



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería
Informática

Fastastic Roads
Documentación Técnica



Presentado por Alejandro Goicoechea Román
en Universidad de Burgos — 8 de julio de 2021
Tutores: Jesús María Alonso Abad y
Mario Alaguero Rodríguez

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	4
Apéndice B Especificación de Requisitos	9
B.1. Introducción	9
B.2. Objetivos generales	9
B.3. Catalogo de requisitos	9
B.4. Especificación de requisitos	12
Apéndice C Especificación de diseño	21
C.1. Introducción	21
C.2. Diseño de datos	21
C.3. Diseño procedimental	22
C.4. Diseño arquitectónico	23
C.5. Diseño de interfaces	25
Apéndice D Documentación técnica de programación	29
D.1. Introducción	29
D.2. Estructura de directorios	29

D.3. Manual del programador	30
D.4. Compilación, instalación y ejecución del proyecto	31
D.5. Pruebas del sistema	33
Apéndice E Documentación de usuario	35
E.1. Introducción	35
E.2. Requisitos de usuarios	35
E.3. Instalación	36
E.4. Manual del usuario	36
Bibliografía	47

Índice de figuras

A.1. Gráfico del ratio de horas estimadas vs. horas reales empleadas.	3
B.1. Diagrama de casos de uso	12
C.1. Diagrama de clases de “Race Controller” y “Checkpoint”	21
C.2. Diagrama de secuencias de “Fastastic Roads”	22
C.3. Diagrama de objetos del grafo de Checkpoints.	23
C.4. Estructura del patrón de comportamiento estrategia.	24
C.5. Estructura del patrón creacional método plantilla.	25
C.6. A pesar de la carencia de colores y un diseño atractivo, un usuario medio es capaz de entenderlo fácilmente.	26
C.7. Los escenarios se han trabajado desde un principio, permitiendo mejoras a medida que se avanza con el proyecto.	27
C.8. El logotipo de “Fastastic Roads” apela a la estética industrial del videojuego.	28
E.1. Menú principal de “Fastastic Roads”.	37
E.2. Instrucciones para jugar.	38
E.3. Selector de modo de juego (solo disponible el modo a contrarreloj).	39
E.4. Selector de personaje (solo disponible el personaje de Pilar Careaga).	40
E.5. Selector de circuito para jugar (solo uno disponible).	41
E.6. Cuenta atrás antes de empezar la carrera.	41
E.7. Los puntos de control se distinguen por su transparencia ligeramente translúcida.	42
E.8. Las flechas indican por dónde debe ir el jugador.	43
E.9. En algún punto del circuito puede haber bifurcaciones por las que circular.	43

E.10.Al pasar de nuevo por meta, se actualiza el número de vueltas y el mejor tiempo.	44
E.11.Hay posibilidad de pausar la partida dándole a la tecla de Escape.	45
E.12.Al finalizar, se bloquea el manejo del vehículo y se salta a la pantalla de estadísticas.	45
E.13.En la pantalla de créditos aparecen los participantes en el desa- rrollo del proyecto.	46

Índice de tablas

A.1. Tabla de costes fijos	5
A.2. Tabla de costes directos	5
A.3. Tabla de costes indirectos	6
A.4. Herramientas con sus respectivas licencias.	7
B.1. CU-01 Jugar	13
B.2. CU-02 Seleccionar modo	13
B.3. CU-03 Seleccionar personaje	13
B.3. CU-03 Seleccionar personaje	14
B.4. CU-04 Seleccionar circuito	14
B.5. CU-05 Conducir vehículo	14
B.6. CU-06 Acelerar vehículo	15
B.7. CU-07 Decelerar vehículo	15
B.8. CU-08 Girar vehículo	16
B.9. CU-09 Recolocar vehículo	16
B.10.CU-10 Partida contrarreloj	16
B.10.CU-10 Partida contrarreloj	17
B.11.CU-11 Conteo de vueltas	17
B.12.CU-12 Cronómetro de tiempo	17
B.12.CU-12 Cronómetro de tiempo	18
B.13.CU-13 Indicador de mejor tiempo	18
B.14.CU-14 Instrucciones	18
B.15.CU-15 Créditos	19
B.16.CU-16 Salir	19
B.17.CU-17 Salir de la partida	19
B.17.CU-17 Salir de la partida	20
B.18.CU-18 Volver a la pantalla anterior	20
B.19.CU-18 Pausar partida	20

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En todo proyecto se requiere de una planificación debidamente organizada, con el fin de cumplir plazos, presupuestos, obligaciones legales y otros aspectos de manera adecuada.

Es por ello que lo ideal es seguir una metodología de planificación adaptada al proyecto, con la que se puedan aprovechar de la mejor forma posible los recursos disponibles en el tiempo marcado de cara a la correcta finalización, tanto de este Trabajo de Fin de Grado como de cualquier otro tipo de proyecto.

A.2. Planificación temporal

Inicialmente, se plantearon distintas metodologías de desarrollo *software*, entre ellas Kanban y SCRUM.

Como se explica en el capítulo de “Técnicas y Herramientas” en el documento de memoria, Kanban permite un trabajo más dinámico y fluido, con un proceso evolutivo e incremental enfocado al desarrollo de tareas pendientes. Es más adecuado para los procesos de integración continua, en las que no se tiene periódicamente entregas o microentregas que hacer y se definen una serie de tareas para esas entregas, sino que se van tomando esas tareas, luego en función de su prioridad se van haciendo y finalmente se van validando tan pronto como estén antes de pasar a la siguiente tarea.

Debido a la magnitud del proyecto, se necesitaron ciclos mucho más rápidos de revisión de las tareas para poder ir avanzando y que el impacto de cualquier “cuello de botella” fuese el menor posible.

Mediante un tablero de proyecto con cuatro columnas, correspondientes a “*To do*” o “Por hacer”, “*In progress*” o “En progreso”, “*QA/Testing*” o “Control de calidad” y “*Done*” o “Hecho”, el flujo de trabajo ha funcionado de la siguiente manera:

- En la columna de “***To do***” se han ido creando las tareas a medida que se iban necesitando realizarlas. Muchos elementos en el proyecto no son necesarios hasta que se realizan otras tareas previas, a la vez que pueden ir surgiendo a medida que van necesitándose en el momento o acaban siendo pensadas. La tarea correspondiente tendrá una descripción, una duración estimada de tiempo que puede conllevar realizarla y una duración real en la que se marca el tiempo que se ha tardado en realizarla.
- En “***In progress***” se tienen todas las tareas que se están realizando. Puede haber varias a la vez, pues se pueden compaginar tengan correlación entre ellas o no, aunque lo ideal es dedicarse a unas pocas que, al menos, dependan entre ellas para no tener una carga de trabajo que no se pueda abarcar. Las tareas irán asignadas a uno o varios usuarios que estén realizándolas.
- En la columna de “***QA/Testing***” se tienen todas las tareas tentativamente completadas y listas para su validación. El usuario habrá marcado el tiempo que le ha llevado realmente realizarla. Si cumple el objetivo de la tarea correctamente, se da por finalizada. En caso de tener fallos o de requerir cambios, se devuelve a “En progreso”, devolviendo la asignación al usuario o usuarios encargados de la misma.
- Finalmente, en “***Done***” se tienen todas las tareas dadas por finalizadas. El control de calidad habrá decidido que está correctamente terminada.

Una idea fundamental de Kanban es medir los tiempos. En este proyecto, aunque no se ha hecho uso de SCRUM, se han empleado *sprints* de dos semanas de duración para hacer una división de tareas por tiempo, calculando el tiempo empleado en cada una de ellas.

Con estos datos, por una parte, puede ser un punto de referencia para ajustar en el futuro las estimaciones que se hagan, y por otra para aquellas personas que realicen hacer proyectos similares, al haber utilizado esta taxonomía de tareas, pueden tomar este histórico de valores como referencia

para el coste de cuánto supondría una tarea referida a ese tipo de tareas (si son tareas de mejora, calcular cuánto se podría tardar).

