PARALLEL RUSH

Íñigo Álvaro Sáenz
Gonzalo Muelas Pozo

Fecha: 11/01/17

Tabla de contenidos

1. Descripcion general del proyecto

Parallel Rush es una aventura en 2D tipo puzzle, basada en un mundo dividido en dos dimensiones en las que habitan peligros insospechados. *¿Conseguirás llegar al final de la aventura?*

1.1. Introduccion

El objetivo principal de este proyecto es la creación de un videojuego básico completo, así como familiarizarse con lenguajes de programación web como HTML5 o Javascript, que, aunque no están pensados originalmente como lenguajes de desarrollo de videojuegos, han tomado cierta importancia en el campo de los videojuegos para jóvenes desarrolladores, debido a entre otras cosas, la portabilidad que ofrecen ya que se pueden ejecutar en cualquier dispositivo preparado con un navegador web. Por ello se ha decidido construir este juego mediante Javascript y HTML5.

En este documento se expone una descripción detallada del proyecto que se ha realizado. Se trabajarán aspectos que van desde la temática del juego, el modo de juego, el género, la usabilidad y el público al que va dirigido, hasta temas mas técnicos como la arquitectura del sistema (uso de patrones, estados del videojuego) e incluso aspectos muy específicos como la implementación de la IA, o de las diferentes clases del juego.

Además, este documento pretende ser una guía para los usuarios del juego, pudiendo así encontrar información sobre los controles, los tipos de trampas y de enemigos, el objetivo del juego, etc…

Sólo esperamos que el juego sea motivo de entretenimiento de los usuarios, seguiremos desarrollándolo y mejorándolo en la medida de lo posible.

1.2. Descripcion general del producto

El tipo de juego que se ha implementado es una aventura 2D en vista aérea, similar a un isométrico, pero con cámara a 90º (cenital).

El juego consiste en guiar a un personaje (robot) a través de un mundo basado en un ambiente futurista, y con una trama simple para justificar la historia: un monstruo ha escapado de su cárcel, un aparato de aspecto futurista que lo contenía, a otra dimensión; al hacerlo abrió un agujero entre ambas dimensiones y nuestro protagonista deberá atravesarlas para poder encerrarlo de nuevo. La peculiaridad del juego radica en que el escenario cambia, ya que existen dos dimensiones en el mundo. Entre estas existirán diferencias en el escenario, los objetos y los enemigos.

En ambas dimensiones existirán peligros, trampas y enemigos que el jugador deberá evitar para no perder sus vidas. En la dimensión de origen hay más variedad de trampas y elementos mientras que en la otra dimensión hay menos peligros, pero el monstruo nos perseguirá; si nos llega a pillar el jugador muere instantáneamente en lugar de perder una vida como en el resto de casos.

Además, se han diseñado niveles con una cierta picardía de forma que sólo se podrán superar resolviendo una especie de **puzle,** en el que intervienen las dos dimensiones. También hay elementos que dificultan nuestro paso, como suelos que nos llevan en una dirección o incluso con trampas que nos quitarán vida. El objetivo de cada nivel será encontrar el camino hasta ciertos objetos clave para la historia (piezas de la cárcel que contenía al monstruo), pero de forma que haya que superar algún obstáculo que se resuelva de forma similar a un puzle. Esto se consigue mediante la doble dimensión y el resto de elementos.

1.3. Funcionalidad

Como previamente se ha mencionado, el videojuego esta ambientado en un mundo futurista, parecido a la tierra, a través del cual tendremos que guiar a nuestro jugador para recolectar todas las piezas de un aparato futurístico. La función de este aparato era retener a un monstruo que vivía en una dimensión paralela. Cuando se rompió, el monstruo pudo escapar y volver a su dimensión de origen. Sin embargo al hacerlo abrió una brecha entre ambas dimensiones, lo que hace que nuestro protagonista pueda ir de una a la otra. De este modo, las dos dimensiones están ligadas, las acciones de una afectan a la otra, por lo que nuestro héroe se verá obligado a ir a través de ambas dimensiones para lograr su objetivo, teniendo que tener especial cuidado cuando atraviese la dimensión del monstruo, ya que este puede encontrarle en su dimensión pasado un tiempo (pequeña IA). En ciertas zonas del mapa podremos cambiar de dimensión a través de “grietas dimensionales” según nos convenga para la misión.

