
    // Creates the socket
    io = io.listen(server);
    io.configure( function(){
        io.set('log level', 1);
        // heroku labs:enable websockets -a myapp
        // Local:
        // io.set('transports',['websocket','xhr-polling','flashsocket','htmlfile', 'jsonp-polling']);
        // Heroku:
        // io.set('transports',['xhr-polling','websocket','flashsocket','htmlfile', 'jsonp-polling']);
        io.set('transports',['websocket','xhr-polling','flashsocket','htmlfile', 'jsonp-polling']);
        io.set('polling duration',10);
    });
    // Connection events
    io.on('connection', function (socket) {
        responseReady(socket);
        responseJoin(socket);
        responseEngaged(socket);
        responseClose(socket);
        responseDisconnect(socket);
        responseSender(socket);
    });

}
exports.createRouter = createRouter;
```

Figura 70: Modificación de código requerida para Heroku

Como se puede observar, si vamos a subir la aplicación a Heroku hay que usar la línea de código que hace un io.set() con “xhr-polling” en primer lugar, de modo que sustituiremos la línea que está activa por ésta. Además, en la consola de comandos ejecutaremos:

```
$ heroku labs:enable websockets –a nombre_de_mi_aplicacion
```

G.5. Despliegue

Una vez configurado, volvemos a la consola de comandos. Nos desplazamos a la carpeta del proyecto y ejecutamos la instrucción de la figura 71. Si todo va bien, al introducir la url que se nos indica en la última línea de texto deberíamos ver el juego ejecutándose.

```
$ git push heroku master
Counting objects: 343, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (224/224), done.
Writing objects: 100% (250/250), 238.01 KiB, done.
Total 250 (delta 63), reused 0 (delta 0)

-----> Node.js app detected
-----> Resolving engine versions
       Using Node.js version: 0.10.3
       Using npm version: 1.2.18
-----> Fetching Node.js binaries
-----> Vendoring node into slug
-----> Installing dependencies with npm
       ....
       Dependencies installed
-----> Building runtime environment
-----> Discovering process types
       Procfile declares types -> web

-----> Compiled slug size: 4.1MB
-----> Launching... done, v9
       http://sharp-rain-871.herokuapp.com deployed to Heroku

To git@heroku.com:sharp-rain-871.git
 * [new branch]      master -> master
```

Figura 71: Despliegue de la aplicación en Heroku

H. Manual del alumno

H.1. Argumento

Eres el único soldado superviviente de una expedición bélica realizada en la luna Europa. El plan consistía en estudiar los materiales y su posible uso en nuevas armas, pero lo único que se encontró fue un portal del que salieron unas extrañas criaturas que exterminaron a todos los integrantes del grupo.

Estas criaturas son inmunes a cualquier arma normal, salvo una: El poder de los polinomios. Has conseguido infiltrarte en el portal y llegar al mundo de las criaturas. Ahora tienes la obligación de llegar hasta el final y destruir a las criaturas para poder salvar la galaxia y a ti mismo.

H.2. Registro y login

Para acceder al juego el profesor debe proporcionarte el enlace para poder introducirlo en la barra de direcciones del navegador (Mozilla Firefox o Google Chrome). Una vez aparezca la pantalla principal verás el formulario de login (figura 72, izquierda). Si no tienes una cuenta puedes registrarte haciendo click en “Registrar”, que te llevará al formulario de la figura 72 (derecha).

Figura 72: Cargar nivel

En cualquiera de los dos casos, tu profesor debe asignarte a un grupo antes de poder jugar.

H.3. Modos de juego

Una vez dentro del juego verás dos opciones:

- Modo para un jugador: En este juego tendrás que enfrentarte a los enemigos sin ninguna ayuda. Puedes empezar una partida nueva o continuar la que dejaste empezada.

- Modo cooperativo: Crea una partida en este modo y deja que un amigo te eche una mano. Dado que él no obtendrá nada a cambio, asegúrate de echarle una mano tú después.

H.4. Mecánica del juego

Tu misión es llegar hasta el final de cada nivel, de un total de 10. En cada uno encontrarás diferentes enemigos que patrullan en tu búsqueda. De algunos podrás esconderte para atacarles en las sombras, pero a otros deberás enfrentarles cara a cara.

Al atacar te saldrán operaciones con polinomios que debes realizar. Si atacaste escondido, no tendrás ninguna presión para hacerlas ya que no habrá límite de tiempo de ningún tipo. Pero si fallas una operación o eres descubierto… prepárate porque comenzará un contrarreloj que te obligará a responder antes de que la cuenta atrás llegue a 0 si no quieres perder vida.

