



ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

ZYCARS: JUEGO DE CONDUCCIÓN 2D

José Jesús Marente Florín

30 de julio de 2011



ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

ZYCARS: JUEGO DE CONDUCCIÓN 2D

- Departamento: Lenguajes y sistemas informáticos
- Director del proyecto: Manuel Palomo Duarte y Juan Manuel Dodero Beardó
- Autor del proyecto: José Jesús Marente Florín

Cádiz, 30 de julio de 2011

Fdo: José Jesús Marente Florín

Agradecimientos

Licencia

Este documento ha sido liberado bajo Licencia GFDL 1.3 (GNU Free Documentation License). Se incluyen los términos de la licencia en inglés al final del mismo.

Copyright (c) 2011 José Jesús Marente FLorín.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Notación y formato

Cuando nos refiramos a un programa o biblioteca en concreto, utilizaremos la notación:

Python.

Cuando nos refiramos a un fragmento de código, usaremos la notación:

Código

Cuando nos refiramos a algún comando introducido en la terminal, usaremos la notación:

```
sudo apt-get install
```


Índice general

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	1
1.3. Estructura del documento	1
2. Descripción general del proyecto	3
2.1. Descripción	3
2.2. Características del videojuego	3
2.2.1. Modos de juego	3
2.2.2. Elementos de juego	4
3. Planificación	7
3.1. Fase inicial	7
3.2. Fase de análisis	7
3.3. Fase Aprendizaje	7
3.4. Fase de desarrollo	8
3.5. Pruebas y correcciones	8
3.6. Diagrama de Gantt	8
4. Análisis	11
4.1. Toma de requisitos	11
4.1.1. Requisitos de interfaces externas	11
4.1.2. Requisitos funcionales	12
4.1.3. Requisitos de rendimiento	13
4.1.4. Restricciones de diseño	13
4.1.5. Requisitos del sistema software	14
4.2. Modelo de casos de uso	14
4.2.1. Diagrama de los casos de uso	14
4.2.2. Descripción de los casos de uso	15
4.3. Modelo conceptual de datos	20
4.3.1. Diagrama de clases conceptuales	20
4.4. Modelo de comportamiento del sistema	21
4.4.1. Diagramas de secuencias del sistema y contrato de las operaciones del sistema.	21
5. Diseño	29
5.1. Interfaz gráfica	29
5.1.1. Diagrama de interacción entre interfaces	29
5.2. Diagrama de clases de diseño	30
5.3. Diagramas de secuencia	32

6. Implementación	33
6.1. Carga desde ficheros	33
6.2. Formato y carga de circuitos	34
6.3. Colisiones	36
6.3.1. Colisión con el escenario	37
6.3.2. Colisiones entre objetos	38
6.4. Inteligencia artificial	38
6.4.1. Realización del recorrido. Algoritmo de búsqueda A*	39
6.4.2. Lanzamiento de items.	40
7. Pruebas y validaciones	41
8. Conclusiones	43
A. Herramientas utilizadas	45
A.1. Lenguaje de programación	45
A.2. Biblioteca gráfica	46
A.3. Analizador de código: Pylint	47
A.4. Sistema de control de versiones	47
A.5. Documentación del código	47
A.6. Redacción de la memoria.	48
A.7. Realización de diagramas: Dia	48
A.8. Programa de edición de escenarios: Tiled	48
B. Manual de instalación	51
B.1. Linux: Ubuntu. Desde código fuente.	51
B.2. Linux: Ubuntu. Desde paquete debian.	52
B.3. Windows.	52
C. Manual de usuario	53
C.1. Menú principal	53
C.2. Modos de juego	53
C.2.1. Carrera rápida	54
C.2.2. Campeonato	54
C.2.3. Contrarreloj	54
C.3. Menú de selección de personaje	54
C.4. Menú de selección de circuito	55
C.5. Menú de Opciones	55
C.5.1. Sonido	56
C.5.2. Pantalla	56
C.5.3. Controles	57
C.6. Items	58
D. Manual para añadir nuevos personajes	61
D.1. Imagenes necesarias	61
D.2. Añadir imagenes a los recursos del juego	63
D.3. Creación del fichero del personaje	63
D.4. Añadir al personaje para que sea seleccionable.	64
Bibliografia y referencias	65

GNU Free Documentation License	69
1. APPLICABILITY AND DEFINITIONS	69
2. VERBATIM COPYING	70
3. COPYING IN QUANTITY	70
4. MODIFICATIONS	71
5. COMBINING DOCUMENTS	72
6. COLLECTIONS OF DOCUMENTS	73
7. AGGREGATION WITH INDEPENDENT WORKS	73
8. TRANSLATION	73
9. TERMINATION	73
10. FUTURE REVISIONS OF THIS LICENSE	74
11. RELICENSING	74
ADDENDUM: How to use this License for your documents	74

Indice de figuras

2.1. Descripción: Logo de Zycars	3
2.2. Descripción: Personaje de Zycars.	4
2.3. Descripción: Caja de item.	5
3.1. Planificación: Diagrama de Gantt 1/2.	9
3.2. Planificación: Diagrama de Gantt 2/2.	9
4.1. Análisis: Diagrama de casos de uso	14
4.2. Análisis: Diagrama de clases conceptuales	21
4.3. Análisis: Diagrama de secuencia Menú principal	21
4.4. Análisis: Diagrama de secuencia Elegir jugador	22
4.5. Análisis: Diagrama de secuencia Elegir circuito	23
4.6. Análisis: Diagrama de secuencia Elegir campeonato	23
4.7. Análisis: Diagrama de secuencia Jugar carrera	24
4.8. Análisis: Diagrama de secuencia Jugar campeonato	25
4.9. Análisis: Diagrama de secuencia opciones	26
4.10. Análisis: Diagrama de secuencia Salir	27
5.1. Diseño: Capturas de la interfaz del sistema	29
5.2. Diseño: Diagrama de interacción	30
5.3. Diseño: Diagrama de clases de diseño	31
6.1. Implementación: conjunto de tiles	35
6.2. Implementación: Mapa de colisiones	36
6.3. Implementación: Colisión con el escenario 1/2	37
6.4. Implementación: Colisión con el escenario 2/2	38
6.5. Implementación: Ejemplo del algoritmo A*	39
6.6. Implementación: Segmentos de la inteligencia artificial	40
A.1. Herramientas utilizadas: Logo de python	46
A.2. Herramientas utilizadas: Logo de pygame	46
A.3. Herramientas utilizadas: Logo de L ^A T _E X	48
A.4. Herramientas utilizadas: Logo de Tiled	48
A.5. Herramientas utilizadas: Captura del editor de mapas Tiled	49
C.1. Manual de usuario: Menú principal	53
C.2. Manual de usuario: Menú selección de personaje	54
C.3. Manual de usuario: Menú selección de circuito	55
C.4. Manual de usuario: Menú opciones - Audio	56
C.5. Manual de usuario: Menú opciones - Pantalla	57
C.6. Manual de usuario: Menú opciones - Controles	57

C.7. Manual de usuario: Bola de item.	58
C.8. Manual de usuario: Misil.	58
C.9. Manual de usuario: Tres misiles.	58
C.10. Manual de usuario: Bola.	59
C.11. Manual de usuario: Chicle.	59
C.12. Manual de usuario: Macha de aceite.	59
C.13. Manual de usuario: Trubo.	59
D.1. Manual para añadir personajes: Ejemplo de coche con dimensiones de 42 x 18 píxeles.	61
D.2. Manual para añadir personajes: Ejemplo de imagen de corredor.	62
D.3. Manual para añadir personajes: Ejemplo de avatar de corredor.	62

Indice de tablas

Capítulo 1

Introducción

1.1. Motivación

Mi interés por el mundo de los videojuegos, desde que tuve una Super Nintendo, y tras haber cursado en la carrera la asignatura optativa de "Diseño de videojuegos", donde aprendí mucho relacionado con el desarrollo de videojuegos, aumentaron mi interés por este mundo y además el desarrollo de ellos. Por lo que desde entonces consideraba realizar como proyecto fin de carrera un videojuego.

También he de añadir que tras conocer abiertamente el mundo del Software libre, gracias a la importancia que se le presta en la Universidad de Cádiz. Se decidió que el proyecto sería software libre bajo licencia GPL 3. Y así cualquier persona interesada en el desarrollo de videojuegos y en el software libre en general, pudiera usar los recursos del proyecto libremente.

1.2. Objetivos

Los objetivos principales del proyecto, es la realización de juego tanto para personas que dedican varias horas a la consecución de videojuegos, tanto para personas casuales, que dedican poco tiempo jugando.

Por lo que ser un juego de conducción el cual no esta compuesto por ninguna historia o trama argumental, facilita que se le pueda jugar pequeños intervalos de tiempo o, sin embargo, dedicarle varias horas al día.

Otro de los objetivos del proyecto, es poder hacerlo ampliable, de forma que cualquier persona mediante indicaciones y manuales pueda añadir tanto nuevo vehículos, como circuitos.

