

Infinite survival

Alejandro Moreno San Vidal
Pablo Pérez Manso

UAM - Videojuegos 2016

7 de diciembre de 2016 12:50

Índice general

I Descripción general del proyecto	2
I.1 Introducción	2
I.2 Descripción general del producto	2
I.3 Funcionalidad	2
I.3.1 Acciones	3
I.3.2 Controles	3
I.3.3 Estados	3
I.3.4 Mecánica del juego	4
I.4 Entorno y herramientas de desarrollo	4
I.5 Equipo y lugar de trabajo	5
I.6 Recursos adicionales	5
II Gestión del Proyecto	6
II.1 Tareas a realizar	6
II.2 Entregables	6
III Análisis y diseño	7
III.1 Modelo de ciclo de vida	7
III.2 Diseño	8
III.3 Interfaz del videojuego	9
IV Implementación	10
V Pruebas	11
A Planificación temporal	12

Capítulo I

Descripción general del proyecto

I.1. Introducción

Este documento contiene las especificaciones de requisitos y el diseño del plan de trabajo del proyecto que se realizará por Alejandro Moreno San Vidal y Pablo Pérez Manso para la asignatura de Introducción a la Programación de Videojuegos.

I.2. Descripción general del producto

Este videojuego se desarrollará durante las prácticas de la asignatura, y su diseño y desarrollo se especifican en este documento. El videojuego será una combinación de las mecánicas que se observan en juegos tipo Endless Run, como son el caso de Zombie Tsunami o el popular Sonic, en la cual un personaje se mueve horizontalmente por la pantalla intentando recoger premios y evitando las cosas que le pueden matar.

I.3. Funcionalidad

El videojuego contiene actualmente las siguientes pantallas:

- Pantalla de inicio
- Pantalla de juego

Cada pantalla contiene las siguientes características y funcionalidades:

- Pantalla de inicio
 - Logo del juego
 - Título del videojuego
- Pantalla de juego
 - Muestra la partida que se está jugando actualmente
 - Los elementos que contiene esta pantalla están determinados más adelante

I.3.1. Acciones

La pantalla del videojuego constará de un personaje que se mueve horizontalmente (de forma automática) y de izquierda a derecha y la acción que actualmente realiza es la siguiente:

- **Saltar:** se producirá el salto para evitar enemigos o para salvar las caídas al vacío que puedan aparecer.

I.3.2. Controles

El control para el salto será:

- **Saltar:**
 - Un toque a la tecla de saltar hará un salto corto
 - Una pulsación larga hará que el salto sea más largo (aunque no influirá en la altura del salto)

I.3.3. Estados

I.3.3.1. Estados del juego

Los estados del juego para esta entrega son los siguientes:

- **Menú inicial**
 - Es el estado inicial de nuestro juego
 - Desde este menú se pueden acceder al resto de las pantallas: juegos, opciones, tablero...
- **Pantalla de juego**
 - En esta pantalla se producen todas las acciones del juego y su historia. Se muestra la pantalla de juego (véase 3.3 Interfaz de videojuego)
- **Death**
 - Se mostrará una pantalla de game over

I.3.3.2. Estados del personaje

El personaje puede encontrarse en los siguientes estados

- **Corriendo**
 - Es el estado que se mantendrá la mayor parte del tiempo
 - Desde este estado se puede llegar a cualquiera de los otros
- **Saltando**

- Se entra en este estado cuando se presiona la tecla de saltar
- Se sale automáticamente al tocar un objeto o al volver a pulsar el botón de salto (en el caso de un salto sencillo)
- Muerto
 - Se llega a través del resto de los estados
 - Indicará que se ha acabado la partida y que se tendrá que lanzar la pantalla de fin de partida

I.3.4. Mecánica del juego

La mecánica del juego consiste en una carrera continua automática sobre la que el jugador no tendrá ninguna capacidad de cambiar. Éste debe utilizar los comandos explicados anteriormente para esquivar los problemas que se le puedan ir planteando a lo largo del juego. Así mismo, el jugador debe cubrir la distancia más larga para conseguir una puntuación mayor así como recoger objetos que le puedan ayudar durante el juego.