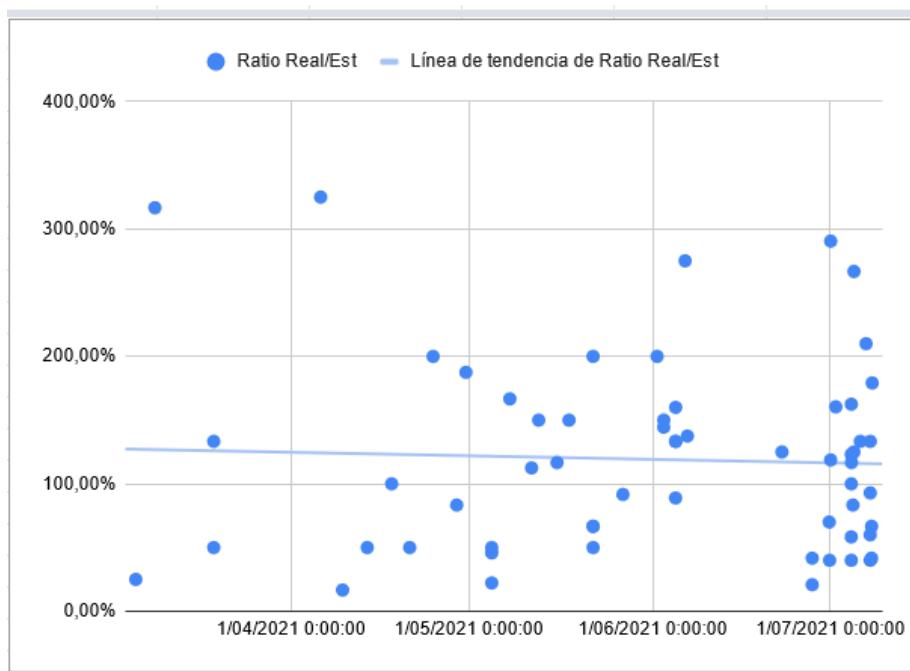


Figura A.1: Gráfico del ratio de horas estimadas vs. horas reales empleadas.

El tiempo estimado en total para las tareas ha sido de aproximadamente 170 horas, y las horas reales empleadas han sido aproximadamente 267. El ratio por tareas se puede comprobar en la gráfica previa. Esta gran diferencia entre las distintas tareas, así como entre la duración calculada total y la real total, viene dada por la magnitud del proyecto y la inexperiencia al respecto. Un videojuego se compone de una gran cantidad de elementos y funciones que no siempre son fácilmente implementables, y la gran mayoría de veces que puede aparentar ser sencillo acaba llevando más tiempo por la investigación adicional correspondiente, la corrección de fallos, la posterior optimización, etc. Es por ello que Kanban ha sido la metodología de planificación ideal, permitiendo esa flexibilidad ante cálculos incorrectos y otros imprevistos. De igual manera, el tiempo total estimado y el tiempo total medido no son realistas, porque en las primeras etapas no se incluyeron las tareas de investigación y aprendizaje, y cuando se empezaron a incluir, no se estimaron inicialmente, de manera que no han quedado reflejadas en ninguna parte. En la carpeta de la documentación se encuentra el fichero “estimaciones-issues.csv” con los datos correspondientes.

Previa a la planificación, hubo una gran cantidad de tiempo dedicada al aprendizaje respecto al uso de Unity, sus funcionalidades y otros aspectos, como se indicó en la memoria. La mayor parte de las tareas del proyecto fueron dirigidas a la implementación de las nuevas funciones, así como a la corrección, mejora y optimización de las mismas.

A.3. Estudio de viabilidad

En todo proyecto se requiere de un estudio de viabilidad para asegurar su realización completa. Es necesario analizar su viabilidad económica, con los costes del proyecto que puede conllevar, así como su viabilidad legal, haciendo mención a las licencias que pueda tener.

Viabilidad económica

Se dividen entre costes del proyecto y beneficios obtenidos.

Costes

Los costes son aquellos gastos que conlleva realizar algunas actividades, así como la adquisición de materiales y otros aspectos. Se divide en costes fijos y costes directos e indirectos [1].

Costes fijos

Los costes fijos son aquellos costes que no cambian en función del volumen de actividad, i.e., aquellos que no varían ajenos al proyecto, siendo el más destacable el de los salarios. Estos datos son calculados en base a 4 meses de trabajo.

Concepto	Coste
Contrato de Luz	180,00 €
Servicio de Internet	200,00 €
Salario x 4 meses	9251,12 €
Salario mensual bruto	2312,78 €
Retenciones por IRPF (24 %)	252,00 €
Cuotas a la Seguridad Social (28,30 %)	321,30 €
Salario mensual neto	1050,00 €
TOTAL	9631,12 €

Tabla A.1: Tabla de costes fijos.

Costes directos

Los costes directos son los que vienen derivados del desarrollo del proyecto (contratación de servicios para el alojamiento del proyecto, determinado *software* necesario para el mismo, *marketing* y otros).

Concepto	Coste
Impresión de Memoria	30,00 €
TOTAL	30,00 €

Tabla A.2: Tabla de costes directos.

Costes indirectos

Los costes indirectos son aquellos que no dependen del desarrollo del proyecto de manera directa (licencias *software* adquiridas, componentes *hardware* y otros). El cálculo de los costes amortizados se ha realizado tomando como base los 4 meses de trabajo mencionados atrás a lo largo de un plazo de 4 años.

Concepto	Coste	
	Coste	amortizado
Ordenador personal	1999,00 €	166.58 €
Licencia Windows 10 Home	145,00 €	12,08 €
TOTAL	2144,00 €	178,66 €

Tabla A.3: Tabla de costes indirectos.

Beneficios

Al tratarse de un proyecto de carácter educativo sin un objetivo de mercado, no se obtiene ningún beneficio económico de explotación. No obstante, tantea ciertos escenarios que, en el futuro con nuevos proyectos, será necesario explorarlos

Viabilidad legal

Para poder distribuir el *software*, se tienen que cumplir determinadas obligaciones legales. La más importante tiene que ver con las licencias, por lo que las nombraremos a continuación.

Licencias

El proyecto no tiene una finalidad comercial. No obstante, esto no lo despoja de derechos de autor y distribución. Es por ello que la licencia de distribución escogida ha sido “Creative Commons”, concretamente la licencia de “Reconocimiento - No Comercial - Sin Obra Derivada 2.5 España” (CC BY-NC-ND 2.5 ES)¹. Esta licencia permite compartir la obra en cualquier medio o formato bajo el reconocimiento de la autoría del producto, sin que se pueda utilizar para una finalidad comercial y sin posibilidad de difundir material remezclado o transformado a partir de este.

Respecto al resto de *software* utilizado, estas son sus correspondientes licencias:

¹Consultable en: <https://creativecommons.org/licenses/by-nc-nd/2.5/es/>

Herramienta	Descripción	Licencia
Unity	Motor de videojuegos.	Gratis/Privativo
Blender	Software de modelaje.	GPL
Texmaker	Editor multiplataforma de texto.	GPL 2
Git	Repositorio de código.	GPL 2
GitHub	Repositorio de código.	Propietaria/Privativa
StarUML	Herramienta de diseño UML.	Propietaria/Privativa

Tabla A.4: Herramientas con sus respectivas licencias.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se detallan los objetivos del proyecto y los requisitos que debe cumplir el *software* desarrollado. Estos requisitos se han indicado al inicio del proyecto, pero a lo largo de su desarrollo se han ido añadiendo otros adicionales, por lo que aquí se recogen y explican todos adecuadamente. Parcialmente, se han seguido indicaciones del estándar IEEE 830-1998 para la especificación de requisitos.

B.2. Objetivos generales

Los objetivos que se pretenden alcanzar en este proyecto son los siguientes:

- Desarrollar un videojuego de carreras para PC, compatible con Windows y Linux.
- Ofrecer una buena jugabilidad y accesibilidad al usuario, con una curva de aprendizaje sencilla.
- Aprender a usar Unity, sus funcionalidades y la programación en el lenguaje que le corresponde.
- Asentar unas bases para proyectos similares.

B.3. Catalogo de requisitos

En este apartado se detallan los requisitos funcionales y no funcionales de la aplicación desarrollada.

Requisitos funcionales

- **RF-1 Jugar:** La aplicación debe permitir iniciar una partida.
 - **RF-1.1 Seleccionar modo:** El usuario debe poder escoger entre los modos de juego disponibles.
 - **RF-1.1.1 Partida contrarreloj:** La aplicación debe poder crear una partida de carrera a contrarreloj.
 - ◊ **RF-1.1.1.1 Conteo de vueltas:** En la partida debe haber un contador de vueltas que indique el número de vuelta en el que se ubica el usuario, así como el número total de vueltas a realizar.
 - ◊ **RF-1.1.1.2 Cronómetro de tiempo:** En la partida debe haber un cronómetro que cuente el tiempo realizado en una vuelta. Este cronómetro debe ponerse a cero cada vez que, habiendo atravesado todos los puntos de control previamente, se atraviese la línea de meta.
 - ◊ **RF-1.1.1.3 Indicador de mejor tiempo:** Cada mejor tiempo realizado al completar una vuelta debe ser actualizado en un letrero. En caso de no realizarse un tiempo mejor, se dejará con el último mejor tiempo. Cada vez que se empiece una nueva partida debe estar restaurado a 0.
 - **RF-1.2 Seleccionar personaje:** El usuario debe poder escoger entre las diferentes personajes disponibles para jugar.
 - **RF-1.3 Seleccionar circuito:** El usuario debe poder escoger entre los diferentes escenarios disponibles en los que realizar la carrera.
 - **RF-1.4 Conducir vehículo:** El usuario debe poder conducir un vehículo escogido previamente.
 - **RF-1.4.1 Aceleración del vehículo:** El vehículo debe poder acelerar a petición del usuario. Debe poder combinarse en su uso simultáneo con el giro.
 - **RF-1.4.2 Deceleración del vehículo:** El vehículo debe poder decelerar e ir en dirección marcha atrás a petición del usuario. Debe poder combinarse en su uso simultáneo con el giro.
 - **RF-1.4.3 Giro del vehículo:** El vehículo debe poder girar hacia el lado izquierdo o hacia el lado derecho a petición del usuario. Debe poder combinarse en su uso simultáneo con la aceleración o la deceleración.