1.3. Estructura del documento

Este documento esta compuesto por las siguientes partes:

- **Introducción:** pequeña descripción del proyecto, así como los objetivos y estructura del documento.
- **Descripción general:** descripción más amplia sobre el proyecto, así como todas las características relevantes que tendrá.

- **Planificación:** exposición de la planificación del proyecto y las distintas etapas que esta compuesto el mismo.
- **Análisis:** fase de análisis del sistema, empleando la metodología seleccionada. Se definirán los requisitos funcionales del sistema, diagramas de caso de uso, diagramas de secuencia y contrato de las operaciones.
- **Diseño:** realización del diseño del sistema, diagramas de secuencia y clases aplicadas al diseño.
- **Implementación:** aspectos mas relevantes durante la implementación del proyecto. Y problemas que han aparecido durante el desarrollo de este.
- **Pruebas y validaciones:** pruebas realizada a la aplicación, con el fin de comprobar su correcto funcionamiento y cumplimiento de las expectativas.
- **Conclusiones:** conclusiones obtenidas tras el desarrollo de la aplicación.
- **Apendices:**
 - **Herramientas utilizadas:** explicación de todas las herramientas usadas a lo largo del desarrollo del proyecto.
 - **Manual de instalación:** manual para la correcta instalación del proyecto en el sistema.
 - **Manual de usuario:** manual de usuario para el correcto uso de la aplicación.
 - **Manual de para añadir nuevos personajes:** manual donde se explica los distintos pasos necesarios para añadir nuevos personajes al juego.
 - **Manual de para añadir nuevos circuitos:** manual donde se explica los distintos pasos necesarios para añadir nuevos circuitos al juego.
- **Bibliografía:** libros y referencias consultado durante el desarrollo del proyecto.
- **Licencia GPL 3:** texto completo sobre la licencia GPL 3, por la cual se rige el proyecto.

Capítulo 2

Descripción general del proyecto

2.1. Descripción

El proyecto consiste en un juego de carreras en dos dimensiones con vista cenital, en el que se podrá competir contra la inteligencia artificial. La idea es realizar un juego entretenido y dinámico, que estará compuesto por varios modos de juego.



Figura 2.1: Descripción: Logo de Zycars

2.2. Características del videojuego

El videojuego ofrece una alternativa libre, gratuita y original para jugar a un juego de conducción en dos dimensiones. Las posibilidades que ofrece son las siguientes:

2.2.1. Modos de juego

En Zycars tendremos distintos modos de juegos, en cada uno de ellos el objetivo que habrá que llevar a cabo será distinto. A continuación se describirán los distintos modos de juegos que tendrá el videjuego:

Carrera rápida

El juego en el modo de carrera rápida ofrece la posibilidad de enfrentarnos a 3 personajes controlados por la inteligencia artificial, a lo largo de un circuito que hayamos seleccionado previamente. El número de vueltas que se realicen durante la carrera estarán a elección del jugador y se podrá elegir el número de las mismas a la hora de seleccionar el circuito.

Campeonato

En este modo de juego podremos competir contra 3 personajes controlados por la inteligencia artificial a lo largo de un campeonato completo, el cual habremos elegido previamente.

El campeonato estará compuesto por cuatro circuitos y el número de vueltas a estos, también estarán a elección del jugador al igual que en el modo de juego explicado anteriormente.

Tras la conclusión de cada una de las carreras, los jugadores obtendrán una puntuación en función de la posición que haya obtenido. El jugador que mayor puntuación haya conseguido tras acabar los cuatro circuitos, se proclamará ganador del campeonato.

Contra reloj

En este último modo de juego y a diferencia de los dos anteriores, el jugador competirá solo sin la ningún oponente.

El objetivo en este modo de juego será la realización de los circuitos ofrecidos y mejorar los tiempos de estos, ya sean la vuelta más rápida del circuito o el tiempo general. El número de vueltas que deberemos dar al circuito será un total de tres, a diferencia de los modos anteriores, no tendremos la posibilidad de modificar el valor.

2.2.2. Elementos de juego

En esta sección se hará una pequeña descripción de los distintos elementos que encontraremos a lo largo del juego, ya sean manipulados por los jugadores, o encontrados a lo largo de los circuitos.

Personajes

Los elementos básicos del juego, habrá disponibles distintos personajes que tendrán asociado un vehículo característico a su personalidad y apariencia. Cada uno de ellos tendrá distintas características, cosa a tener en cuenta a la hora de hacer nuestra elección por uno de ellos, como la velocidad, la aceleración y el giro.

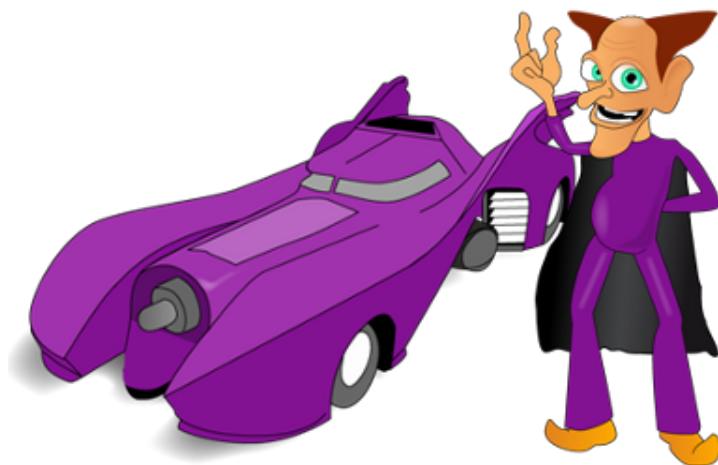


Figura 2.2: Descripción: Personaje de Zycars.

Cajas de items

A lo largo de los circuitos en los que estemos compitiendo contra la inteligencia artificial, podremos encontrar distintas cajas que al colisionar con ellas nos proporcionen aleatoriamente una habilidad o item que nos ayuden en la competición contra nuestros rivales.



Figura 2.3: Descripción: Caja de item.

Tipos de items

Los items que podremos obtener a partir de la caja de items, los podremos diferenciar principalmente en tres tipos:

Ataques a distancia Estos nos permitirán lanzar ataques de forma que podamos interceptar a los competidores que se encuentren lejos de nosotros.

Obstáculos Estos nos permitan dejar obstáculos en el recorrido, que reduzcan nuestra velocidad considerablemente o aquellos que al pasar por encima perdamos completamente el control de nuestro vehículo por unos instantes de tiempo.

Velocidad Esto nos darán la opción de aumentar nuestra velocidad durante un pequeño intervalo de tiempo.

Capítulo 3

Planificación

La planificación realizada para el desarrollo del proyecto, está dividida en varias partes:

3.1. Fase inicial

La primera fase consistió en plantear la idea del proyecto, con la ayuda del tutor. Tras varias propuestas y la deliveración sobre las mismas, se decidió realizar este proyecto.

También se pensó en que lenguaje se desarrollaría el proyecto, así como las principales bibliotecas que se usarían durante la realización del mismo.

3.2. Fase de análisis

Esta etapa está dividida principalmente en las dos partes siguiente:

- **Especificación de los requisitos:** estudio de los requisitos que deberá cumplir el juego.
- **Recurso necesarios:** recursos necesarios que deberemos usar durante el desarrollo del proyecto.

3.3. Fase Aprendizaje

Dado que el proyecto se realizaría con un lenguaje de programación del que no se tenían conocimientos, en este caso *Python*, así como de la biblioteca que usaríamos en el desarrollo, como es *Pygame*, esta fase se dividió en dos partes:

- **Aprendizaje de Python:** periodo empleado para el aprendizaje del lenguaje de programación *Python*, durante esta etapa se consultó varios libros sobre lenguaje, así como foros de internet y páginas web. Para un aprendizaje más ameno y llevadero, se realizaban problemas ya resuelto en otros lenguajes.
- **Familiarización con la biblioteca Pygame:** tras el periodo de aprendizaje del lenguaje, debía familiarizarme con la biblioteca principal que se usaría en el desarrollo del proyecto, como es *Pygame*. Durante su aprendizaje se realizaban pequeñas aplicaciones sencillas, para asentar bien los conocimientos.