La dificultad del juego residirá en que durante la carrera, se irán poniendo diferentes travas para que la dificultad vaya en aumento:

- La **velocidad aumentará** de modo que los nuevos enemigos y la pantalla aparecerá cada vez más rápido dando menos tiempo al jugador a reaccionar ante estos imprevistos, siendo los reflejos clave para poder continuar jugando.
- La **densidad de obstáculos aumentará**, de modo que cada vez haya más enemigos y cada vez más juntos siendo necesario aplicar saltos más complicados o tener que calcular de una forma más precisa las caídas de los saltos.

Además durante el juego se le ofrecerán al jugador mejoras que podrá ir cogiendo durante el juego, tales como powerups que le harán más fácil la partida o le añadirán características interesantes durante el desarrollo del juego.

El juego terminará si la vida del personaje tras chocharse con objetos llega a cero, siendo esta recuperable con diferentes objetos del juego. Además también se terminará cuando la pantalla coma al jugador, es decir, que cuando un objeto se interponga en el avance del jugador y éste desaparezca por el lado izquierdo de la pantalla.

I.4. Entorno y herramientas de desarrollo

El juego va a ser desarrollado bajo la plataforma Phaser.io. Esta plataforma ha de ser trabajada con el lenguaje de programación Javascript y mediante un servidor web. Las herramientas que hemos utilizado son las siguientes:

- **Linux:** Utilizamos un entorno de desarrollo linux junto con las herramientas de texlive para la generación de documentos mediante el lenguaje \LaTeX .
- **Git y Github:** como control de versiones el proyecto lo sostenemos sobre git, ya que permite una estructura descentralizada de desarrollo en la que se puede editar en concurrencia. Además, para mantener un backup online y una forma de

sincronizar código, utilizamos la plataforma github, así como registro online del trabajo que vamos desarrollando.

- **Adobe Photoshop e Illustrator:** con la intención de desarrollar los diferentes aspectos gráficos del juego usamos estos programas con el fin de hacer correcciones o crear nuevos gráficos para el juego.
- **Adobe Audition:** edición de los diferentes temas musicales y efectos sonoros del juego.
- **Sublime Text, Brackets y/o vim:** ya que el lenguaje que se utiliza no tiene ningún entorno de desarrollo establecido y que ofrezca ventajas significativas, usamos un editor de código sencillo para el desarrollo del juego.
- **Brackets y XAMPP:** dada la necesidad de necesitar un servidor web que se encargue de los diferentes aspectos del juego, éstos son los escogidos para la realización de las pruebas.

Como la mayoría de estos programas son gratuitos o se disponen de las diferentes licencias necesarias para el desarrollo de todo el sistema, el coste en herramientas del desarrollo del proyecto será gratuito.

I.5. Equipo y lugar de trabajo

El lugar de trabajo que mas hemos empleado,han sido los laboratorios de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid, ya que todos disponen en su mayoría de los requisitos de software y hardware descritos anteriormente. Así mismo, también hemos hecho uso de nuestros propios ordenadores.

I.6. Recursos adicionales

En lo referente a recursos hemos utilizado:

- **Sprites:** personajes y diferentes objetos cogidos de presets ya creados por otros usuarios de internet.
- **Menús y fondos:** al igual que los sprites hemos realizado búsquedas por internet y procedido a la generación de los mismos.
- **Tipografías:** las diferentes tipografías utilizadas en el juego han sido extraídas de Google Fonts.

Capítulo II

Gestión del Proyecto

En esta sección se mostrarán los aspectos de la planificación del proyecto, desde tareas, plazos de entrega y la línea temporal, hasta los entregables y su contenido.

II.1. Tareas a realizar

El gráfico con las tareas del desarrollo del proyecto y su plan temporal se encuentran en los anexos de este documento. Si se necesita más información se puede recurrir al capítulo 3 de este documento en el que se especifica más del Análisis y Diseño.

II.2. Entregables

Actualmente nos encontramos en la segunda entrega del proyecto, y quedaría una versión final del mismo:

- **Prototipo del proyecto:** En este prototipo, podemos probar un funcionamiento básico del mismo. El jugador podrá recoger monedas e intentar superarse en la distancia máxima recorrida. Para ello, por el momento solo podrá disponer de la habilidad de salto sin ningún tipo de powerup, los cuales incluiremos para la versión final del proyecto.
- **Entrega final:** se entregará una versión del desarrollo completamente operativa y con todas las funcionalidades previamente descritas ya implementadas. Así mismo se entregará una memoria de todo el desarrollo del proyecto y un manual de uso del juego en el que se expondrán todos los aspectos relevantes del juego. Su entrega se preve con la finalización del plazo.