- **RF-1.4.4 Recolocación del vehículo:** El vehículo debe poder ser recolocado manualmente en un punto de control previo a petición del usuario. En caso de caer al vacío, debe poder ser recolocado automáticamente.
- **RF-2 Instrucciones:** La aplicación debe ofrecer al usuario la posibilidad de consultar unas instrucciones de juego para
- **RF-3 Créditos:** La aplicación ha de ofrecer la posibilidad de conocer al usuario quiénes han realizado la aplicación en los distintos aspectos que la componen.
- **RF-4 Salir:** El usuario debe poder salir de la aplicación mediante una opción ofrecida por la misma.
 - **RF-4.1 Salir de la partida:** El usuario debe poder salir de la partida en curso mediante una opción en el menú.
- **RF-5 Volver a la pantalla anterior:** La aplicación debe ofrecer la posibilidad de volver a la pantalla anterior al menú en el que se encuentra el usuario.
- **RF-6 Pausar partida:** La aplicación debe ofrecer la posibilidad de pausar la partida a petición del usuario, así como continuarla una vez pausada.

Requisitos no funcionales

Estos requisitos se han obtenido en base a las características el estándar internacional para la evaluación de la calidad del software.

- **RNF-1 Usabilidad:** La aplicación debe ser sencilla de jugar y divertida. La curva de aprendizaje ha de ser baja para que el usuario no desista en comprender el funcionamiento y mejorar, a la vez que tiene que ser capaz de divertirse gracias a las mecánicas del videojuego. Por ello, la interfaz es amigablede cara al usuario.
- **RNF-2 Eficiencia:** La aplicación debe poder ser ejecutada en computadoras de gama media, con tiempos de carga entre pantallas aceptables y una jugabilidad fluida sin ralentizaciones de *frames*. Además, debe poderse incrementar los recursos de los que dispone la aplicación realizando distintas mejoras que impliquen una mayor eficiencia.
- **RNF-3 Multiplataforma:** La aplicación debe poder ser distribuida para sistemas operativos Windows y Linux, con su correspondiente correcta ejecución en los mismos.

- **RNF-4 Facilidad de mantenimiento:** Tiene que asegurar la posibilidad de ser extensible (pudiendo añadir funcionalidades nuevas al videojuego), modificable y corregible de manera sencilla.
- **RNF-5 Portabilidad:** Debe ser fácilmente transferible y adaptable a nuevas plataformas de juego.

B.4. Especificación de requisitos

En esta sección se muestra el diagrama de casos de uso de la aplicación, así como el desglose de cada uno de esos casos.

Diagrama de casos de uso

El diagrama de casos de uso corresponde al siguiente mostrado:

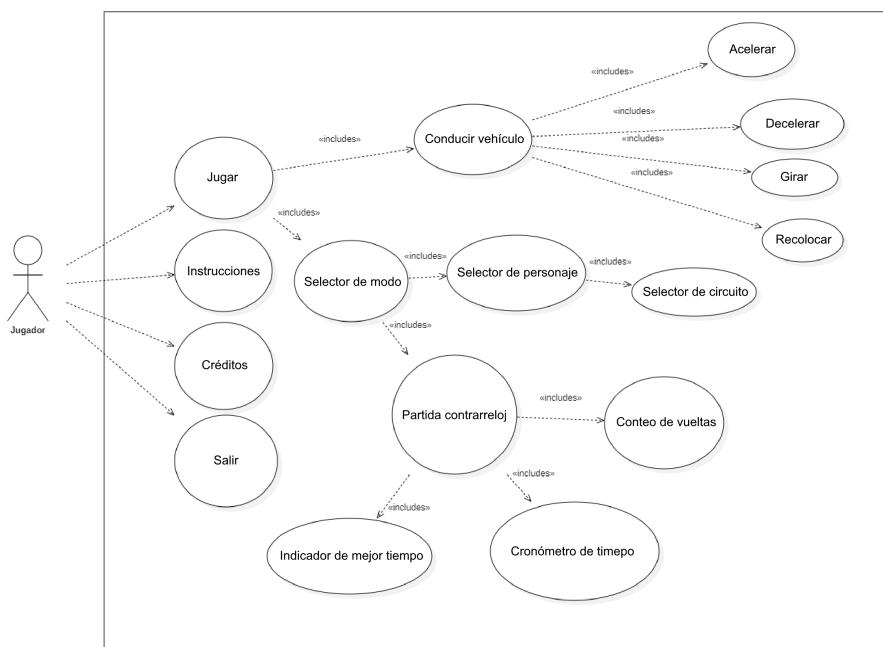


Figura B.1: Diagrama de casos de uso

Actores

El número de actores que interactúan con este sistema es de uno, siendo el usuario que lo utiliza el único actor.

Casos de uso

CU-01	Jugar
Descripción	Jugar una partida.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1
Precondición	Tener el videojuego iniciado.
Acciones	<ol style="list-style-type: none"> 1. Si no lo ha realizado previamente, el usuario ejecuta la aplicación. 2. El usuario pulsa en el botón de "Jugar" en el menú principal.
Postcondición	Se muestra la pantalla de selección de modo de juego.
Excepciones	-
Importancia	Alta.

Tabla B.1: CU-01 Jugar

CU-02	Seleccionar modo
Descripción	Escoger entre los diferentes modos de juego disponibles.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.1
Precondición	Haber seleccionado la opción de "Jugar".
Acciones	<ol style="list-style-type: none"> 1. Pulsar en el modo de juego deseado.
Postcondición	Se muestra la pantalla de selección de personaje.
Excepciones	-
Importancia	Alta.

Tabla B.2: CU-02 Seleccionar modo

CU-03	Seleccionar personaje
Descripción	Escoger un personaje de entre todos los personajes disponibles para conducir con él.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.1.1, RF-1.2

Tabla B.3: CU-03 Seleccionar personaje

CU-03	Seleccionar personaje
Precondición	Haber escogido un modo de juego.
Acciones	1. Pulsar en un ícono de personaje.
Postcondición	Se muestra la pantalla de selección de circuito.
Excepciones	-
Importancia	Alta.

Tabla B.3: CU-03 Seleccionar personaje

CU-04	Seleccionar circuito
Descripción	Seleccionar un circuito de entre todos los disponibles para recorrerlo.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.1.1, RF 1.2, RF-1.3
Precondición	Haber seleccionado un personaje.
Acciones	1. Pulsar en un ícono de circuito.
Postcondición	Empieza la partida con las opciones previas seleccionadas.
Excepciones	-
Importancia	Alta.

Tabla B.4: CU-04 Seleccionar circuito

CU-05	Conducir vehículo
Descripción	Conducir el vehículo escogido por el usuario.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.4, RF-1.4.1, RF-1.4.2, RF-1.4.3
Precondición	Haber comenzado una partida.
Acciones	1. El usuario pulsa cualquiera de las acciones asociadas a la conducción del vehículo.
Postcondición	El vehículo debe moverse en base a la acción realizada.
Excepciones	-
Importancia	Alta.

Tabla B.5: CU-05 Conducir vehículo

CU-06	Acelerar vehículo
Descripción	Acelerar el vehículo durante el tiempo que desea el usuario.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.4, RF-1.4.1, RF-1.4.3
Precondición	Haber comenzado una partida.
Acciones	<ol style="list-style-type: none"> 1. El usuario pulsa la tecla "W." o la tecla de "flecha arriba" de manera intermitente o constante. 2. De manera opcional, el usuario pulsa, combinando con la tecla de aceleración, las teclas de giro ("A"/"flecha izquierda" o "D"/"flecha derecha").
Postcondición	El vehículo se desplaza hacia adelante o hacia adelante con giro según lo pulsado previamente.
Excepciones	-
Importancia	Muy alta.