3.4. Fase de desarrollo

Tras la consecución de las etapas anteriores, se comenzó el desarrollo del proyecto. Esta etapa del desarrollo es la mas extensa de todas, como es comprensible. Y la que más subetapas contiene, las principales son la siguientes:

- **Motor básico:** implementación de las necesidades básicas del proyecto, como control del teclado, carga de recursos, movimiento de los vehículos.
- **Carga de escenario:** carga de los circuitos que compondría el juego de forma que no fuera necesario tocar código para la ampliación del juego.
- **Creación de menús:** implementación de toda la interfaz de menús de la que estaría compuesto el juego, menú de opciones, selección de personaje, selección de circuito, etc.
- **Colisiones:** unos de los aspectos más básico y esenciales de cualquier juego, se debía implementar las colisiones con el escenario, así como con otros elementos del juego como pueden ser items u otros vehículos.
- **Items:** implementación del comportamiento y efecto que producirían cada uno de los items que están disponibles en el juego.
- **Inteligencia articial:** planteamiento y desarrollo de los vehículos que serían manejados por la inteligencia artificial, estos deberían de ser capaces de evitar obstáculos, realizar recorridos y lanzar items.
- **Modos de juego:** realización de los modos de juego que componen el proyecto, como serían carrera rápida, contrarreloj y campeonato,

3.5. Pruebas y correcciones

Una de las etapas más importantes, si no es la que más, del desarrollo de cualquier proyecto. Esta etapa se realizaría en paralelo a la de desarrollo, ya que conforme se implementan nuevas funcionalidades, cada una debía ser probada exhaustivamente en cualquiera de las posibles situaciones que pudiera suceder.

3.6. Diagrama de Gantt

A continuación se muestra la planificación anteriormente comentada, en su correspondiente diagrama de Gantt:

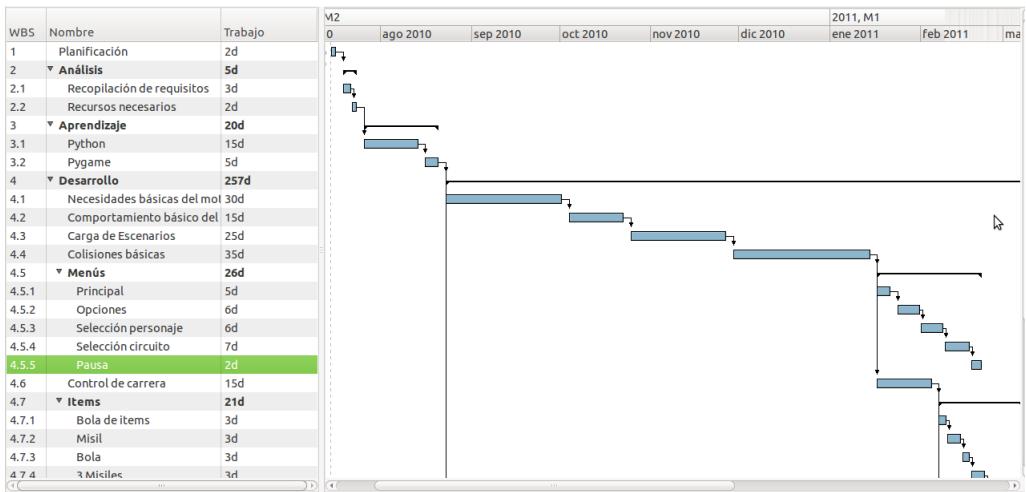


Figura 3.1: Planificación: Diagrama de Gantt 1/2.

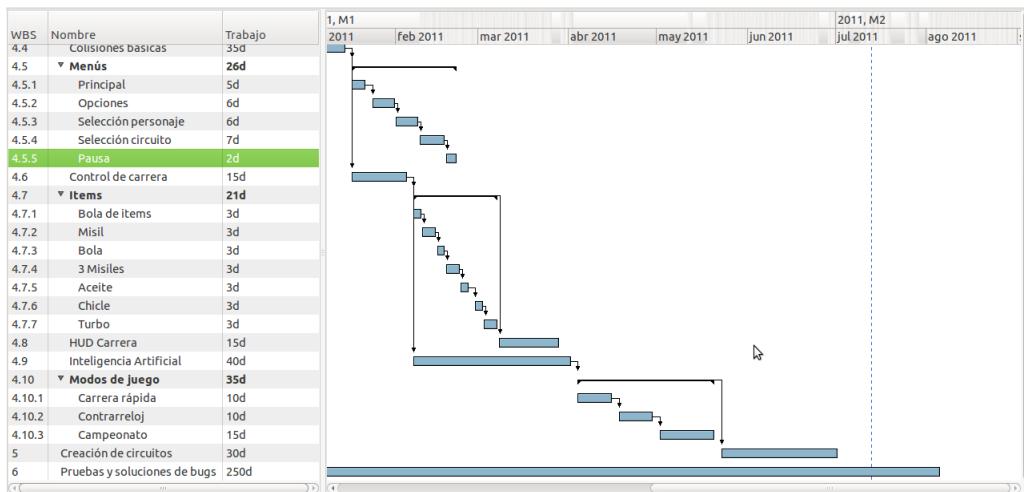


Figura 3.2: Planificación: Diagrama de Gantt 2/2.

Capítulo 4

Análisis

4.1. Toma de requisitos

Para la creación de cualquier producto software, es necesario establecer las distintas condiciones y necesidades que ha de satisfacer. Seguiremos un esquema que nos permita describir los requisitos de una forma metódica y racional.

4.1.1. Requisitos de interfaces externas

En este apartado se describirá los requisitos de conexión del software y el hardware, así como la interfaz de usuario.

La conexión entre el software y el hardware se encarga la librería *SDL*, mediante el wrapper *Pygame* para el lenguaje de programación *Python*. Por lo que al ser un sistema preestablecido, no será necesario realizar el diseño, ni el análisis, sólo haremos uso de él.

Así que pasamos a definir la interfaz entre el usuario y el videojuego. Todas las ventanas de la aplicación tendrán una resolución de 800x600 píxeles, siendo posible establecer el modo de pantalla completa¹. A continuación se distinguen las distintas ventanas que el usuario encontrará en el sistema:

Ventana de introducción En esta primera ventana se mostrará únicamente el logotipo del juego, situando al usuario en contexto para introducirlo en la ejecución de la aplicación.

Ventana de menú principal La ventana del menú principal muestra el menú de inicio de *Zycars*, así como todas las opciones generales del juego disponibles, que son las siguientes:

- Carrera Rápida
- Campeonato
- Contrarreloj
- Opciones
- Salir

En este menú y en los siguientes que se describan se usará el ratón para navegar por ellos y solo será necesario un click sobre la opción deseada para acceder a ella.

¹El modo de pantalla completa se podrá establecer a través del menú de opciones

Ventana de opciones Desde la ventana de opciones se podrán modificar las distintas características de la configuración del juego como el audio o controles. Esta ventana se podría dividir en tres partes diferencias que se indican a continuación:

- Opciones de audio: podremos modificar el volumen de efectos de sonido y música del juego. También estará la opción de silenciar cualquier tipo de sonido.
- Opciones de pantalla: a través de esta ventana podremos activar o desactivar el modo de pantalla completa.
- Opciones de control: en esta ventana podremos modificar los controles del juego. Tanto de dirección, lanzamiento de items y pausa del juego.

Ventana de selección de personaje Esta ventana será compartida por los tres modos de juego disponibles. En ella podremos elegir al personaje que desearemos controlar a lo largo de las carreras. De estos jugadores se nos mostrarán sus distintas habilidades.

Ventana de selección de circuito Ventana compartida por el modo carrera rápida y contrarreloj. En esta ventana deberemos elegir el circuito en el que deseamos competir. Se nos mostrará una imagen de cada circuito que seleccionemos.

Ventana de selección de campeonato Ventana muy similar a la descrita anteriormente. En ella se nos mostrarán todos los circuitos de cada campeonato disponible. Pero al contrario que la anterior en esta ventana elegiremos el campeonato del circuito seleccionado en el momento.

Ventana de juego Ventana principal de todo el juego. Mostrará la carrera actual que se esté disputando, así como los distintos marcadores aclaratorios sobre el estado de la carrera, como pueden ser posiciones de los jugadores, item actual y tiempos de carrera. Según la tecla indicada en los controles del menú de opciones (ESC o p) se podrá acceder al menú de pausa del juego.

Ventana de pausa Únicamente accesible desde la ventana de juego. Esta nos permitirá detener el juego en curso, siendo posible reanudar el juego, reiniciar el mismo o volver al menú principal.

Ventana de posiciones de carrera Ventana mostrada al terminar alguna de las carreras. En ella nos muestra el resultado de la última carrera disputada. Nos permite continuar al siguiente circuito, en el caso del modo campeonato, o seguir hacia el menú principal, en el modo carrera rápida. También se nos permite reiniciar la última carrera disputada.

Ventana de posiciones de campeonato Ventana mostrada en el modo campeonato tras la ventana de posiciones de carrera, en ella se nos muestra las posiciones de los competidores en el campeonato actual.

Ventana de tiempos de contrarreloj Ventana mostrada al completar algún circuito en el modo contrarreloj. En ella se nos muestran los distintos tiempos conseguidos a lo largo del circuito y se nos indicará si hemos batido algún record. Esta ventana nos permite continuar hacia el menú principal, así como reiniciar el circuito disputado.

4.1.2. Requisitos funcionales

Los requisitos funcionales que el sistema debe ofrecer son los siguientes:

- Salir de la aplicación desde cualquier ventana.
- Seleccionar los distintos modos de juego.