Capítulo III

Análisis y diseño

Procederemos con los temas referentes a la planificación del proyecto, incluyendo el ciclo de vida del software, planteamiento de diseño a seguir y los diagramas que nos permitirán desarrollar e implementar todas las funcionalidades y aspectos descritos en el capítulo 1.

III.1. Modelo de ciclo de vida

Para el ciclo de vida de nuestro proyecto se tomará un diseño basado en cascada con realimentación, de modo que nos permite rápidamente volver a la fase anterior para corregir los errores que se pudieran haber cometido. Constará de las siguientes fases:

1. **Análisis:** se analizan y se definen todas las funcionalidades y los requisitos que necesitará el proyecto, de modo que se pueda tener claro en la fase de diseño qué debería recoger.
2. **Diseño:** se escoge cual es el mejor método para llevar a cabo el proyecto en base a las especificaciones que se observaron en la fase de análisis y se plasman en una serie de documentos. Esto incluirá la planificación temporal, funcionalidades y forma de implementarlas y todos los demás aspectos del proyecto.
3. **Implementación:** se comenzará con el desarrollo de toda la parte técnica del proyecto, de modo que sea coherente con todos los detalles que se planificaron previamente en la fase de diseño. Así mismo se podrán generar prototipos de todo el proyecto o de partes de él para que puedan pasar a la siguiente fase.
4. **Pruebas:** una buena parte del desarrollo es probar que todo lo anteriormente implementado funcione como se especificó en la parte de diseño. En esta fase se podrían incluso mostrar diversos prototipos al cliente final para que pueda observar el estado del proyecto. En caso de un error detectado se notificará y se devolverá a la fase de implementación o diseño, dependiendo del error.
5. **Entregas:** una vez que los prototipos que ya se planificaron en la fase de diseño, pasen la fase de pruebas se realiarán diferentes entregas de material al cliente para que se pueda ir evaluando el trabajo que se está desarrollando y que éste localice diferentes errores que como cliente no desee o no quedaran claras en la fase previa al análisis.

6. **Entrega del producto y mantenimiento:** una vez terminada toda las fases de desarrollo del proyecto se procederá con la entrega del mismo al cliente, y se mantendrá un periodo de mantenimiento en el que el cliente podrá notificar posibles errores o mejoras que se podrán implementar.

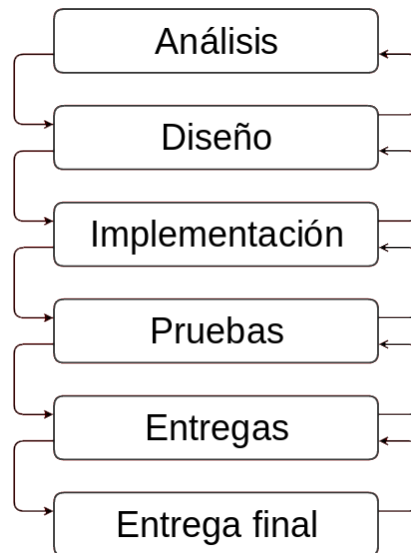


Figura III.1: Cascada con realimentación

III.2. Diseño

Para poder proceder correctamente con la implementación del proyecto es necesario realizar un diseño modular de todo el código del videojuego. Se diseñarán las clases del código a implementar que darán forma a los elementos del videojuego.

En esta sección se especificarán las implementaciones de la mayoría de los elementos descritos en la primera sección del documento, por lo que es recomendable que se consulte como referencia si en alguna parte de esta sección se tienen dudas.

Todo el videojuego será controlado por una clase Main que contendrá todo el resto de los elementos del sistema. Esta contendrá al jugador, powerups(objetos que el jugador podrá obtener), los enemigos (elementos perjudiciales para el jugador) y el mapa.