Tabla B.6: CU-06 Acelerar vehículo

CU-07	Decelerar vehículo
Descripción	Decelerar el vehículo durante el tiempo que desea el usuario.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.4, RF-1.4.2, RF-1.4.3
Precondición	Haber comenzado una partida.
Acciones	<ol style="list-style-type: none"> 1. El usuario pulsa la tecla "S." o la tecla de "flecha abajo" de manera intermitente o constante. 2. De manera opcional, el usuario pulsa, combinando con la tecla de deceleración, las teclas de giro ("A"/"flecha izquierda" o "D"/"flecha derecha").
Postcondición	El vehículo se desplaza hacia atrás o hacia atrás con giro según lo pulsado previamente. Si el vehículo está en movimiento, el vehículo decelera.
Excepciones	-
Importancia	Muy alta.

Tabla B.7: CU-07 Decelerar vehículo

CU-08	Girar vehículo
Descripción	Girar vehículo sobre sí mismo si está parado o hacia el lado deseado en estado de movimiento.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.4, RF-1.4.1, RF-1.4.2, RF-1.4.3
Precondición	Haber comenzado una partida.
Acciones	1. El usuario pulsa la tecla de giro a la izquierda (“A”/“flecha izquierda”) o la tecla de giro a la derecha (“D”/“flecha derecha”) mientras el vehículo está en movimiento o mientras el vehículo está en estado de reposo.
Postcondición	El vehículo gira en la dirección indicada.
Excepciones	-
Importancia	Muy alta.

Tabla B.8: CU-08 Girar vehículo

CU-09	Recolocar vehículo
Descripción	Permite recolocar el vehículo manualmente en el último punto de control atravesado.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.4, RF-1.4.4
Precondición	Haber comenzado una partida.
Acciones	1. Pulsar la tecla “R” para recolocar el vehículo.
Postcondición	El vehículo se recoloca en el punto de control previamente atravesado.
Excepciones	-
Importancia	Alta.

Tabla B.9: CU-09 Recolocar vehículo

CU-10	Partida contrarreloj
Descripción	Jugar una partida contrarreloj con todos los elementos que conforman ese modo de juego.
Autor	Alejandro Goicoechea Román
Requisitos	RF-1, RF-1.1.1, RF-1.2, RF-1.3, RF-1.4

Tabla B.10: CU-10 Partida contrarreloj

CU-10	Partida contrarreloj
relacionados	
Precondición	Haber seleccionado este modo de juego.
Acciones	<ol style="list-style-type: none"> 1. Jugar la partida siguiendo las instrucciones previas vistas.
Postcondición	La partida se comporta como un juego de carreras contrarreloj.
Excepciones	-
Importancia	Alta.

Tabla B.10: CU-10 Partida contrarreloj

CU-11	Conteo de vueltas
Descripción	Contar las vueltas que realiza el usuario, así como el número total a realizar.
Autor	Alejandro Goicoechea Román
Requisitos	RF-1, RF-1.1.1, RF-1.1.1.1
relacionados	
Precondición	Haber comenzado una partida.
Acciones	<ol style="list-style-type: none"> 1. Se muestran las vueltas a realizar. 2. Se muestran las vueltas en la que se encuentra el usuario. 3. Por cada vuelta que realice el usuario, el contador actualizará el número de vueltas sumando una a éstas.
Postcondición	El contador se actualiza sumando nuevas vueltas.
Excepciones	-
Importancia	Alta.

Tabla B.11: CU-11 Conteo de vueltas

CU-12	Cronómetro de tiempo
Descripción	Contar el tiempo por vuelta realizada.
Autor	Alejandro Goicoechea Román
Requisitos	RF-1, RF-1.1.1, RF-1.1.1.2
relacionados	
Precondición	Haber comenzado una partida.
Acciones	<ol style="list-style-type: none"> 1. El cronómetro cuenta el tiempo por vuelta

Tabla B.12: CU-12 Cronómetro de tiempo

CU-12 Cronómetro de tiempo	
	realizada.
	2. Por cada vuelta realizada, se restaura a 0.
Postcondición	Muestra el tiempo correctamente.
Excepciones	-
Importancia	Alta.

Tabla B.12: CU-12 Cronómetro de tiempo

CU-13 Indicador de mejor tiempo	
Descripción	Indicar el mejor tiempo de vuelta realizada.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.1.1, RF-1.1.1.3
Precondición	Haber comenzado una partida.
Acciones	1. Por cada vuelta completada, si el tiempo es mejor al previo indicado, el contador se actualiza.
Postcondición	Muestra el mejor tiempo realizado.
Excepciones	-
Importancia	Alta.

Tabla B.13: CU-13 Indicador de mejor tiempo

CU-14 Instrucciones	
Descripción	Visualizar las instrucciones de juego.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-2
Precondición	Encontrarse en el menú principal.
Acciones	1. Pulsar en el botón de instrucciones.
Postcondición	Se visualizan las instrucciones del juego.
Excepciones	-
Importancia	Alta.

Tabla B.14: CU-14 Instrucciones

CU-15	Créditos
Descripción	Visualizar los créditos del juego con los participantes involucrados en el desarrollo del mismo.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-3
Precondición	Encontrarse en el menú principal
Acciones	1. Pulsar el botón de “Créditos”.
Postcondición	Se visualizan los créditos del juego.
Excepciones	-
Importancia	Media.

Tabla B.15: CU-15 Créditos

CU-16	Salir
Descripción	Salir del juego cerrando la aplicación y devolviendo al usuario al sistema operativo.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-4
Precondición	Encontrarse en el menú principal.
Acciones	1. Pulsar en el botón de “Salir”.
Postcondición	La aplicación se encuentra cerrada.
Excepciones	-
Importancia	Muy alta

Tabla B.16: CU-16 Salir

CU-17	Salir de la partida
Descripción	Salir del juego
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.1.1, RF-4,
Precondición	Haber comenzado una partida.
Acciones	1. Pulsar la tecla de “Escape” para abrir el menú de pausa.

Tabla B.17: CU-17 Salir de la partida

CU-17 Salir de la partida	
Postcondición	2. Pulsar en el botón de “Salir del juego”.
Requisitos relacionados	La partida finaliza, devolviendo al usuario al menú.
Excepciones	-
Importancia	Alta.

Tabla B.17: CU-17 Salir de la partida

CU-18 Volver a la pantalla anterior	
Descripción	Volver a la pantalla previa a aquella en la que se sitúa el usuario.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.1, RF-1.2, RF-1.3, RF-2, RF-3, RF-4, RF-5
Precondición	El usuario se encuentra en una pantalla de menú.
Acciones	1. Pulsar en el botón correspondiente a la vuelta a la pantalla anterior, con el respectivo nombre que le toque.
Postcondición	Se traslada a la pantalla previa.
Excepciones	-
Importancia	Media.

Tabla B.18: CU-18 Volver a la pantalla anterior

CU-18 Pausar partida	
Descripción	Pausar la partida comenzada, con opción a continuarla después.
Autor	Alejandro Goicoechea Román
Requisitos relacionados	RF-1, RF-1.1, RF-6
Precondición	El usuario se encuentra en una partida comenzada.
Acciones	1. Pulsar la tecla de “Escape”.
Postcondición	La partida se pausa.
Excepciones	-
Importancia	Media.

Tabla B.19: CU-18 Pausar partida

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se recogen las características de diseño de *software* que implementan los requisitos previamente explicados.

C.2. Diseño de datos

En “Fastastic Roads” no hay ningún tipo de base de datos o elementos que se le asemejen, al no ser necesario para los requisitos de este proyecto. De igual manera, cuenta con diferentes entidades, especialmente controladores, que conforman el buen funcionamiento del juego como uno de carreras contrarreloj.

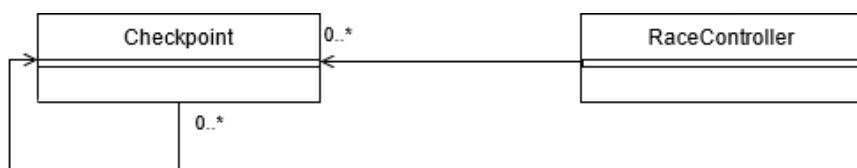


Figura C.1: Diagrama de clases de “Race Controller” y “Checkpoint”.

C.3. Diseño procedimental

La ejecución de la aplicación es sencilla, puesto que la aplicación se ejecuta bajo un único núcleo exportado desde Unity.