- Permitir al jugador competir contra la inteligencia artificial.
- Modificar la configuración (audio, pantalla y controles) del juego.
- Pausar el juego.
- Seleccionar uno de los jugadores propuestos.
- Seleccionar cualquiera de los circuitos disponibles.
- Seleccionar cualquiera de los campeonatos disponibles.
- Reiniciar cualquier carrera una vez terminada o en curso.
- Reiniciar cualquier campeonato una vez terminado o en curso.
- Lanzamientos de items durante cualquier carrera.

Los distintos tipos de jugadores son:

- **Humano:** es el controlado por una persona
- **Inteligencia artificial:** controlado por el ordenador.

Existen tres modos de juego:

- **Carrera rápida:** consiste en la realización de un único circuito, compitiendo contra la inteligencia artificial.
- **Campeonato:** el jugador competirá contra la inteligencia artificial a lo largo de 4 carreras, en las que obtendrá una puntuación según la posición obtenida en cada una de las carreras. El ganador será el que mejor puntuación haya conseguido al concluir el campeonato.
- **Contrarreloj:** en este modo de juego, el jugador competirá solo, con el fin de mejorar las marcas de tiempo de cada uno de los circuitos.

4.1.3. Requisitos de rendimiento

El rendimiento de la aplicación debe ser tal que permita un desempeño agradable de juego.

- Por lo que la respuesta a las acciones realizadas por el usuario deben ser respondidas lo mas rápido posible, sacrificando en el caso de que sea necesario el consumo de la memoria principal.
- La inteligencia artificial debe estar optimizada de forma que no se realentice la partida en el tiempo dedicado a los cálculos necesarios para tomar decisiones.

4.1.4. Restricciones de diseño

Como comenté en uno de los puntos del apartado anterior el tiempo de respuesta tiene que primar sobre el consumo de memoria principal o secundaria. Esta será la principal restricción de diseño que tendrá nuestra aplicación.

Los videojuegos están pensados como aplicación principal, de forma que no tenga que compartir recursos con otros procesos, por lo que se permitirá que consuma muchos recursos del sistema.

4.1.5. Resquisitos del sistema software

La aplicación deberá cumplir los siguiente requisitos del sistema:

- Deberá ser multiplataforma, al menos en los siguiente sistemas:
 - **Microsoft Windows**: realizando las pruebas sobre la versión Windows 7.
 - **GNU/Linux**: usando la distribución Ubuntu 10.10 como principal sistema para pruebas.
- El código con el que se desarrolle la aplicación no debe ser dependiente del sistema en el que se desarrolle.
- El código debe ser mantenible y facilmente ampliable para futuras versiones.

4.2. Modelo de casos de uso

Para describir los distintos comportamientos que tendrá el sistema, usaremos el lenguaje de modelado de sistemas *UML*; que representa los requisitos funcionales del sistema, centrando en que hace y no cómo lo hace.

4.2.1. Diagrama de los casos de uso

En primer lugar mostramos el modelo de casos de uso, que representa la funcionalidad completa de la aplicación. Se ha usado el siguiente esquema:

1. Identificar los usuarios del sistema y los roles que pueden tener.
2. Para cada rol, identificar las distintas formas de interactuar en el sistema. En el caso de *Zycars* existe un único rol de acceso a la aplicación, por lo que la especificación del usuario será única.
3. Creación de los casos de uso para todos los objetivos que queramos cumplir.
4. Estructurar dichos casos de uso.

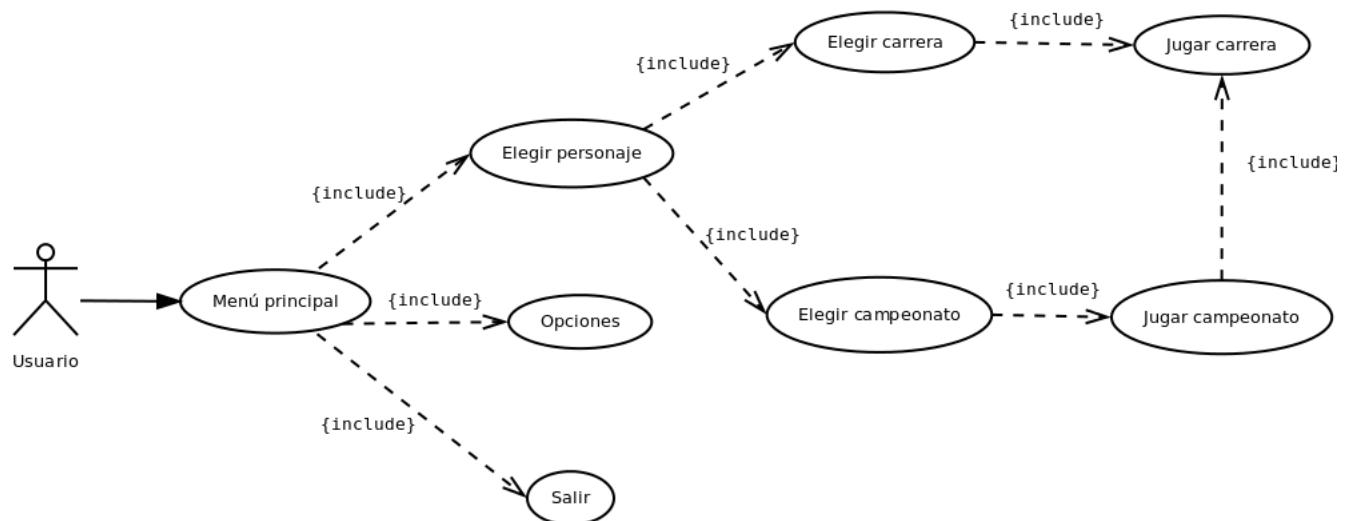


Figura 4.1: Análisis: Diagrama de casos de uso

4.2.2. Descripción de los casos de uso

A continuación pasamos a la descripción de cada uno de los casos de uso, para la cual usaremos una notación forma usando plantillas. El texto debe ser legible y comprendido por un usuario que no sea experto.

Caso de uso: Menú principal

Caso de uso Menú principal

Descripción Se muestra el menú principal de la aplicación, donde es posible elegir uno de los modos de juego disponibles o acceder al menú de opciones.

Actores Usuario

Precondiciones Ninguna

Postcondiciones Ninguna

Escenario principal

1. El sistema muestra el menú principal del juego en pantalla.
2. El usuario selecciona la opción **carrera rápida**.
3. El sistema inicia el modo de elección de personaje.

Extensiones — flujo alternativo

- ***a** El usuario cierra la ventana de la aplicación y sale de la aplicación
- 2a** El usuario selecciona la opción **campeonato**.
 1. El sistema inicia el modo de elección de personaje
- 2b** El usuario selecciona la opción **contrarreloj**.
 1. El sistema inicia el modo de elección de personaje
- 2c** El usuario selecciona la opción **opciones**.
 1. El sistema inicia las opciones
- 2d** El usuario selecciona la opción **salir**.
 1. El sistema sale de la aplicación.

Caso de uso: Elegir personaje

Caso de uso Elegir personaje

Descripción El usuario desea seleccionar el personaje con el que competirá en el juego, ya sea carrera rápida, campeonato o contrarreloj.

Actores Usuario

Precondiciones El usuario ha elegido previamente en el menú principal una de las opciones: **carrera rápida, campeonato o contrarreloj**

Postcondiciones Se almacenará en la configuración de juego el jugador seleccionado para su posterior uso.

Escenario principal

1. El usuario debe seleccionar al jugador que usará en el juego.
2. El sistema muestra el primer personaje junto a sus características.
3. El usuario está satisfecho con su selección y pulsa el botón aceptar.
4. El sistema almacena en la configuración el jugador seleccionado.
5. El sistema pasa a selección de circuito, si la precondición es **carrera rápida o contrarreloj**

Extensiones — flujo alternativo

- ***a** El usuario cierra la ventana de la aplicación y sale de la aplicación
- ***b** El usuario pulsa la opción cancelar.
 1. El sistema vuelve al menú principal
- 3a** El usuario no esta satisfecho con ese jugador y pulsa la flecha derecha.
 1. El sistema muestra el siguiente jugador.
- 3b** El usuario no esta satisfecho con ese jugador y pulsa la flecha izquierda.
 1. El sistema muestra el anterior jugador.

Caso de uso: Elegir circuito

Caso de uso Elegir circuito

Descripción El usuario desea seleccionar el circuito en el que desea competir.

Actores Usuario

Precondiciones El usuario ha elegido previamente en el menú principal una de las opciones: **carrera rápida o contrarreloj**

Postcondiciones El sistema almacena en la configuración el circuito seleccionado.

Escenario principal

1. El usuario desea seleccionar el circuito en el que competir.
2. El sistema muestra el primer circuito de todos los circuitos del campeonato actual.
3. El usuario esta satisfecho con su elección, y pulsa el botón aceptar.
4. El sistema almacena el circuito en la configuración del sistema.
5. El sistema inicia la carrera.