El jugador podrá moverse en cualquier dirección del eje x e y del mapa (arriba, abajo, derecha, izquierda) pero no podrá saltar. Tampoco podrá pasar por encima de las paredes. Existen elementos que modifican las físicas del juego, como casillas que resbalan y que mueven al jugador en una determinada dirección.

Para completar cada nivel tendremos que interactuar con diferentes elementos (puertas, botones…) que nos permitan seguir avanzando. Hay enemigos en todas las dimensiones, pero son distintos: los enemigos de la dimensión normal solo están en determinadas zonas del mapa y nos quitan una vida cada vez que nos tocan, mientras que el monstruo de la otra dimensión va libremente por todo el mapa y si nos alcanza, moriremos de inmediato. Además cuando estemos en la dimensión del monstruo, que denominaremos *oscura*, nuestra visibilidad se limitará a nuestro personaje y no

podremos ver mas que un círculo y unas tres casillas a la redonda. También emplearemos músicas distintas para nuestra dimensión y la oscura.

Cada vez que consigamos una pieza de la cárcel pasaremos al siguiente nivel. Para que el jugador pueda completar el juego en distintos momentos introduciremos una funcionalidad adicional de guardado: hemos utilizado *cookies* para guardar el nivel en el que se encuentra el jugador el jugador. El guardado y el cargado será automático, es decir, no tendremos que seleccionar un perfil para seguir por último nivel al que habíamos llegado. Sin embargo, las cookies tienen un tiempo de expiración de dos días, con lo que si llevamos más de 48 horas sin jugar perderemos todo nuestro avance.
Este método es similar a otros juegos de la web, en los que no es necesario guardar el proceso ya que se almacena automáticamente y puedes seguir desde dónde lo dejaste la próxima vez que entres.

Por último, se han añadido funcionalidades que mejoran la accesibilidad a la hora de movernos a través de los estados del videojuego. Nos referimos a los botones de control del sonido, y a los de *quit* (nos lleva al menú del juego) y *restart* (reinicia el nivel), esto dos últimos son accesibles desde la pantalla del videojuego. Además, cuando nos quedamos sin vida o el monstruo nos alcanza, morimos y el juego termina. Entonces aparece una pantalla en la que pone “GAME OVER” y podemos elegir entre volver al menú o jugar de nuevo. Implementamos esto con un nuevo estado llamado *dead.* También tenemos un estado para cuando el jugador consigue terminar el nivel, *win*, desde donde podemos pasar a elegir el siguiente nivel. Para elegir el nivel también tuvimos que añadir un estado *chooseLevel,* desde el cuál podremos acceder al siguiente nivel y a los que hayamos completado. El resto de niveles aparecen con un candado y solo se desbloquean cuando hemos superado el anterior.

El nivel principal se ha implementado en forma de cuadrícula de forma que cada una de sus habitaciones contiene una trampa o peligro a superar.
Se han creado tres niveles, ambientados en diferentes mundos o escenarios, lo que aporta un elemento gráfico interesante y algo más de variedad al juego.

El juego se ha desarrollado en inglés para llegar a un mayor público, aunque no supone ningún impedimento para ningún hablante, ya que es muy simple.

1.4. Entorno y herramientas de desarrollo

La aplicación se ha desarrollado utilizando los lenguajes de desarrollo web HTML5 y Javascript en el editor de texto Brackets, que permite visualizar el resultado de nuestro código en el navegador que deseemos (en este caso Google Chrome) mediante su opción de “Live preview”. De este modo no será necesario lanzar otro servidor HTTP para poder ver el resultado de nuestro videojuego. Además nuestro framework ha sido Phaser, utilizaremos las funciones de su API para facilitar el desarrollo de nuestro videojuego.

Hemos utilizado el modulo para Javascript RequireJS para implementar los patrones y estructurar el videojuego en clases. De esta forma se ha estructurado el código de una forma muy clara.