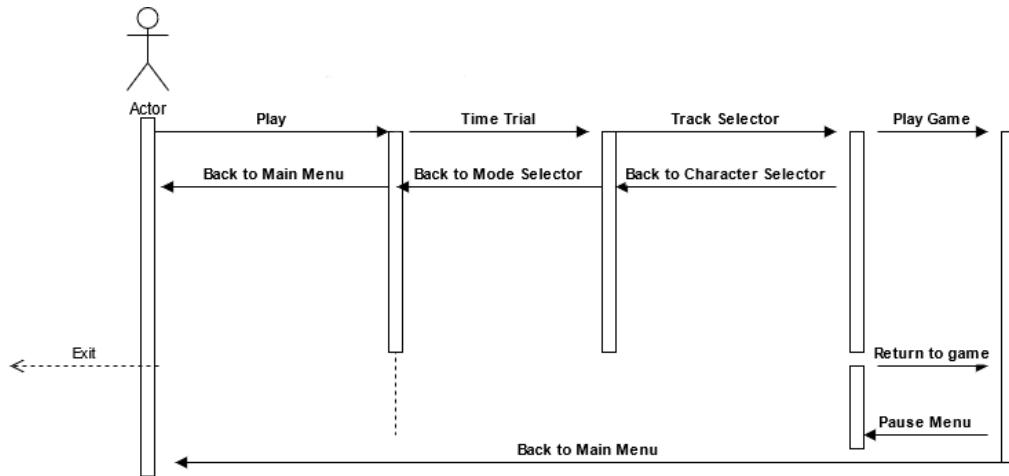


Figura C.2: Diagrama de secuencias de “Fastastic Roads”.

Entre los principales controladores esenciales se encuentra el controlador de carrera. Con “RaceController” se ha mantenido una relación directa con los *checkpoints* para el control de los mismos y su correcto funcionamiento en el circuito. Los *checkpoints* actúan a su vez tanto como puntos de control como línea de meta/salida. Éstos dependen entre sí para el debido transcurso de la carrera, permitiendo al vehículo bifurcaciones por distintos caminos donde pudiese haber puntos de control diferentes sin que esto afectara a la imposibilidad de finalizar una partida.

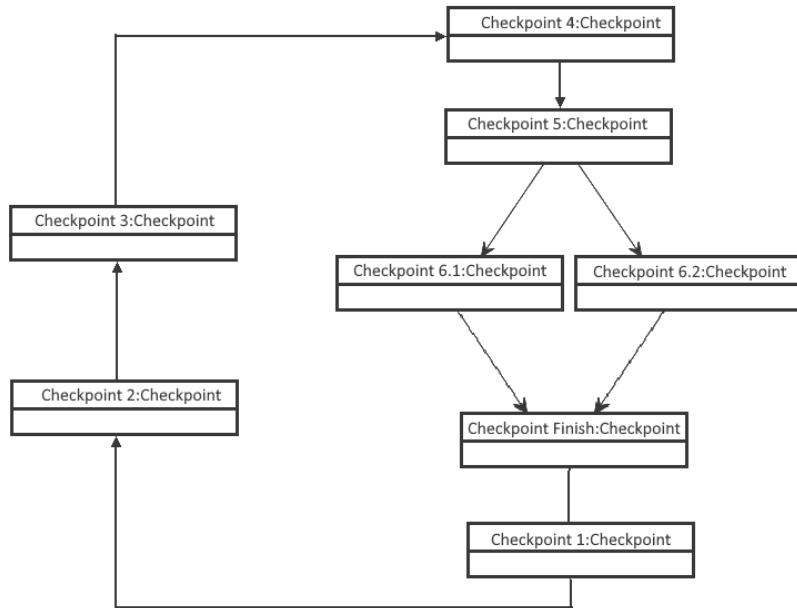


Figura C.3: Diagrama de objetos del grafo de Checkpoints.

C.4. Diseño arquitectónico

En “Fastastic Roads” se han aplicado algunos patrones de diseño a la hora de desarrollar el proyecto. Unity permite la implementación de patrones estandarizados, como el “Modelo-Vista-Controlador”. A continuación, se procede a explicar los patrones implementados.

Patrón estrategia

Mediante la implementación de este patrón de comportamiento, se permite al algoritmo variar independientemente de los clientes que lo utilizan. Aplicado al proyecto, el control en el videojuego se abstrae, materializándose en distintas estrategias (para el control local de un jugador o multijugador, control en red, etc.) para poder incluir nuevas funcionalidades sin tener que modificar el código independientemente de quién los utiliza [2].

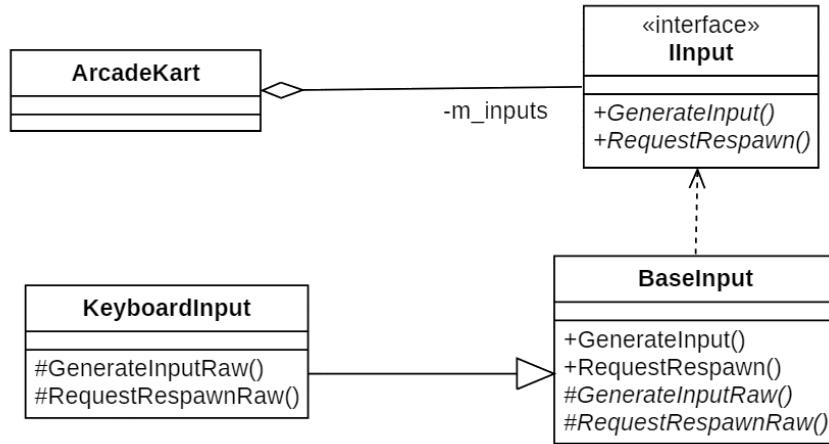


Figura C.4: Estructura del patrón de comportamiento estrategia.

Patrón método plantilla

En el patrón de comportamiento método plantilla se define el esqueleto de un algoritmo en una operación, dejando algunos de los pasos para las subclases, para que éstas redefinan ciertos pasos del algoritmo sin cambiar su estructura general [3].

En este proyecto, en el nivel abstracto de “BaseInput” se implementaban controles de bloqueo de las entradas comunes para todos los tipos de entradas y se delegaba a las subclases la captura de las entradas específicas.

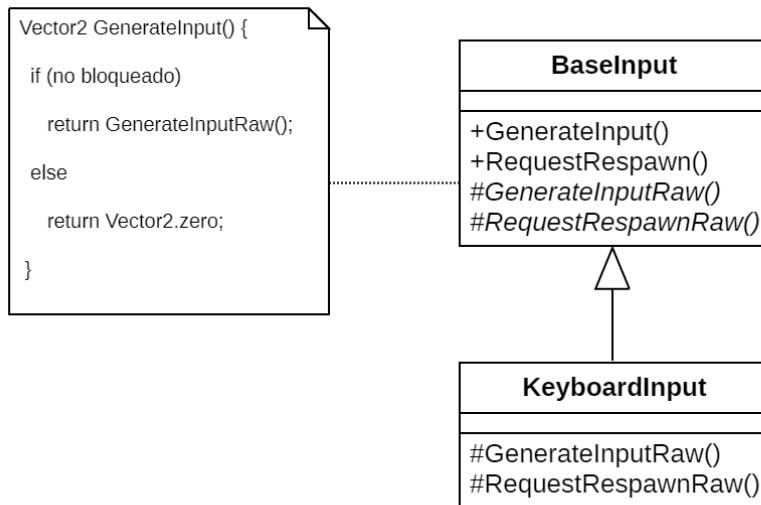


Figura C.5: Estructura del patrón creacional método plantilla.

C.5. Diseño de interfaces

Por último, respecto al diseño de interfaz se decidió por elementos simples.

Un buen diseño en una aplicación donde los componentes gráficos son notables y muy importantes de cara a la buena recepción del usuario como son los videojuegos, es muy importante. Sin embargo, en el desarrollo de un videojuego la interfaz, así como otros tantos elementos y funcionalidades del mismo, no se perfecciona dejándolo visualmente llamativo hasta que la aplicación está bastante avanzada, e incluso cercana a terminar.

No obstante, eso no implica que se tenga que descuidar, pues ha de ser accesible al usuario tanto en el juego final como en las fases de desarrollo. Por ello, se decidió la colocación de botones grandes en los distintos menús, fáciles de entender y eficientes con su función asignada.