- El jugador tendrá entidad única para el mismo debido a su diferencia con el resto de instancias del juego. Además se le podrán cambiar diferentes aspectos de su física natural en base a los objetos que pueda ir recogiendo.
- El mapa se irá generando automáticamente o cargando de ficheros externos para poder ir dibujando todo el sistema de juego.

Además se realizarán clases e interfaces que resuelvan las diferentes acciones que puedan ocurrir en el juego, como las colisiones entre objetos, los controles por parte de los diferentes dispositivos, así como las conexiones con el servidor de puntuaciones.

III.3. Interfaz del videojuego

La parte del videojuego que el cliente termina viendo es una de las más importantes a la hora del desarrollo del videojuego, por lo que en esta sección se pretende dar una idea de la composición de la interfaz de usuario. Se mostrará un ejemplo de otro juego con una interfaz similar a la que se quiere implementar.

Los elementos que se mostrarán son los que siguen:

- **Fondo:** se mostrará un fondo que refleje la situación del personaje dependiendo del traje que lleve en cada momento, de forma que pueda hacer que el usuario tenga una inmersión en el videojuego mayor.
- **Personaje:** el personaje principal será una imagen que por norma general se encontrará en el centro del escenario. Dependiendo de diferentes mejoras del juego se podrá cambiar la imagen del mismo para poder reflejar mejor las mejoras que pueda haber conseguido.
- **Barra superior:** en esta se mostrarán los principales indicadores del juego para que el jugador las pueda ver:
 - **Distancia:** distancia (puntuación) recorrida por el personaje en la partida
- **Objetos recogidos:** se pueden ir recogiendo objetos por el juego que luego se podrán usar con diferentes acciones. Para poder conocer que objetos tienes disponibles se dispondrán en una tira en un lado de la pantalla.

La interfaz sería parecida a la siguiente:



Figura III.2: Interfaz similar a la descrita

Capítulo IV

Implementación

Para facilitar la creación y la relación entre las clases definidas en la fase de diseño, se ha considerado el uso de patrones de diseño. Además se tratará de que el diseño sea lo más modular posible.

Se van a usar dos tipos de patrones: de creación, para ayudar a crear los nuevos objetos en tiempo de ejecución y estructurales, para mejorar la organización del código.

Para los patrones de creación se considera el siguiente:

- **Singleton:** nos permite que algunas de las clases solo puedan ser creadas una vez de modo que su control es mucho más fácil por el resto de las clases. Sería los casos del juego o del personaje.

En cuanto a los patrones estructurales se considera el siguiente:

- **Modelo Vista Controlador:** ayuda a organizar la lógica interna de modo que se separa por completo la parte de la interfaz de los datos que se manejan. De este modo se crea más independencia entre cada uno de los módulos pudiendo cambiar cada uno de ellos sin necesidad de cambiar el otro.

En lo que respecta a las fases de la implementación se consideran tres fases:

- **Lógica:** nos referimos a la parte de toda la codificación de la parte lógica del videojuego, personajes, objetos, pantalla...
- **Gráficos:** búsqueda de posibles materiales gráficos para la temática del juego y su adaptación a las necesidades del proyecto.
- **Interfaz:** una vez finalizada la parte de gráficos, construcción de la interfaz basada en esos gráficos y su unión con la lógica del juego.

Capítulo V

Pruebas

Para una correcta integración de todas las partes del proyecto, se deben realizar pruebas para que una vez que se realice la integración no haya ningún problema.

- **Lógica:** se hará una comprobación de que todas las clases tienen un funcionamiento correcto y más tarde se probará su funcionalidad conjunta. De este modo nos aseguraremos que esta parte funciona de una forma correcta para que al añadir la parte de interfaz se minimicen los errores.
- **Gráficos:** se cargarán todos los sprites, dibujos, imágenes y sonidos sobre una instancia de phaser nueva para comprobar que funcionan de un modo de adecuado con la plataforma al ser cargados.
- **Interfaz:** se probarán la cohesión que tienen la creación de la parte gráfica con la lógica del videojuego. Se comprobará que la comunicación entre la lógica y la interfaz se produce de manera correcta, viéndose en la pantalla lo que corresponde por la parte del backend. Además se deberán hacer diferentes pruebas de velocidad del sistema para corregir errores de implementación a la hora de pintar la interfaz.