Desde el principio, nuestra idea era la de crear un juego completamente original, creando tanto los gráficos como la música desde cero. Finalmente ha sido posible llevarla a cabo y no ha sido necesario recurrir a recursos externos. En este sentido, para los gráficos se ha utilizado un editor de imagen avanzado como GIMP, que es gratuito y permite crear gráficos con resultados excelentes, y para crear los escenarios se ha utilizado la aplicación Tiled, que permite exportar los proyectos como formato ".json" e importarlos desde Phaser. La música se ha compuesto mediante varios softwares de composición musical. Únicamente los efectos de sonido se han obtenido de librerías de libre uso.

Se han utilizado recursos didácticos para desarrollo de Javascript estructurado que han sido muy útiles. Algunas guías relacionadas con RequireJs han sido de gran ayuda también, así como las charlas impartidas. Gracias a ello el juego se ha estructurado en forma de objetos o clases (POO) y ha sido mucho más fácil de desarrollar.

1.5. Requisitos Finales

Para utilizar el videojuego se requieren los siguientes elementos:

- Ordenador con Google Chrome (debe soportar HTML5 y Javascript) y Brackets o cualquier otro ordenador en el que ejecutar un servidor HTTP y un navegador, si no se dispone de Brackets o Google Chrome.
- Teclado (es suficiente para los controles del jugador). Finalmente, no se ha implementado ningún control con *Gamepad*, ya que los controles son demasiado simples para incorporarlo.

2. Analisis/Diseno

Nuestro videojuego sigue un patrón de estados general: *boot* (boot.j), *load* (load.js), *menu* (menu.js), *play* (play.js y play2.js). Sin embargo, nosotros hemos añadido además nuevos estados como *dead* (dead.js) al que vamos cuando morimos, o *chooseLevel*, desde el cual podremos elegir el nivel que queremos jugar. Desde éstos podemos ir a los estados *menu* y *play*. El estado *game* (game.js) es necesario para configurar los estados, crear el juego e inicializar *Phaser*. Cuando integremos el diseño estructurado en clases necesitaremos un primer estado adicional para configurar RequireJS.

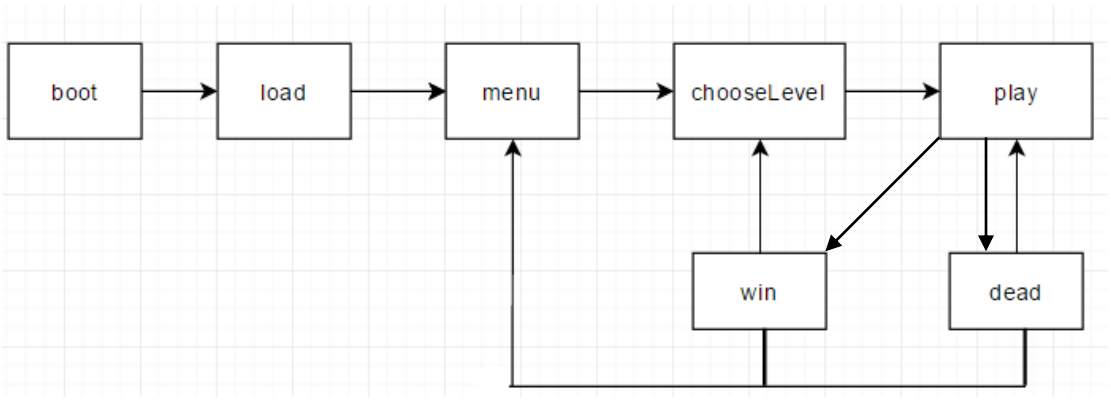


Imagen 1. Secuencia de estados del videojuego

Como hemos mencionado, el videojuego final está estructurado en clases siguiendo el siguiente diseño:

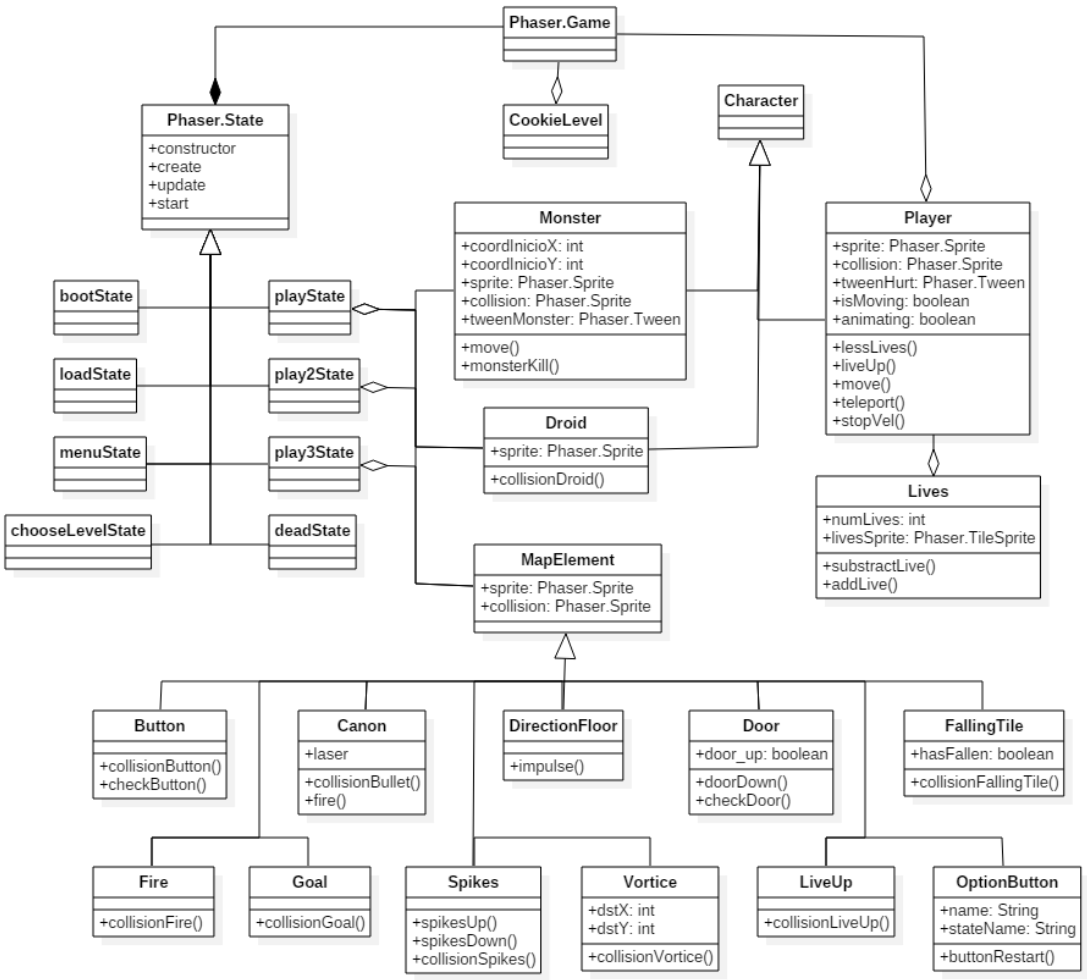


Imagen 2. Diagrama de clases.

3. Implementación

En esta sección se describe el alcance de esta versión final y sus diferencias con las anteriores entregas. Además, más abajo se puede encontrar una guía del juego detallada, con el objetivo, los personajes, los enemigos, las trampas y otros elementos sobre los que pudieran surgir dudas al usuario.

Cada elemento del videojuego conforma una clase distinta, implementada en un fichero Javascript distinto. Hemos dividido el código fuente en varios tipos de objetos. Por un lado, hemos implementado los personajes (*characters*), el jugador principal, el monstruo y los droides. Cada uno implementa sus propios métodos por lo que es más fácil programar de forma ordenada. En una primera versión pensamos en hacer que estas clases heredasen de la clase Phaser.Sprite, sin embargo, finalmente hemos decidido que el *sprite* sea sólo uno de los atributos de la clase. De esta forma hemos podido añadir otros atributos útiles a los personajes que simulan otras de sus características y propiedades. Por otro lado, hemos creado elementos del mapa (mapElements) organizados de la misa forma que los personajes. Cada elemento tiene un *sprite* y otros atributos adicionales además de los métodos de su clase, como por ejemplo el que define qué hacer cuando se produce una colisión. Por último, hemos creado una clase dedicada a trabajar con las cookies, con el objetivo de guardar el progreso del jugador. Esta clase puede crear, chequear y actualizar las cookies.