También contarás con numerosos cofres a lo largo del juego, pero como nada es gratis en esta vida… para obtener su contenido deberás responder correctamente a una pregunta teórica de “verdadero” o “falso”.

H.5. Elementos de la pantalla de juego

El aspecto de la partida una vez empieces a jugar es el que muestra la figura 73.

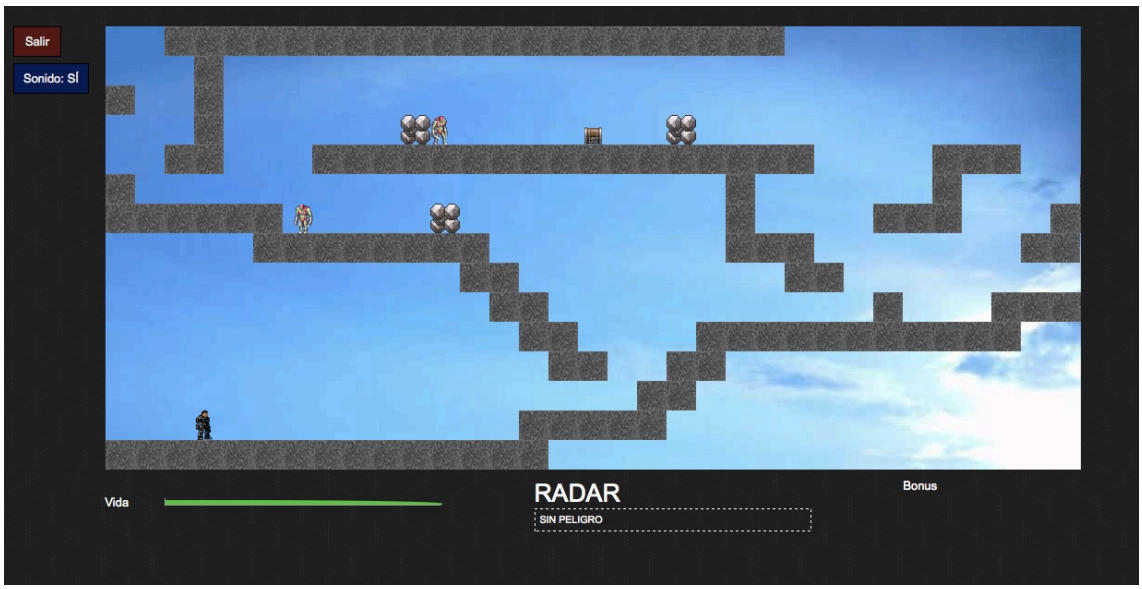


Figura 73: Pantalla de juego

H.5.1. Barra de vida

Es la energía de tu personaje. Si llega a 0, morirás y tendrás que empezar el nivel de nuevo. Ten en cuenta que las operaciones fallidas y caer en la lava te restará vida.

Cuando estés peleando contra un enemigo, su vida aparecerá debajo.

H.5.2. Radar

Tu radar es el elemento que te avisará si hay un enemigo cerca. Tiene varios tipos de peligro:

- **Sin peligro:** No hay nadie cerca. Puedes moverte con total tranquilidad.
- **Escondido:** Si estás escondido detrás de las piedras, serás invisible y tu radar se pondrá verde. Si además puedes atacar, tu radar se pondrá cian.
- **Cuidado:** Si estás al alcance de un enemigo pero aún está lejos, tu radar se pondrá amarillo y te avisará. Deberías esconderte o tomar precauciones. Si estás detrás del enemigo y demasiado próximo, te saldrá otro aviso, que te indicará que si el enemigo da la vuelta te descubrirá.

- **Alerta:** Si te descubren tu radar se volverá de color rojo y no habrá vuelta atrás, tendrás que luchar.

H.5.3. Bonus

Estos son los bonus que puedes obtener al responder correctamente a las preguntas que te planteen los numerosos cofres del juego:

Figura 74: Iconos de bonus

- **Vida:** Recuperarás una parte de tu salud.
- **Escudo:** Blindaje que te protegerá del daño. Su uso es limitado, así que ten cuidado.
- **Poder:** Te proporcionará unos golpes más contundentes durante 3 operaciones. Puedes ir acumulando más.
- **Tiempo extra:** Te proporcionará tiempo extra durante un combate entero. Puedes ir acumulando más.

H.6. Controles

Durante todo el juego se usarán únicamente estos controles:

- **Flechas de dirección:** Mover al personaje. Con la flecha "Arriba" haremos que salte.