Extensiones — flujo alternativo

- ***a** El usuario cierra la ventana de la aplicación y sale de la aplicación
- ***b** El usuario pulsa la opción cancelar.
 1. El sistema vuelve a la selección de personaje.
- 3a** El usuario no está satisfecho con ese circuito y pulsa sobre otro circuito del mismo campeonato.
 1. El sistema muestra el nuevo circuito seleccionado.
- 3b** El usuario no está satisfecho con los circuitos de ese campeonato y seleccionado otro campeonato.
 1. El sistema muestra el primer circuito de los del campeonato seleccionado.

Caso de uso: Elegir campeonato

Caso de uso Elegir campeonato

Descripción El usuario desea seleccionar el campeonato que desea realizar.

Actores Usuario

Precondiciones El usuario ha elegido previamente en el menú principal una de las opciones: **campeonato**

Postcondiciones El sistema almacena en la configuración todos los circuitos del campeonato seleccionado.

Escenario principal

1. El usuario desea seleccionar el circuito en el que competir.
2. El sistema muestra el primer circuito de todos los circuitos del campeonato actual.
3. El usuario está satisfecho con su elección, y pulsa el botón aceptar.
4. El sistema almacena todos los circuitos del campeonato en la configuración del sistema.
5. El sistema inicia la carrera.

Extensiones — flujo alternativo

***a** El usuario cierra la ventana de la aplicación y sale de la aplicación

***b** El usuario pulsa la opción cancelar.

1. El sistema vuelve a la selección de personaje.

3a El usuario pulsa sobre otro circuito del mismo campeonato.

1. El sistema muestra el nuevo circuito seleccionado.

3b El usuario no está satisfecho con los circuitos de ese campeonato y selecciona otro campeonato.

1. El sistema muestra el primer circuito de los del campeonato seleccionado.

Caso de uso: Jugar carrera

Caso de uso Jugar carrera

Descripción El usuario juega una carrera

Actores Usuario

Precondiciones El usuario ha elegido previamente el personaje y el circuito donde jugar.

Postcondiciones El usuario completa una carrera

Escenario principal

1. El sistema carga el circuito.
2. El sistema carga el jugador.
3. El sistema carga la inteligencia artificial.
4. El sistema muestra la pantalla de juego.

5. El usuario y el sistema interactuan durante la carrera.
6. El usuario completa la carrera.
7. El sistema muestra las posiciones finales de la carrera.
8. El usuario pulsa continuar.
9. El sistema pasa a la siguiente pantalla dependiendo del modo de juego.

Extensiones — flujo alternativo

- *a** El usuario cierra la ventana de la aplicación y sale de la aplicación
- 3a** El sistema no carga la inteligencia artificial porque estamos en el modo campeonato.
- 8a** El usuario pulsa reiniciar.
 1. El sistema renicia la carrea.
 2. Volvemos al punto 1.

Caso de uso: Jugar campeonato

Caso de uso Jugar campeonato

Descripción El usuario juega un campeonato.

Actores Usuario

Precondiciones El usuario selección previamente la opcion **campeonato** en el menú principal.

Postcondiciones Se juega un campeonato.

Escenario principal

1. El usuario desea jugar un campenato
2. El sistema carga el circuito actual
3. El sistema y usuario interactuan en la carrera. Include Jugar carrera.
4. Una vez terminada la carrera el sistema muestras las posiciones del campeonato.
5. El usuario seleccion la opción continuar.
6. El sistema pasar al siguiente circuito.
7. Volvermos al punto 2.

Extensiones — flujo alternativo

- *a** El usuario cierra la ventana de la aplicación y sale de la aplicación
- 2a** Ya no hay más circuitos restantes.
 1. El sistema muestra la clasificación final de todos los jugadores del campeonato.
 2. El usuario pulsa la opción continuar.
 3. El sistema muestra la posición del jugador.

Caso de uso: Opciones

Caso de uso Opciones

Descripción El usuario desea modificar las opciones del juego.

Actores Usuario

Precondiciones El usuario seleccionó en el menú principal la opción **Opciones**.

Postcondiciones El usuario modifica las opciones del juego.

Escenario principal

1. El usuario desea modificar las opciones del juego.
2. El sistema muestra inicialmente las opciones de audio.
3. El usuario modifica las distintas opciones de audio.
4. El usuario esta conforme con los cambios realizados y pulsa sobre el botón aceptar.
5. El sistema almacena todos los cambios en la configuración.
6. El sistema vuelve al menú principal.

Extensiones — flujo alternativo

***a** El usuario cierra la ventana de la aplicación y sale de la aplicación.

***b** El usuario selecciona la opción cancelar.

1. El sistema vuelve al menú principal.

4a El usuario desea modificar las opciones de pantalla y pulsa sobre el botón de opciones de pantalla.

1. El sistema muestra las opciones de pantalla.
2. El usuario modifica las distintas opciones de pantalla.

4b El usuario desea modificar las opciones de controles y pulsa sobre el botón de opciones de controles.

1. El sistema muestra las opciones de controles.
2. El usuario modifica las distintas opciones de control.

Caso de uso: Salir

Caso de uso Salir

Descripción El usuario desea cerrar la aplicación.

Actores Usuario

Precondiciones Ninguna

Postcondiciones Se sale de la aplicación.

Escenario principal

1. El usuario desea salir de la aplicación.
2. El usuario pulsa la opción salir del menú principal.
3. El sistema cierra la aplicación.

Extensiones — flujo alternativo

***a** El usuario cierra la ventana de la aplicación y sale de la aplicación

4.3. Modelo conceptual de datos

Este apartado del análisis sirve para especificar los requisitos del sistema y las relaciones estáticas que existen entre ellos.

Para este fin se utiliza como herramienta los diagramas de clase. En estos diagramas se representan las clases de objetos, las asociaciones entre dichas clases, los atributos que componen las clases y las relaciones de integridad.

4.3.1. Diagrama de clases conceptuales

En este apartado se muestra una lista con las diferentes clases necesarias para la realización de sistema. Junto a cada una de las clases habrá una pequeña descripción sobre la labro que desempeña cada una.

Juego Clase principal de la aplicación, encargada de inicializar el sistema y el flujo entre unos apartados y otros.

Estado Clase virtual, con las necesidades básicas de los estados del juego.

Menú Básico Clase virtual, con las necesidades básicas de los menús.

Menú principal Clase que gestiona el menú principal.

Menú selección personaje Clase que gestiona el menú de selección de personaje.

Menú selección circuito Clase que gestiona el menú de selección de circuito.

Menú opciones Clase que gestiona el menú de opciones.

Cursor Curso de los menús.

Boton Clase que representa el botón en los menús.

Modo de juego Clase virtual, con las necesidades básicas de los modos de juego.

Carrera rápida Clase que gestiona el modo de juego carrera rápida.

Campeonato Clase que gestiona el modo de juego campeonato.

Contrarreloj Clase que gestiona el modo de juego contrarreloj.

Control de juego Clase encargada del control de la carrera, controlando la interacción del jugador con el circuito, así como el jugador con la inteligencia artificial. Aspectos básicos como colisiones, scroll de pantalla, control de vueltas, control de posiciones.

Circuito Clase encargada de cargar y dibujar el circuito.

Gestor de colisiones Clase encargada de detectar y gestionar las colisiones.

Objeto de juego Clase virtual con las necesidades básicas de los objetos del juego.

Caja de item Clase que representa las cajas que proporcionan items a los jugadores.

Vehículo básico Clase virtual con las necesidades básicas de los vehículos del juego.

IA Clase que representa el comportamiento de los vehículos controlados por inteligencia artificial.

Jugador Clase que representa al vehículo controlado por el jugador.