La estructura del código es la siguiente. Nuestro juego hereda de la clase Phaser.Game y tiene atributos propios del juego, es decir, genéricos. Por ejemplo, el jugador siempre será el mismo y por tanto no depende del nivel que se esté jugando, por ello se ha añadido como atributo del juego. Otros ejemplos similares son los estados, que heredan de Phaser.State. Los estados definen las diferentes pantallas del videojuego y los niveles.

La forma de almacenar el progreso mediante cookies es la siguiente. Aunque actualmente existen pocos niveles, se ha dejado implementado de forma que se pueda ampliar. Cada vez que se llega al estado *win* se actualiza la cookie escribiendo el número de nivel hasta el que se puede jugar. A continuación, en el estado *chooseLevel* se lee la cookie y se desbloquean los niveles correspondientes. Tienen una duración de dos días, después se pierde el progreso.

4. Pruebas

Se han realizado pruebas sobre varios aspectos. Primero, se ha comprobado que la implementación realizada mediante el módulo RequireJS funcionase correctamente ya que es el nexo de unión de nuestro proyecto. Más adelante pudimos probar que la transición entre estados fuese la correcta. Para ordenar de alguna manera el videojuego. Además, sobre cada clase se han realizado pruebas para verificar que

estaba correctamente acoplada al proyecto con RequireJS y que funcionaba si se le invocaba desde el estado del juego.

Otro módulo que ha sido necesario probar ha sido el relacionado con las cookies. Se ha ejecutado en diferentes navegadores para verificar que escribía y leía correctamente las cookies.

Los archivos de audio también varían entre navegadores y por tanto se han probado, así como los gráficos y el juego en general.

Guía de Juego

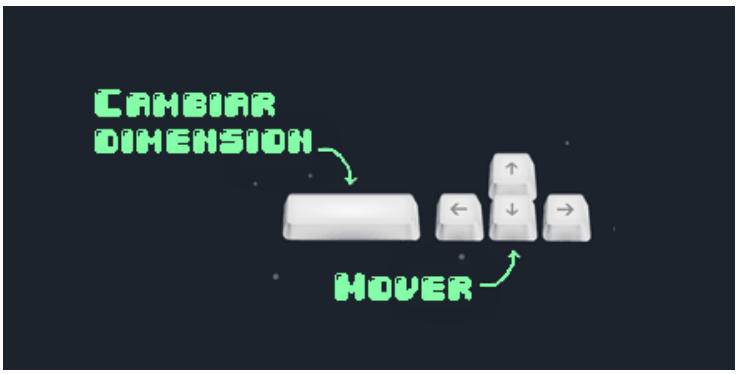
1. Objetivo

El objetivo principal es guiar a nuestro robot a lo largo de los distintos niveles con el fin de obtener el aparato (Imagen 1) que nos permitirá volver a encerrar al monstruo en su prisión. Una vez que hayamos llegado a él podremos jugar al siguiente nivel.

Imagen 1

2. Personaje y Controles

Nuestro héroe es un simpático robot al que podemos mover en todas direcciones (arriba, izquierda, derecha y abajo). Para ello (como se ilustra en la Imagen) utilizaremos las flechas del teclado. Para cambiar de dimensión en un determinado vórtice es necesario no solo estar sobre él sino también pulsar la barra espaciadora.



3. Enemigos

Se trata de personajes con los que el jugador puede interactuar y de los que recibe daño.

a. Monstruo

Nuestro mayor enemigo es el monstruo que habita en la dimensión oscura. Este tratará de encontrarnos en todo momento, con lo que hay que ser especialmente cuidadosos cuando no nos encontremos en la dimensión de origen ya que si nos encuentra y nos llega a alcanzar nos quitará todas las vidas, es decir, moriremos automáticamente.

b. Enemigos simples

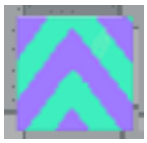
Estos están en ambas dimensiones, cuando choquemos con ellos perderemos una vida, lo que puede resultar peliagudo sin venimos de regreso de la dimensión oscura, donde seguramente perdamos alguna vida debido a la escasa visibilidad.