- **Tecla A:** Permite atacar a un enemigo siempre y cuando esté dentro de nuestro alcance.
- **Tecla S:** Abre un cofre de tesoro.
- **Ratón:** Permite contestar preguntas y hacer click en los diversos botones como el botón para salir del juego o el de desactivar sonido.

- **Batallas:** En las batallas, para introducir polinomios, usa el teclado numérico, la tecla "x" y "Shift + ^" para introducir el gorrito del exponente. Por ejemplo $4x^2$ es 4x^2.

H.7. Sistema de batalla

Las rocas que hay esparcidas a lo largo de los niveles son un recurso valioso ya que te permiten protegerte de los enemigos y lanzarles un ataque que disparará una batalla sin contrarreloj.

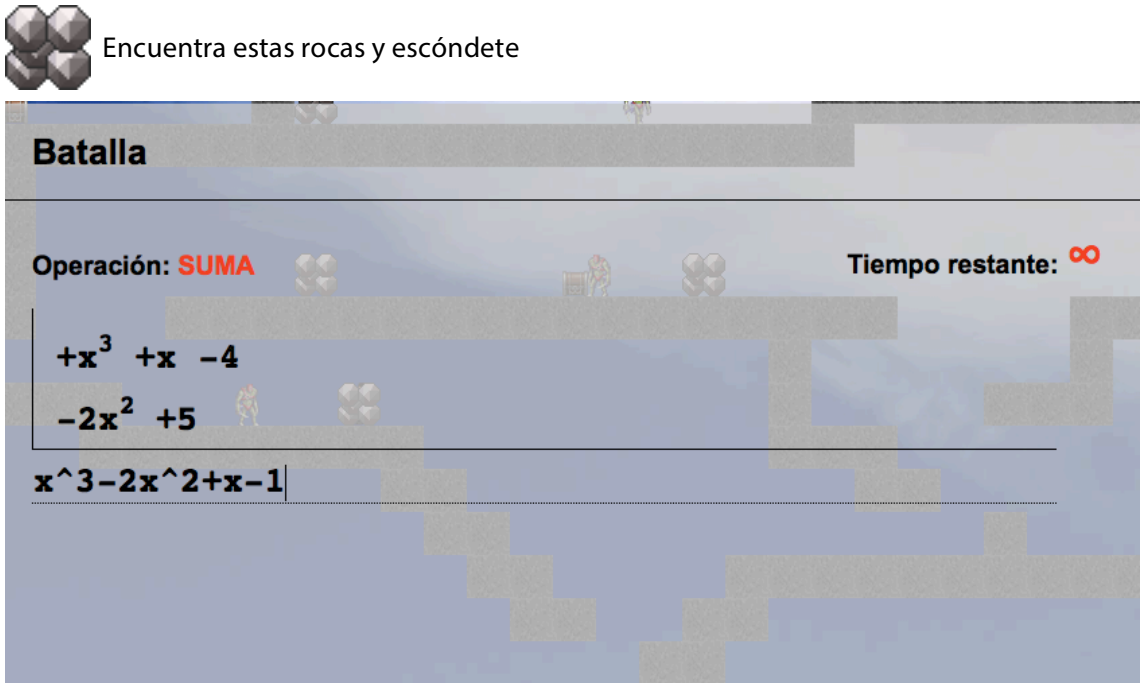


Figura 75: Batalla sin tiempo

En cambio, si fallas o eres descubierto, pasarás al sistema de batalla con tiempo. Si éste llega a cero perderás vida. Cada tipo de operación (sumas, restas, multiplicaciones, productos notables y divisiones) tienen un límite de tiempo diferente, así que ten cuidado.