En la siguiente imagen podemos ver el diagrama de clases asociado a los requisitos obtenidos:

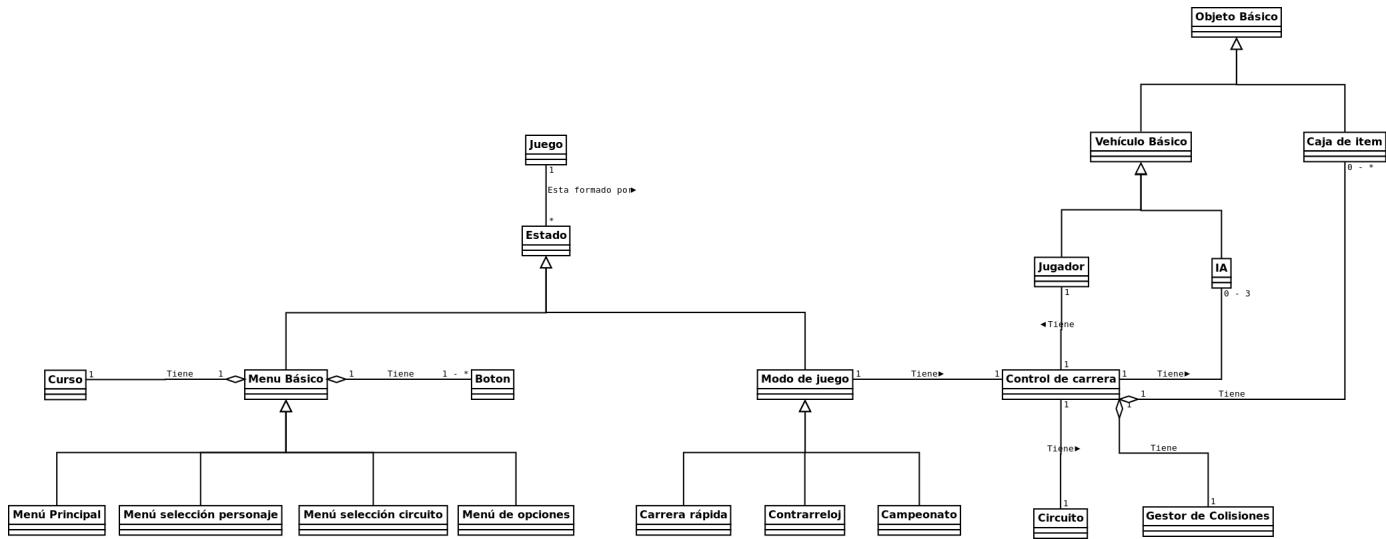


Figura 4.2: Análisis: Diagrama de clases conceptuales

4.4. Modelo de comportamiento del sistema

El modelo de comportamiento especifica como debe actuar el sistema. El sistema es el que engloba todos los objetos, y el modelo consta de dos partes:

- Diagramas de secuencias del sistema que muestra la secuencia de eventos entre el usuario y el sistema.
- Contrato de las operaciones del sistema, describen el efecto que producen las operaciones en el sistema.

4.4.1. Diagramas de secuencias del sistema y contrato de las operaciones del sistema.

Modelo de comportamiento menú principal

Diagrama de secuencia

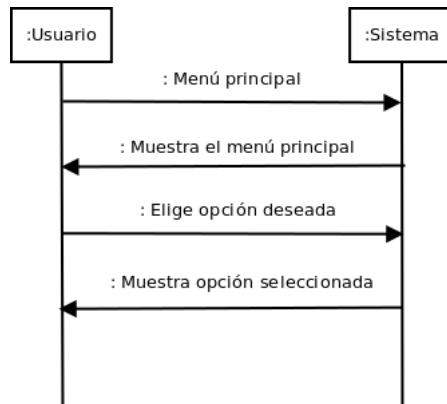


Figura 4.3: Análisis: Diagrama de secuencia Menú principal

Contrato operaciones

Operación Menú principal

Actores Usuario, Sistema

Responsabilidades Selecciona la opción principal del juego.

Precondiciones Ninguna

Postcondiciones Ninguna

Modelo de comportamiento elegir personaje

Diagrama de secuencia

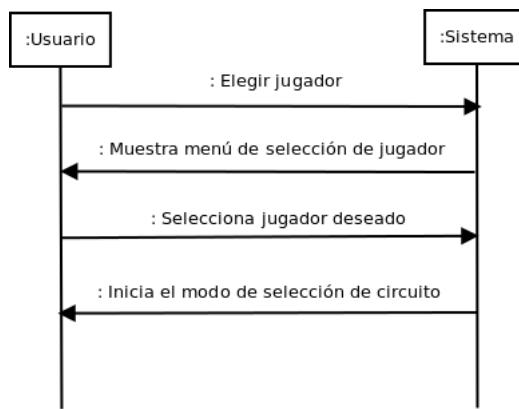


Figura 4.4: Análisis: Diagrama de secuencia Elegir jugador

Contrato operaciones

Operación Elegir personaje

Actores Usuario, sistema

Responsabilidades Selecciona el personaje a controlar en el juego.

Precondiciones Ha elegido previamente uno de los modos de juego en el menú principal.

Postcondiciones Selecciona al personaje.

Modelo de comportamiento elegir circuito

Diagrama de secuencia

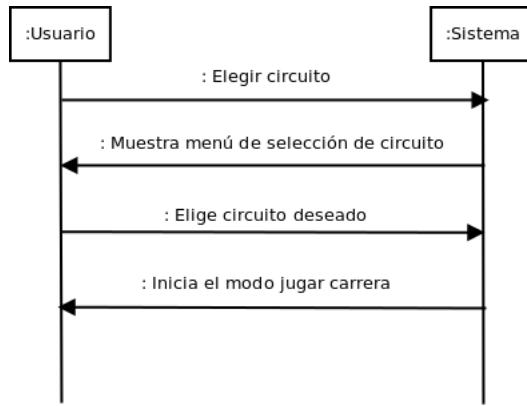


Figura 4.5: Análisis: Diagrama de secuencia Elegir circuito

Contrato operaciones

Operación Elegir circuito

Actores Usuario, sistema

Responsabilidades Selecciona el circuito donde competirá el jugador.

Precondiciones Ha elegido previamente el personaje que competirá y el modo de juego carrera rápida o contrarreloj

Postcondiciones Selecciona el circuito.

Modelo de comportamiento elegir campeonato

Diagrama de secuencia

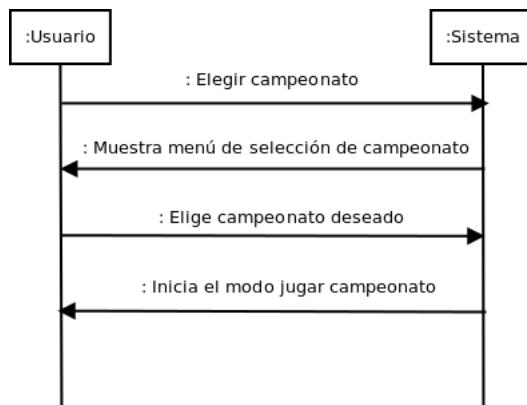


Figura 4.6: Análisis: Diagrama de secuencia Elegir campeonato

Contrato operaciones

Operación Elige campeonato

Actores Usuario, sistema

Responsabilidades Seleccionar el campeonato en el que competirá el jugador.

Precondiciones Ha elegido previamente el personaje que competirá y el modo de juego campeonato.

Postcondiciones Selecciona el campeonato.

Modelo de comportamiento jugar carrera

Diagrama de secuencia

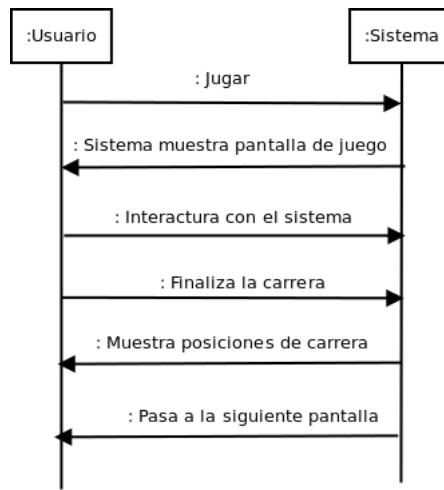


Figura 4.7: Análisis: Diagrama de secuencia Jugar carrera

Contrato operaciones

Operación Jugar

Actores Usuario, sistema

Responsabilidades Muestra la pantalla de juego desde la que el usuario puede jugar la partida.

Precondiciones Previamente se han elegido jugador y circuito.

Postcondiciones Un jugador gana la carrera.

Modelo de comportamiento jugar campeonato

Diagrama de secuencia

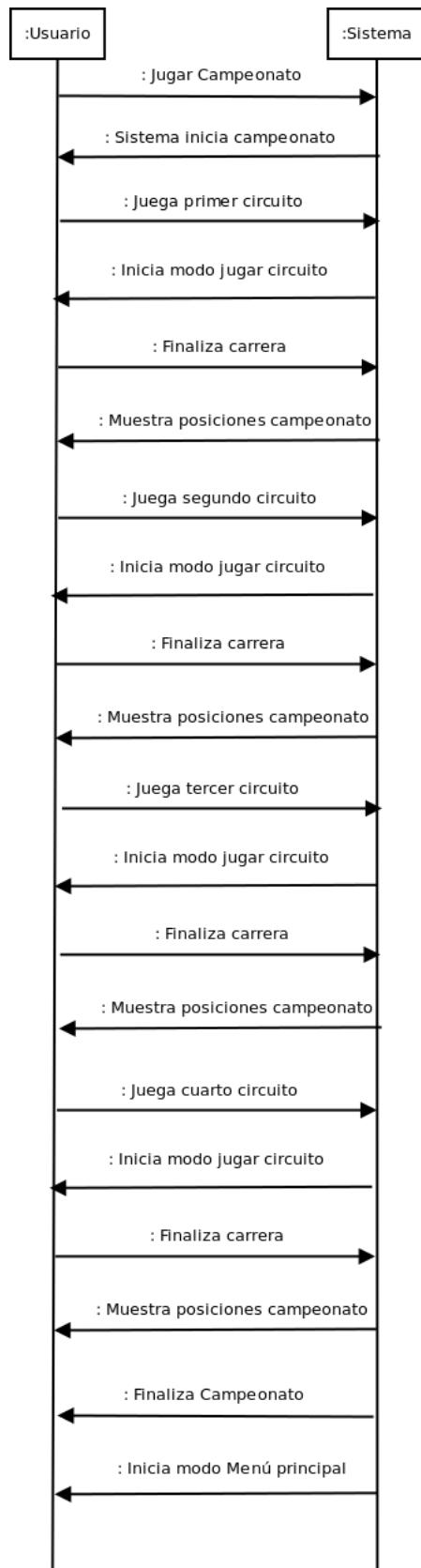


Figura 4.8: Análisis: Diagrama de secuencia Jugar campeonato

Contrato operaciones

Operación Jugar Campeonato

Actores Usuario, sistema

Responsabilidades Desarrolla todas la carreras del campeonato

Precondiciones Se eligió el modo campeonato en el menú principal.