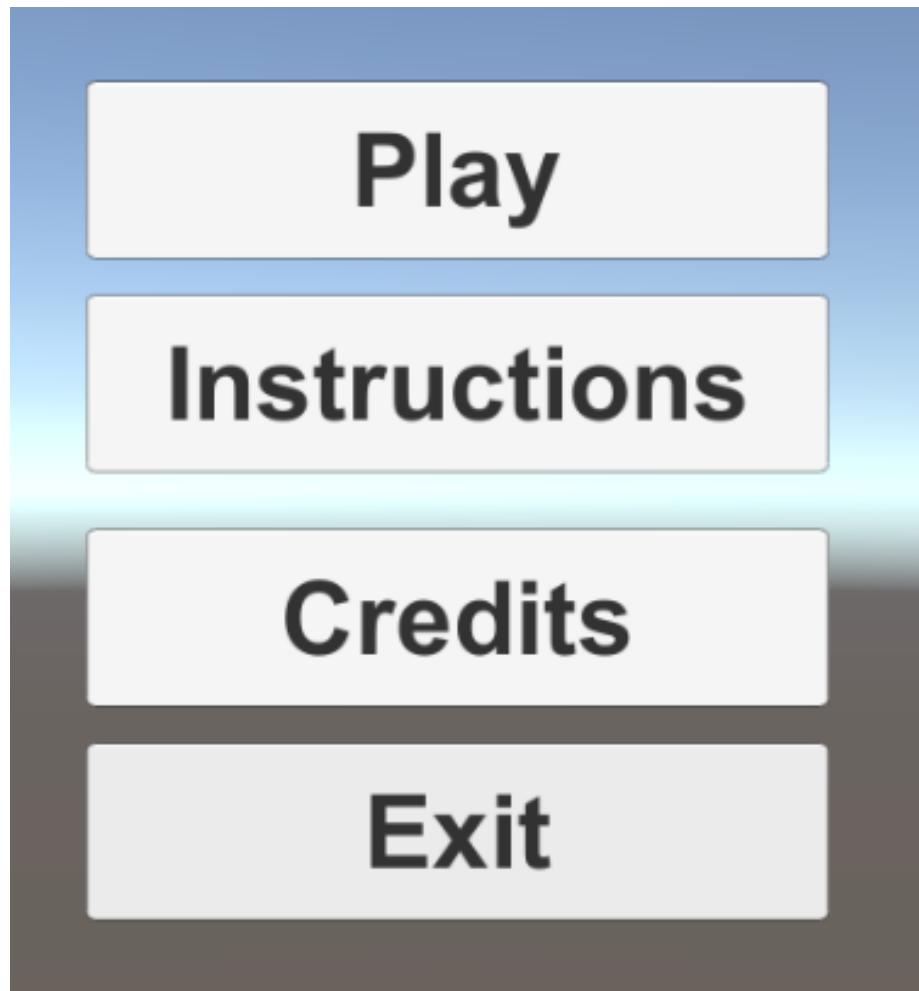


Figura C.6: A pesar de la carencia de colores y un diseño atractivo, un usuario medio es capaz de entenderlo fácilmente.

A medida que el proyecto vaya creciendo y se vayan creando nuevas funcionalidades, los menús tendrán tipografías más agradables a la vista y los botones serán más llamativos que los actualmente asignados.

Respecto al juego en sí, es importante tener una buena implementación de las distintas funciones para poder disfrutar de él, pero con un mal diseño puede acabar aburriendo o cansando al usuario. Es por ello que este aspecto es más importante tenerlo trabajado desde un principio, por lo que el juego tiene escenarios altamente detallados y llamativos, así como unos diseños de vehículos que invitan al usuario a disfrutar de una conducción divertida con ellos.



Figura C.7: Los escenarios se han trabajado desde un principio, permitiendo mejoras a medida que se avanza con el proyecto.

A ello se le añade un buen diseño de logotipo para la identificación del videojuego. Este logotipo tuvo una fase previa, y ahora mismo es el diseño final disponible con el cual se puede distinguir a “Fastastic Roads”.

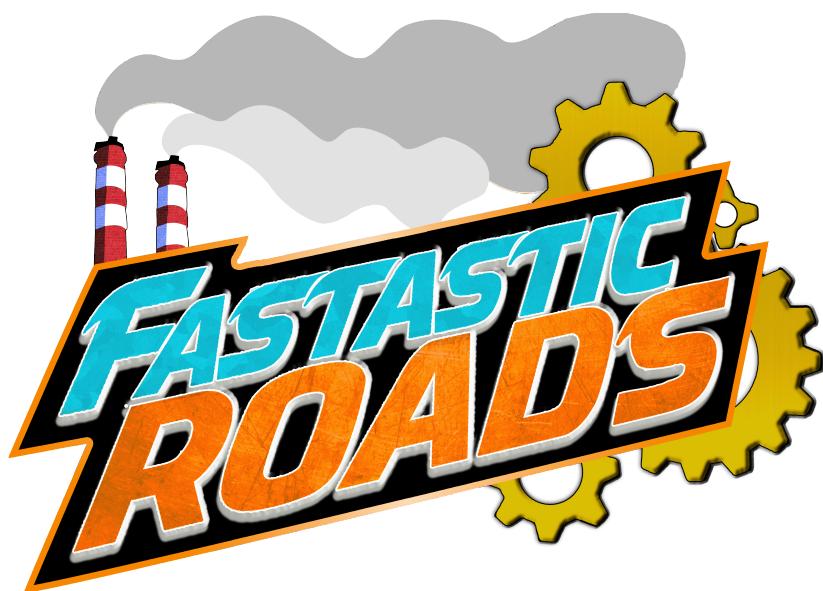


Figura C.8: El logotipo de “Fastastic Roads” apela a la estética industrial del videojuego.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apéndice se procede a explicar la organización por directorios del proyecto, así como el manual de programador y el resto de la información al respecto de la programación que pueda ser de utilidad.

D.2. Estructura de directorios

La estructura de directorios ubicada en el código del proyecto de GitHub¹ es la siguiente:

- /docs: contiene toda la documentación del proyecto, incluido este fichero de anexos.
- /docs/API: contiene toda la documentación del API construida con Doxygen.
- /game: directorio raíz del juego en Unity.
- /game/Assets: guarda todos los ficheros de modelos, animaciones, *prefabs*, escenas y *scripts*. Se hará hincapié en esta última carpeta, pues es la que contiene los ficheros que se han programado para el funcionamiento del videojuego con modo a contrarreloj.

¹Repositorio disponible en: <https://github.com/Kencho/fastastic-roads>

- /game/Assets/Scripts: contiene todos los *scripts* programados y utilizados para el funcionamiento del proyecto como videojuego de carreras con modo a contrarreloj.
- /game/Packages: contiene los ficheros correspondientes a la información de paquetes dentro de Unity.
- /game/ProjectSettings: contiene todos los ficheros de configuración de proyecto, necesarios para que Unity pueda cargar el proyecto al añadirlo.
- /game/UserSettings: contiene los ficheros de configuración de usuario.
- /builds: contiene todos los ficheros de datos del videojuego con sus respectivos ejecutables para Windows y Linux (solo disponibles en el material digital adjunto al proyecto).

D.3. Manual del programador

En este apartado, se explica detalladamente todo aquello necesario para cargar el proyecto correctamente, compilarlo y ejecutarlo.

Lo esencial necesario para poder cargar el proyecto, analizarlo y ejecutarlo es tener Unity instalado. Para ello, la versión utilizada tiene que ser igual o superior a la del proyecto. En este caso, se ha hecho uso de la versión 2020.3.1f1.²

No es necesaria la creación de una cuenta de Unity, pero es recomendable de cara a futuro si se va a usar más veces.

Una vez se haya descargado, se procede a instalar. Acto seguido, ha de descargarse el proyecto del repositorio de GitHub.

En Unity, para la gestión de proyectos, se tiene un “Unity Hub”, en el cual se pueden gestionar las versiones de Unity instaladas, así como la descarga de proyectos de aprendizaje y la creación e importación de nuevos proyectos. En este caso, lo que interesa es añadir uno ya existente, por lo que se pulsa en “Add”, se selecciona la carpeta “game” y se le da a “Aceptar”. En ese momento se tendrá el proyecto ya añadido.

Para la programación, lo ideal es tener instalado Visual Studio en el ordenador, siendo la versión “Community” la recomendada. Para asegurarse de que *IntelliSense* en Visual Studio funcione correctamente con el motor de Unity, es necesario ejecutar un proyecto cualquiera y pinchar, en la barra

²Enlace de descarga de la versión 2020.3.1f1 de Unity: https://unity3d.com/es/get-unity/download?thank-you=update&download_nid=64582&os=Win

de arriba, en “Edit”, a continuación en “Preferences” y, dentro de “External Tools”, en “External Script Editor” escoger “Visual Studio Community” (o el editor que se tenga).

D.4. Compilación, instalación y ejecución del proyecto

Como se ha mencionado en el apartado anterior, una vez se tiene el proyecto ya importado, ya se puede ejecutar.

Al abrirlo, frente al usuario aparece la pantalla principal de Unity. En ella se verá cargada la escena última abierta. Se pueden abrir las diferentes escenas pinchando, dentro de la pestaña de “Project”, en la carpeta “*Scenes*”, ubicada en la parte inferior izquierda de la pantalla. La nomenclatura de las escenas ha sido escogida de manera que sea lo más orientativa de cara al proyecto.

La distribución de los elementos en pantalla puede ser cambiada como prefiera el usuario, arrastrando las pestañas por las diferentes posiciones de la pantalla para recolocarlos y pudiendo aumentar o disminuir su tamaño . En caso de querer recuperar la posición original de estos bloques, se puede restaurar la vista pinchando en “*layout*” y, a continuación, en “*Default*” (o vista por defecto).