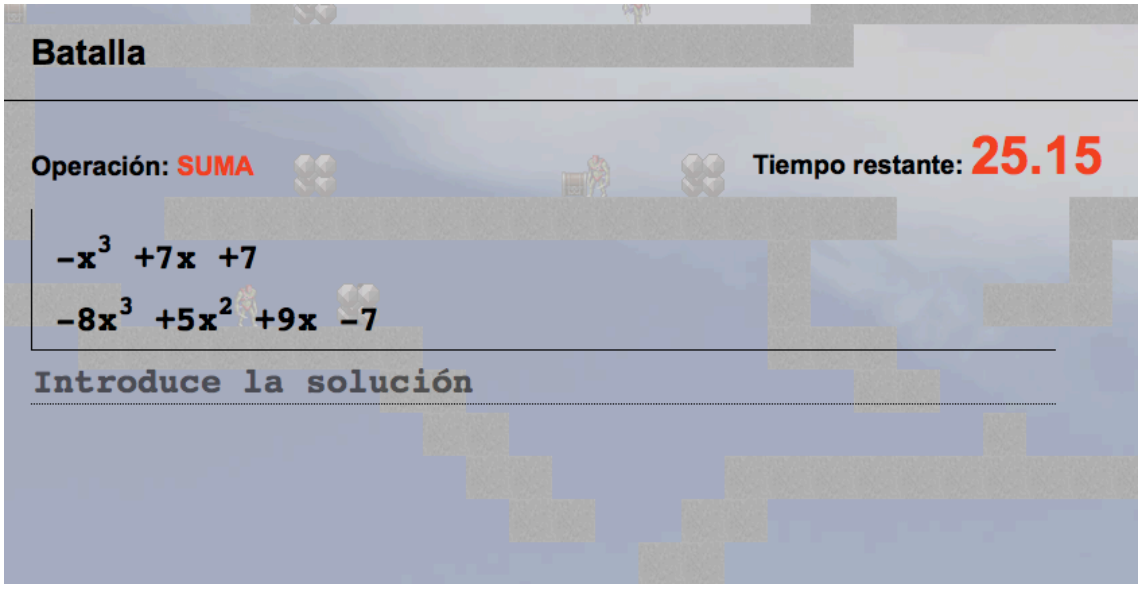


Figura 76: Batalla con tiempo

H.8. Multijugador

En el multijugador puedes compartir tu partida con un amigo para que te eche una mano. Podréis derrotar a los enemigos juntos para llegar a la meta más fácilmente. A lo largo del transcurso podrás ver todos sus movimientos en tu partida con el personaje de color verde.

Cada uno tiene sus propios cofres así que no tendréis que poneros de acuerdo para ver quién coge cual. Si alguno de los dos muere volverá a empezar desde el principio, pero el único requisito para pasar de nivel es que uno de los dos cruce la meta.

H.9. Consejos

- Ten cuidado con los precipicios que terminan en un foso con lava.
- Para las operaciones complejas como multiplicaciones o divisiones quizá necesites lápiz y papel.
- Si te atascas, comienza un cooperativo con un amigo. Juntos lograréis llegar más lejos.
- Presta mucha atención al tipo de operación que te están pidiendo.

III

Bibliografía, glosario e índice de figuras

Glosario

AJAX: Acrónimo de *Asynchronous JavaScript And XML*, permite realizar una llamada asíncrona al servidor para realizar una petición en segundo plano mientras el cliente sigue usando la aplicación web.

Asíncrono: Evento que se ejecuta en segundo plano mientras se atienden otros eventos, devolviéndose el resultado del mismo más tarde, sin bloquear el elemento que lo disparó.

Back-end: Parte de un sistema que se encarga de procesar los datos enviados desde el cliente (front-end) y devolver una salida al mismo.

Componente: En CraftyJS, conjunto de propiedades y funciones que son asignables a una entidad para que ésta las herede.

CSS: Acrónimo de *Cascading Style Sheets*, es el lenguaje de hojas de estilo utilizado para describir el aspecto y el formato de un documento escrito en un lenguaje de marcas como HTML.

Entidad: En CraftyJS, elemento unidad a la que se asignan componentes y a las que se aplican funciones y eventos, para convertirlos en un elemento del juego.

Framework: Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

JSON: Acrónimo de *JavaScript Object Notation,* es el formato que usa JavaScript para la transmisión ligera de información. Más simple que XML debido a su sintaxis clave-valor, es usado comúnmente en llamadas AJAX.

Índice de figuras

Bibliografía

[1] *"A Must-Have Guide To Gaming In The Classroom"*, Constance McKenzie, 2012
http://www.onlineschools.com/in-focus/gaming-in-classroom

[2] *"The gamification of education"*, Knewton, 2012
http://www.knewton.com/gamification-education/

[3] "When gaming is good for you", Robert Lee Hotz, 2012
http://online.wsj.com/news/articles/SB10001424052970203458604577263273943183932

[4] *"The Node Beginner Book"*, Manuel Kiessling, 2013
http://www.nodebeginner.org/

[5] *"Simplemente, ¿qué es node.js?"*, Michael Abernethy, 2011
http://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/

[6] *"Introduction to Crafty"*, Darren Torpey, 2013
http://buildnewgames.com/introduction-to-crafty/

[7] "MongoDB in Action", Kyle Banker, 2011
http://www.manning.com/banker/

[8] *"How to create a self-signed SSL Certificate"*
http://www.akadia.com/services/ssh_test_certificate.html