Postcondiciones Un jugador gana el campeonato.

Modelo de comportamiento opciones

Diagrama de secuencia

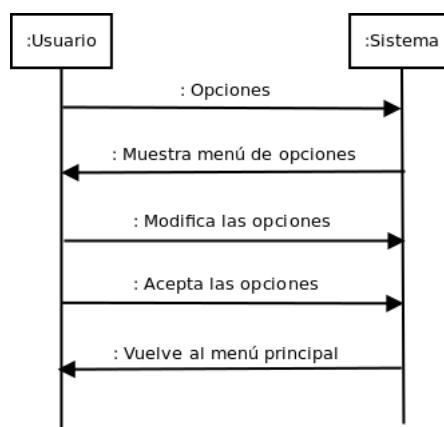


Figura 4.9: Análisis: Diagrama de secuencia opciones

Contrato operaciones

Operación Opciones

Actores Usuairo, sistema

Responsabilidades Se muestran las distintas opciones que el usuario puede modificar, como son el audio, pantalla y controles.

Precondiciones Ninguna

Postcondiciones Se modifican las opciones del juego.

Modelo de comportamiento salir

Diagrama de secuencia

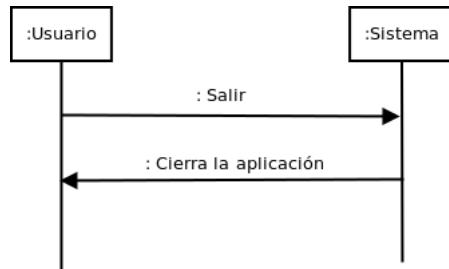


Figura 4.10: Análisis: Diagrama de secuencia Salir

Contrato operaciones

Operación Salir

Actores Usuario, Sistema

Responsabilidades Permite al usuario salir del sistema

Precondiciones Ninguna

Postcondiciones Se sale de la aplicación

Capítulo 5

Diseño

Al igual que en el apartado referente al análisis del sistema, en el diseño de este también usaremos una metodología orientada a objetos mediante UML.

Este proceso es mucho mas sencillo una vez que ya se ha especificado que hace el sistema en el capítulo anterior. También decir, que al igual que el análisis del sistema, el diseño no contempla muchos detalles del sistema final, sólo una idea orientativa de como se implementará el sistema.

En este capítulo no se han añadido descripciones de las clases que componen el sistema, para ello se encuentra disponible toda la documentación del código, donde esta toda la información necesaria referente a las clases y archivos que componen la aplicación.

5.1. Interfaz gráfica

Tras los resultado que se han obtenido en la fase de análisis del sistema, es necesario desarrollar una interfaz sencilla y agradable para el usuario de la misma. En el diseño de estos, se intentará en todo momento que el usuario no tenga la posibilidad de insertar datos erroneos que lleven a una ejecución anómala de la aplicación.



Figura 5.1: Diseño: Capturas de la interfaz del sistema

5.1.1. Diagrama de interacción entre interfaces

En la imagen que se muestra a continuación podemos observar la interacción entre las distintas interfaces graficas que se han desarrollado para la aplicación.

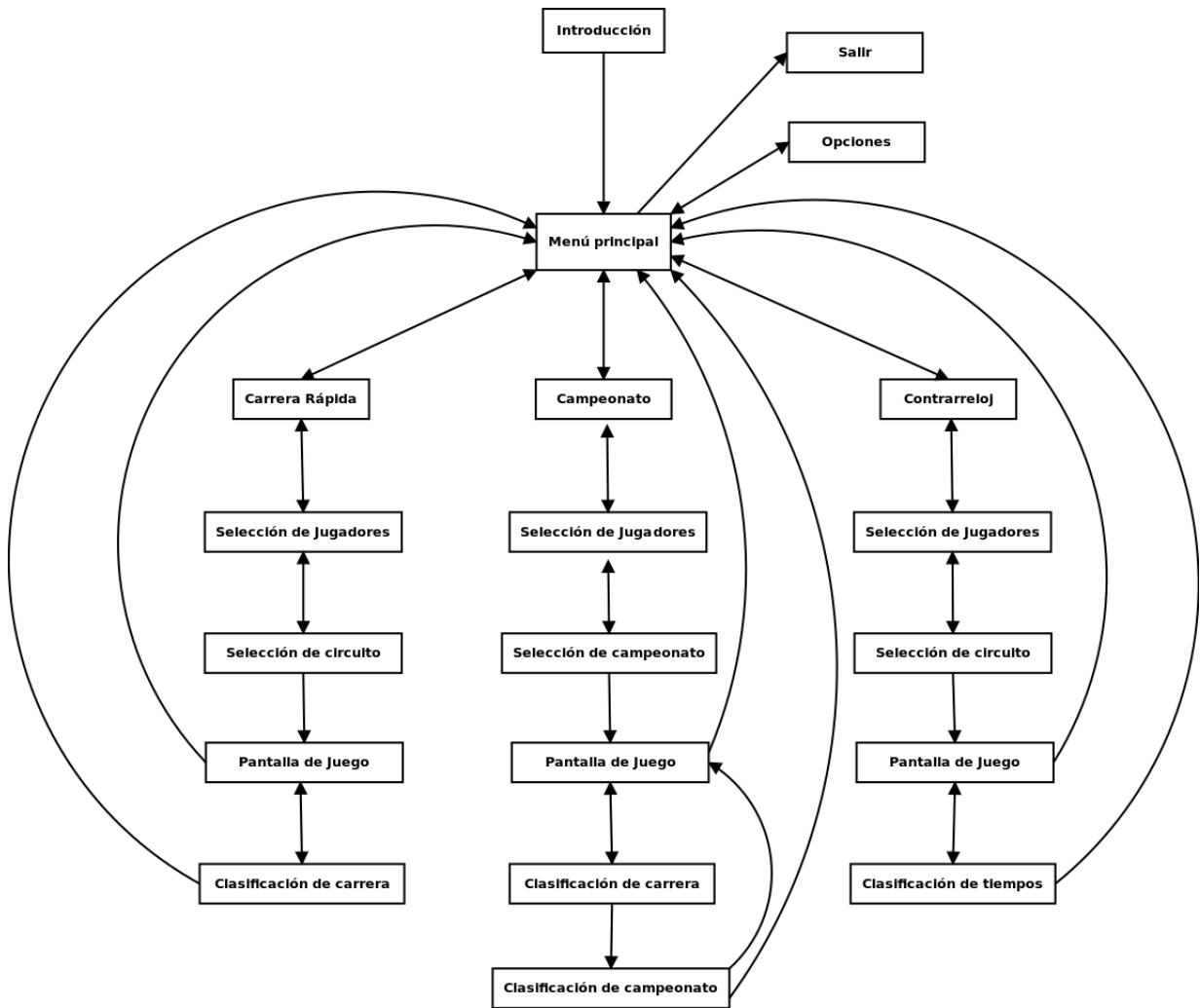


Figura 5.2: Diseño: Diagrama de interacción

5.2. Diagrama de clases de diseño

A continuación se presenta el diagrama de clases de diseño de *Zycars*.