Los diferentes elementos que componen las escenas se encuentran en la “Jerarquía de objetos”, ubicada a la izquierda de la pantalla. Cada vez que se pincha en un objeto, en el “*Inspector*” ubicado en la derecha de la pantalla aparecerá los distintos componentes del mismo. Todos tendrán un componente “*Transform*” no eliminable, pues es el que indica la posición en la escena mediante un sistema de coordenadas. Esta posición, con las mismas coordenadas, no será igual para un objeto que depende de la escena que para un objeto que depende de otro objeto. Esta dependencia se puede comprobar cuando un objeto se encuentra “dentro” de otro objeto (los objetos aparecerán con flechas a su izquierda, donde desplegará otra jerarquía local propia de objetos).

Entre los distintos componentes se encuentran los *scripts*, los cuales aparecen con un nombre acompañado de dos paréntesis en los que pone ese término (por ej., “Prueba (Script)”). Estos *scripts* pueden abrirse con el editor de código pulsando en los tres puntos ubicados a la derecha del nombre del *script* y, a continuación, en “Edit Script”. También pueden abrirse yendo a la carpeta en la que se ubican, en este caso la carpeta “*Scripts*”, y haciendo

doble *click* en ellos. Al pulsar, se abrirá la pantalla del editor de código y se podrá proceder a realizar las modificaciones pertinentes. En caso de querer crear un *script*, se puede realizar pulsando en “Add Component” en el inspector de un objeto y escribiendo un nombre inexistente del *script* que queremos crear para que se genere un nuevo archivo. De la misma manera, se puede pulsar con el botón derecho dentro de la carpeta y pinchando en “Create”, luego en “C# Script”.

Entendiendo el sistema básico de *scripts*, queda comprender el funcionamiento de las escenas. Para poder ejecutar una escena, ha de pulsarse en el botón de “Play”, ubicado en la parte superior de la pantalla. Este botón tiene la forma clásica de triángulo para indicar la reproducción, y va acompañado de otro botón de “Pause” con la forma clásica de dos rectángulos paralelos.

Al darle a “Play”, la escena se ejecuta. El usuario puede interactuar con la escena en base a los elementos implementados. En el caso de la escena del circuito, se puede jugar directamente con el teclado la partida. Las escenas están conectadas entre sí, significando esto la posibilidad de poder pausar el juego, terminar la partida y volver al menú, etc. Hay posibilidad de ver, además, la escena desde la cámara ubicada en la pestaña “Scene”, pudiendo moverse pinchando con el botón derecho del ratón en ella para mover la cámara y desplazándose, a la vez, con las teclas “W”, “A”, “S” y “D”. A la vez, se pueden analizar los distintos elementos que hay activos en la escena, pudiendo ser seleccionados desde la jerarquía de objetos y modificados sus valores, aunque estos cambios se revierten una vez se pausa la escena.

Para pausar la escena, se ha de pulsar en el botón de “Play” de nuevo.

En caso de que haya errores de compilación en alguna de las clases de las que dependa alguno de los componentes del juego, la escena no arrancará hasta que sean solventados, indicando un mensaje por pantalla.

Para exportar el juego, ha de pulsarse, en la barra superior, en “File” y, acto seguido, en “Build Settings...”. Aquí se añaden las escenas que deseemos en el juego para luego interconectarlas mediante *scripts*. En este caso ya se encuentra realizada esa configuración, por lo que solo ha de escogerse la plataforma deseada a exportar y pulsar en “Build”, escogiendo una carpeta donde guardarlo, o “Build And Run” para exportar y ejecutar directamente. Una vez se ha exportado, obtenemos el juego adjuntado con este proyecto.

D.5. Pruebas del sistema

En Unity hay posibilidad de realizar pruebas automáticas. El “Unity Test Framework” funciona bajo la API “Test Runner” y permite ejecutar pruebas de forma programática a través de cualquier *script* suyo [4].

No obstante, en este caso no se hace uso de ese tipo de pruebas, ya que requiere de un estudio mayor y para el flujo de trabajo llevado no es el adecuado. Es por ello que se hace uso de pruebas de aceptación basadas en los casos de uso.

La forma de trabajar con estas pruebas es sencilla. Se crean las tareas basadas en los requisitos del proyecto. Cuando una tarea está tentativamente completada, se establece en la fase de pruebas de su ciclo de vida, donde se realizan las pruebas de aceptación correspondientes. Una vez superadas estas pruebas, la tarea se da por terminada. En caso contrario, si no las superan, la tarea continúa su desarrollo devolviéndola a su fase anterior en el ciclo de vida. En el [Anexo B: Especificación de Requisitos](#) se encuentra cómo se realizan estas pruebas.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado se recogen los requisitos mínimos para poder ejecutar la aplicación, así como los pasos a seguir para ejecutarla y un manual de uso de la misma.

E.2. Requisitos de usuarios

Los requisitos mínimos para ejecutar el videojuego, calculados en base a los mencionados por Unity en su página web, son los siguientes:

- Procesador: cualquiera con arquitectura x86/x64 con soporte para set de instrucciones SSE2
- Memoria RAM: 4 gbs
- Tarjeta gráfica: cualquiera compatible con, al menos, DirectX 10
- Sistema operativo: Windows 7/8/10, solo en versión de 64 bits
- Ratón
- Teclado

De igual manera, se ha probado en diferentes PCs y los mínimos recomendados obtenidos, en base a esos PCs mencionados, son los siguientes:

- Procesador: Intel i7-8750H/AMD FX 6350
- Memoria RAM: 8 GBs
- Tarjeta gráfica: Intel UHD Graphics 630 (media de entre 25 - 30 fps)

- Sistema operativo: Windows 7
- Espacio ocupado en SSD/HDD: 250 MBs aprox.

Tras haber probado con diferentes pantallas, la escalabilidad de la resolución con los elementos en pantalla no es automática, por lo que la resolución de pantalla recomendada para una correcta visualización es de 1920x1080.

E.3. Instalación

El videojuego se exporta directamente desde el mismo motor Unity como aplicación ejecutable. Gracias a ello no es necesaria la instalación del mismo, sino que se puede abrir haciendo doble *click* directamente en el ejecutable con el nombre de “Fastastic Roads”.

Es recomendable tener actualizada la tarjeta gráfica, así como los controladores correspondientes para que el videojuego pueda ser ejecutado.

E.4. Manual del usuario

El uso de la aplicación es muy sencillo. Se empezará ejecutando el juego, haciendo doble *click* en el nombre de “Fastastic Roads.exe”, correspondiente al ejecutable. Aparecerá el logo de Unity y, a continuación, aparecerá el menú principal.

Menús

Las pantallas por las que se mueve el usuario son menús. En estos menús habrá botones, en los cuales se puede pinchar con el botón izquierdo del ratón. Se procede a explicar las diferentes pantallas. Todos los menús tienen un botón para retroceder a la pantalla anterior en la esquina inferior izquierda.

Menú principal

Este menú es la pantalla principal del juego. Permite proceder a las distintas pantallas del juego, así como a salir de él.

- **Play:** permite trasladarse a la pantalla de selección de modo de juego.
- **Instructions:** muestra en pantalla las instrucciones de manejo de vehículo, para entender cómo jugar en una partida.

- **Credits:** aparecen las distintas personas involucradas en el proyecto y sus roles.
- **Exit:** permite salirse del videojuego, devolviendo al usuario a *Windows*.



Figura E.1: Menú principal de “Fastastic Roads”.

Instrucciones

Muestra las instrucciones de juego. El manejo de vehículo se puede realizar con las flechas del teclado, o bien con las teclas “W”, “A”, “S” y “D” siguiendo la misma posición que las flechas (flecha arriba - W, flecha izquierda - A, flecha abajo - S, flecha derecha - D). En caso de volcado de vehículo o cualquier otra situación que lo requiera, hay posibilidad de recolocar el vehículo en el último punto de control atravesado pulsando la tecla “R”.

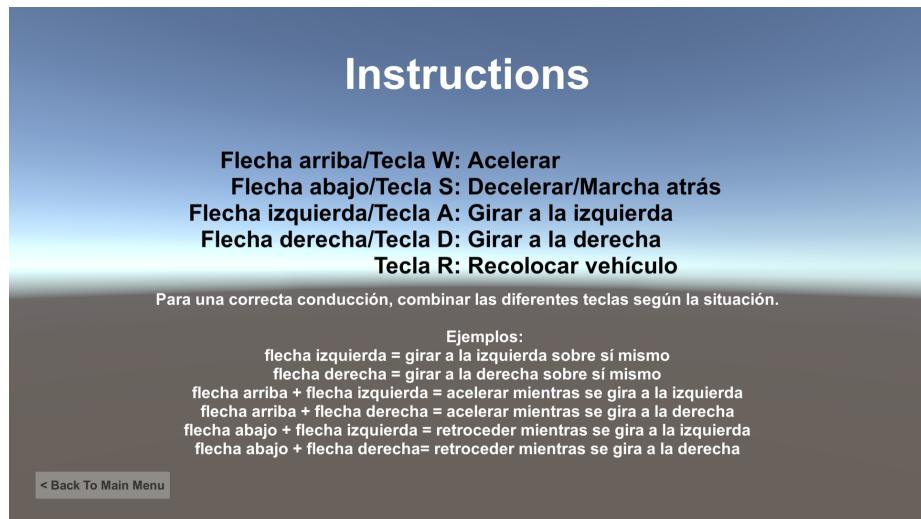


Figura E.2: Instrucciones para jugar.