Cabe destacar, que debido a la complejidad de algunos de los componentes de la parte lógica, su correcto funcionamiento será comprobado mediante su salida por la interfaz, de modo que se comprueba a la vez parte de la lógica y parte de la interfaz.

Apéndice A

Planificación temporal

Nombre	Duración	Inicio	Terminado
Diseño	12 days?	6/10/16 8:00	21/10/16 17:00
Análisis de requisitos	3 days?	6/10/16 8:00	10/10/16 17:00
Diseño	9 days?	11/10/16 8:00	21/10/16 17:00
1º entrega	0 days	21/10/16 8:00	21/10/16 8:00
Implementación	36 days?	24/10/16 8:00	12/12/16 17:00
Backend	18 days?	24/10/16 8:00	16/11/16 17:00
Controles	5 days?	24/10/16 8:00	28/10/16 17:00
Interacciones	6 days?	24/10/16 8:00	31/10/16 17:00
Juego	14 days?	27/10/16 8:00	15/11/16 17:00
Objetos	8 days?	7/11/16 8:00	16/11/16 17:00
Interfaz	18 days?	17/11/16 8:00	12/12/16 17:00
2º entrega	0 days?	27/11/16 8:00	28/11/16 17:00
Pruebas	10 days?	13/12/16 8:00	26/12/16 17:00
Lógica	4 days?	13/12/16 8:00	16/12/16 17:00
Interfaz	5 days?	17/12/16 8:00	23/12/16 17:00
3º entrega	0 days	24/12/16 8:00	26/12/16 17:00

Figura A.1: *Tiempos de cada tarea*

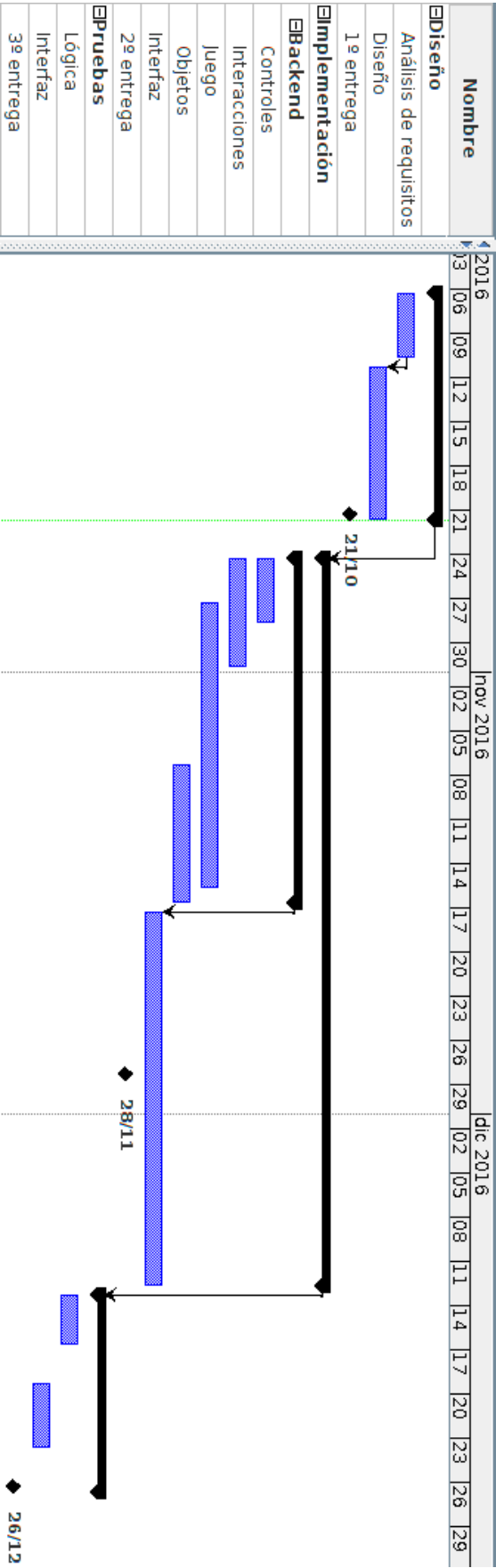


Figura A.2: Diagrama de Gant