4. Trampas

A la hora de avanzar por los niveles debemos tener en cuenta distintos elementos que nos afectaran de forma distinta.

a. Suelo direccional

Esta trampa, a priori inofensiva, puede causarnos algún que otro quebradero de cabeza, ya que una vez estemos sobre la casilla el jugador seguirá la dirección que esta indica. De este modo podremos acabar en una trampa al final del camino o en el mejor de los casos, en una vida.



b. Proyectiles bombas

Cañones que cada cierto tiempo lanzarán un proyectil que, en el caso de alcanzarnos, nos quitará una vida.

c. Pinchos

Uno de los elementos más letales, se trata de un suelo con pinchos que aparecen y desaparecen. Habrá que ser especialmente cuidadoso y pasar por encima cuando estos no estén activos.

d. Llamas

Fuego colocado para disuadirnos de ir a ciertos lugares, si pasamos por encima de él nos quemaremos y perderemos una vida. Algunos botones (de los que hablaremos más adelante) apagan el fuego y nos permiten pasar sin problemas.

e. Suelo trampa

Junto al monstruo, es el elemento más mortal de todo el videojuego. Tenemos dos tipos: un suelo agrietado el, cuál se romperá del todo cuando pasemos por encima de él dejando al descubierto un agujero por el que caeremos, muriendo directamente. La otra es el suelo con el agujero, es decir no podremos pasar por encima de él.

5. Otros elementos del juego

No infringen ningún tipo de daño al jugador, y le ayudan a avanzar por el nivel. Existen los siguientes tipos de elementos interactivos.

A. Vortices dimensionales

Los vórtices nos permiten ir de un punto de una dimensión a otra pulsando la barra espaciadora. Esto es importante para el juego ya que para avanzar es necesario abrir algunas puertas que se abren con botones de otra dimensión.

B. Vidas

Se trata de baterías que aumentan en uno la vida del jugador. El máximo de vidas es tres, de este modo si el jugador ya tiene tres vidas no aumentará su contador.

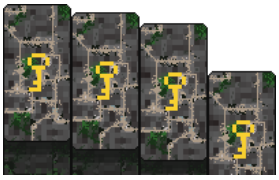
C. Puertas y botones

En algunas ocasiones veremos que nuestro camino esta obstaculizado por una puerta, en este caso tendremos que encontrar su correspondiente botón para abrirla. Además de haber botones que se presionan con el jugador, los hay también que tienen que ser pulsados por el monstruo, por lo que no debemos de pasarlos por alto.

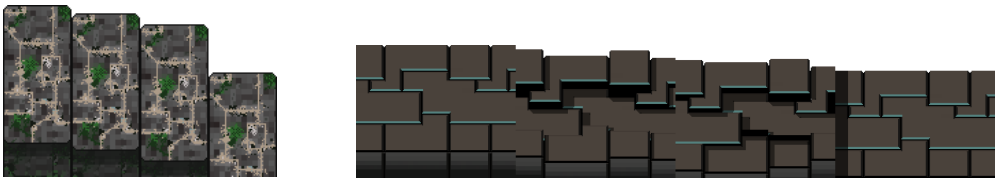
Botón monstruo

Diferentes ejemplos de botones de jugador

Existen varios tipos de puertas, aunque todas se abren mediante botones. Algunas se abren solo mediante botones de monstruo o normales y otras solo mediante botones de llave. Las puertas que se abren por botones llave llevan una marca de una llave dibujada para que quede más claro el tipo de botón que hay que buscar.



Puertas que necesitan botón llave



Puertas comunes

6. Niveles

Existen tres niveles actualmente, que se irán desbloqueando a medida que vayas avanzando por el juego. Los tres mundos o escenarios son distintos y variados.

Nivel 1

Nivel 2

Nivel 3 (dimensión oscura y normal)

Hasta aquí la guía del juego, esperamos que sea útil y que disfrutes de esta aventura.