Para jugar correctamente, se explican los siguientes puntos:

- El vehículo se mueve manteniendo pulsadas estas teclas. Si una tecla se pulsa pero no se mantiene, realizará un escaso movimiento. De igual manera, si una tecla se mantiene pulsada de manera constante, es probable que el usuario pierda el control del vehículo, por lo que tiene que hacer combinaciones de pulsaciones continuas y soltar las mismas teclas. Para jugadores menos experimentados, es aconsejable realizar toques cortos en las teclas para un mejor manejo en la partida.
- Si el vehículo está parado y se pulsa la tecla correspondiente al giro a la izquierda o al giro a la derecha, éste se moverá sobre sí mismo hacia el lado correspondiente de la tecla pulsada.
- Si el vehículo está en movimiento sin acelerar y se pulsa la tecla correspondiente al giro a la izquierda o al giro a la derecha, éste se moverá siguiendo la inercia del mismo hacia el lado correspondiente de la tecla pulsada.
- Si se pulsa la tecla de aceleración sin combinar con ninguna tecla de giro, el vehículo se moverá en línea recta.
- Si se pulsa la tecla de deceleración, sin estar previamente el vehículo en movimiento y sin combinar con ninguna de giro, el vehículo se moverá marcha atrás en línea recta.
- Si se pulsa la tecla de deceleración estando previamente el vehículo en movimiento y sin combinar con ninguna de giro, el vehículo decelerará hasta detenerse. Si no se suelta la tecla de deceleración una

vez el vehículo esté a punto de detenerse, iniciará la marcha atrás inmediatamente después.

- Si se pulsa la tecla de aceleración más una tecla de giro a la vez, el vehículo girará hacia la dirección deseada mientras avanza.
- Si se pulsa la tecla de deleración más una tecla de giro a la vez, el vehículo girará hacia la dirección deseada mientras avanza marcha atrás.

Para continuar hacia la partida, el usuario pulsará en “*Play*” (“Jugar”).

Selector de modo

En esta pantalla se escoge entre los diferentes modos disponibles. En el caso de este proyecto, solo está disponible para seleccionar el modo contrarreloj (*Time Trial*), con idea a futuro de integrar más modos, como el otro mostrado en rojo no seleccionable con el mensaje “*Coming soon*” (“Disponible próximamente”).



Figura E.3: Selector de modo de juego (solo disponible el modo a contrarreloj).

El usuario pulsará en “*Time Trial*” para continuar a la partida.

Selector de personaje

Permite escoger entre los distintos personajes disponibles. Aunque aparezcan varios nombres, en este proyecto, independientemente de dónde se pulse, solo se podrá jugar con el personaje de “Pilar Careaga”.

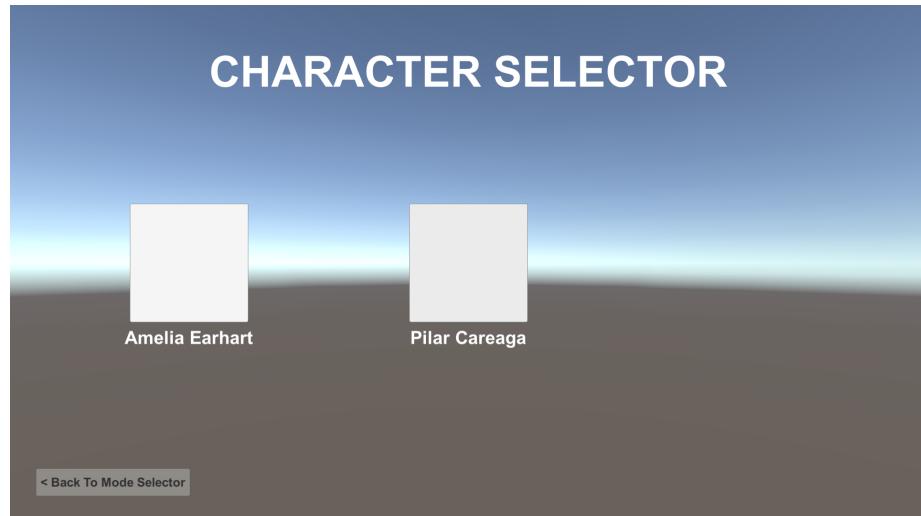


Figura E.4: Selector de personaje (solo disponible el personaje de Pilar Careaga).

Por ello, el usuario pulsará en cualquiera de los dos disponibles.

Selector de circuito

Antes de comenzar la partida, se escoge entre los distintos circuitos disponibles. De nuevo, solo hay un circuito disponible, por lo que independientemente de dónde se pulse llevará al mismo.



Figura E.5: Selector de circuito para jugar (solo uno disponible).

Por ello, el usuario escogerá cualquiera de los dos en pantalla.

Partida contrarreloj

Una vez comienza la partida, primero aparece una cuenta atrás en la cual el usuario no puede mover el vehículo hasta que haya terminado, donde aparecerá el mensaje “*Go!*” (equivalente a “¡Ya!”).



Figura E.6: Cuenta atrás antes de empezar la carrera.

Con la cuenta atrás finalizada, el usuario puede empezar a moverse. Siguiendo las instrucciones anteriormente mencionadas, el usuario tiene que seguir por el circuito avanzando por la carretera atravesando los diferentes puntos de control hasta llegar a meta.



Figura E.7: Los puntos de control se distinguen por su transparencia ligeramente translúcida.

Si el usuario se mueve en dirección contraria, los puntos de control que atraviese no serán válidos hasta que no llegue al que le corresponde atravesar inicialmente. Para que sea distinguible, hay diferentes señales con flechas indicativas en el escenario para orientar al jugador.



Figura E.8: Las flechas indican por dónde debe ir el jugador.

En la interfaz se puede observar un letrero de “*Lap*”, el cual indica el número de vuelta en el que se encuentra el usuario y el número total de vueltas de la carrera, así como otro letrero que indica el “*Lap time*” o tiempo de vuelta, y el “*Best lap time*” o mejor tiempo de vuelta.



Figura E.9: En algún punto del circuito puede haber bifurcaciones por las que circular.

El usuario puede encontrarse durante el recorrido de la partida diferentes caminos por los que ir. Podrá escoger libremente entre cualquiera de ellos, donde la diferencia de cada uno la marca la dificultad de recorrido.



Figura E.10: Al pasar de nuevo por meta, se actualiza el número de vueltas y el mejor tiempo.

Una vez atraviesa la meta después de haber pasado a través de todos los anteriores puntos de control, el contador de “Tiempo por vuelta” de restaura a cero, y si el tiempo ha sido mejor que el previo, se actualiza el contador de “Mejor tiempo por vuelta”.



Figura E.11: Hay posibilidad de pausar la partida dándole a la tecla de Escape.

En caso de que sea necesario, el videojuego se puede pausar pulsando la tecla de “Escape” en el teclado, permitiendo al usuario continuar el juego cuando se desee o, en caso contrario, abandonarlo, devolviendo al usuario de esta manera al menú principal.



Figura E.12: Al finalizar, se bloquea el manejo del vehículo y se salta a la pantalla de estadísticas.

Una vez se completen todas las vueltas, el vehículo bloquea su control impidiendo moverlo con el teclado, y devolviendo al usuario a la pantalla de estadísticas.

Créditos

Para finalizar, aquí se muestra una descripción del proyecto y los distintos participantes que han colaborado en él.



Figura E.13: En la pantalla de créditos aparecen los participantes en el desarrollo del proyecto.

Bibliografía

- [1] Asturias Corporación Universitaria. Clasificación de los costes. URL: https://www.centro-virtual.com/recursos/biblioteca/pdf/analisis_costos/unidad1_pdf2.pdf, 2021. Páginas 7 - 11. Último acceso: 07/07/2021.
- [2] Jesús María Alonso Abad. Universidad de Burgos. Estrategia. diseño y mantenimiento de software. Material universitario, 2019. Página 3. Último acceso: 08/07/2021.
- [3] Jesús María Alonso Abad. Universidad de Burgos. Método plantilla. diseño y mantenimiento de software. Material universitario, 2019. Página 3. Último acceso: 08/07/2021.
- [4] Unity. Prueba el código de tu juego con unity test framework. URL: <https://unity.com/es/how-to/unity-test-framework-video-game-development>, 2021. Último acceso: 06/07/2021.