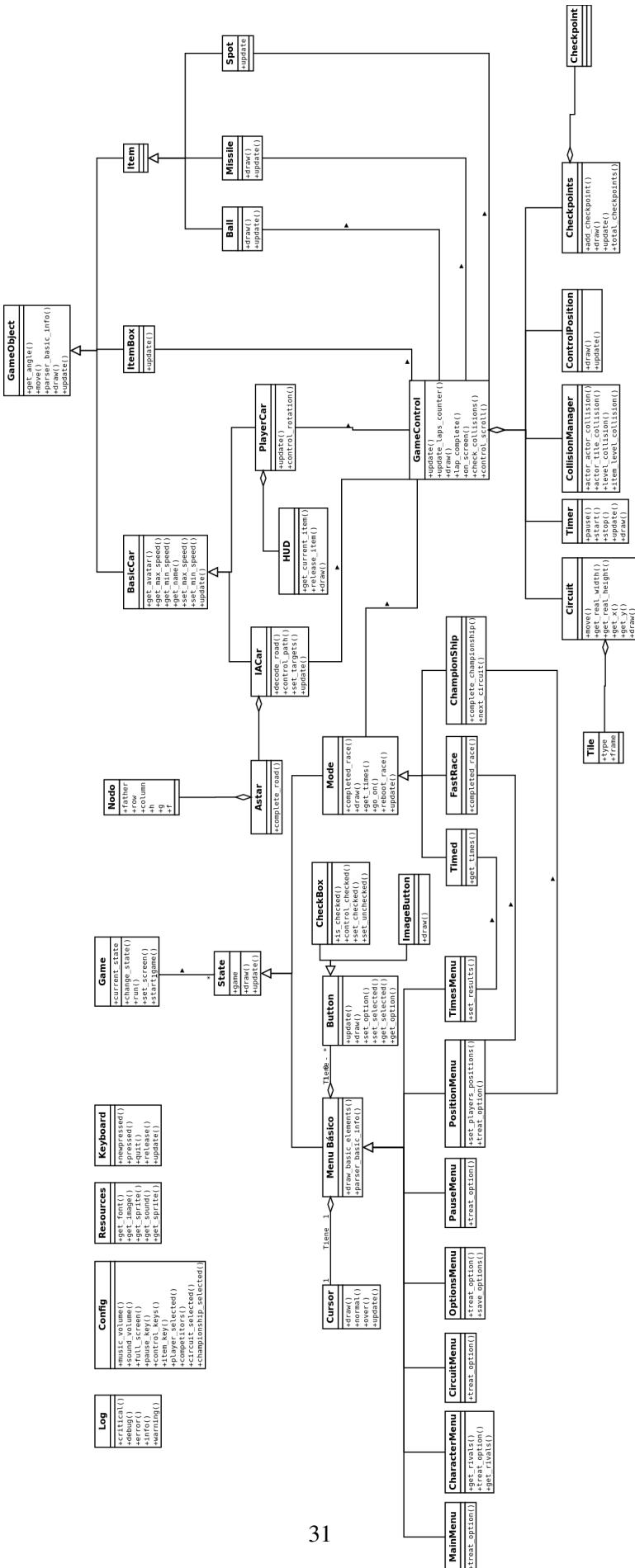


Figura 5.3: Diseño: Diagrama de clases de diseño

5.3. Diagramas de secuencia

lalalal

Capítulo 6

Implementación

Durante todo el desarrollo del proyecto, han ido apareciendo diversas dificultades y problemas que se debieron ir resolviendo para el correcto y continuo avance del proyecto.

Por lo que en este capítulo se exlicarán las distintas soluciones para los principales problemas encontrados.

6.1. Carga desde ficheros

Desde un primer momento se pensó en realizar el videojuego de forma que fuera fácilmente ampliable siguiendo un manual adecuado, donde se explicaran todos los pasos necesarios ¹.

Para ello se optó por realizar toda la carga de circuitos, personajes e interfaces de los menús desde archivos. El formato de dichos archivos sería XML.

De esta forma cualquier persona ya sea programador o no, podrá modificar aspectos tan sencillos como el posicionamiento de los botones en los menús, y las distintas imágenes que pueden aparecer en estos. Respecto a los personajes podrán modificar las características de estos, las imágenes que los representan o añadir nuevos personajes. En los circuitos podremos modificar objetos que aparezcan en estos, así como obstáculos o aparición de ítems.

Un ejemplo de fichero XML con la configuración de un menú se puede ver a continuación:

```
<mainmenu background="background_menu" cursor="cursor.xml"
music='la maryjane - bob wizman.ogg'>
    <title text="Menu Principal" font="cheesebu" size="30" r="0" g="0" b="0"
x="15" y="15"/>
    <images>
        <image image_code="separator_line" x="0" y="240"></image>
        <image image_code="logo_menu" x="130" y="5"></image>
    </images>
    <options>
        <option xml_file="menu/mainoption2.xml" font="cheesebu" text="Carrera Rapida"
center="True" x="400" y="295"></option>
        <option xml_file="menu/mainoption1.xml" font="cheesebu" text="Campeonato"
center="True" x="400" y="335"></option>
```

¹En el apéndice de este documento están los distintos manuales para añadir personajes y circuitos

```

<option xml_file="menu/mainoption2.xml" font="cheesebu" text="Contrarreloj"
center="True" x="400" y="400"></option>
<option xml_file="menu/mainoption1.xml" font="cheesebu" text="Opciones"
center="True" x="400" y="440"></option>
<option xml_file="menu/mainoption2.xml" font="cheesebu" text="Creditos"
center="True" x="400" y="505"></option>
<option xml_file="menu/mainoption1.xml" font="cheesebu" text="Salir"
center="True" x="400" y="545"></option>
</options>
</mainmenu>

```

Como podemos ver podemos modificar las imágenes que aparecen en el menú o las posiciones de los botones. En el código únicamente deberíamos dar funcionalidad a cada uno de las opciones del menú.

Otro ejemplo de archivo de XML para los personajes que aparecen en el juego sería el siguiente:

```

<car name_character="Camaro Bun" sprite_code="yellow" max_speed="6.1"
min_speed="3" acceleration="0.5" desacceleration="0.08" rotation_angle="0.35"
avatar="avatar1" racer_image='car1'>
<!--NORMAL, NOACTION, RUN, FORWARD, REVERSE, DAMAGED, ERASE, YAW-->
<animations>
    <animation name="normal" frames="0" delay="1"/>
    <animation name="noaction" frames="0" delay="1"/>
    <animation name="foward" frames="0" delay="1"/>
    <animation name="run" frames="0" delay="1"/>
    <animation name="reverse" frames="0" delay="1"/>
    <animation name="damaged" frames="0" delay="1"/>
    <animation name="erase" frames="0" delay="1"/>
    <animation name="yaw" frames="0" delay="1"/>
    <animation name="fall" frames="0" delay="1"/>
    <animation name="turbo" frames="0" delay="1"/>
</animations>
</car>

```

Como podemos ver se pueden modificar los aspectos más básicos del comportamiento de los personajes, como pueden ser la velocidad máxima de este, velocidad en marcha atrás, aceleración, desaceleración y angulo de giro. También todas las imágenes que representan al jugador.

Incluso si la imagen del coche del personaje es un sprite podemos indicar que frames de este pertenecen a cada uno de los posibles estados del personaje.

6.2. Formato y carga de circuitos

Una de las primeras dudas que surgieron al poco tiempo de comenzar el desarrollo de *Zycars*, fue el formato que deberían tener los distintos circuitos o niveles que aparecerían a lo largo del juego. Para ellos tenía varias alternativas:

- **Opción 1:** los circuitos estarían compuestos por una única imagen realizada previamente. En un fichero aparte se podría indicar las zonas colisionables que tendría la imagen u otras características relevantes.

- **Opción 2:** crear los circuitos mediante un sistema de tiles, de formas que en un fichero de texto plano, indicaramos los tiles que componen el circuito, así como la característica de estos.
- **Opción 3:** usar algún software que nos permitiera la creación y edición de niveles, de forma sencilla, mediante tiles.

Finalmente se optó por la opción 3, para ello se usó el programa *Tiled*², dicho programa me proporcionaba todas las necesidades básicas, como una sencilla edición y creación de niveles, así como la gestión de capas, para poder poner elementos en el circuito a un nivel superior o inferior. Para ello se debía crear una imagen con todos los tiles que compondrían un circuito.

Este programa generaba como resultado un archivo *XML*, que se procesaría posteriormente en tiempo de ejecución. También hay que añadir que esta opción elegida era una de las que menos nos ocuparía en memoria, ya que no es lo mismo tener un circuito completo en una única imagen, ya que para un circuito demasiado grande dicha imagen ocuparía bastante memoria. Mientras que con esta opción tendríamos una imagen pequeña con el conjunto necesario de tiles para el circuito en cuestión.

Una de las únicas cosas que no proporcionaba el programa era poder indicar cuáles de los tiles eran colisionables, atravesables o de cualquier otro tipo. Así que para solventar este problema se eligió tener a parte de la imagen que contendría el conjunto de tiles otra imagen con las mismas características, como tamaño y el tamaño de los tiles, solo que esta última los tiles tendrían colores planos indicando de qué tipo serían. Así cuando cargamos el circuito que necesitamos en ese momento y con ello el conjunto de tiles relacionado, se comprobaría qué color tiene cada uno de los tiles en la otra imagen y así almacenar de qué tipo son. A continuación se muestran dos imágenes como ejemplo:

- Este sería el aspecto de una imagen con el conjunto de tiles necesarios:



Figura 6.1: Implementación: conjunto de tiles

²Se comenta su uso y características en el apéndice relacionado con las herramientas utilizadas

- Y esta la imagen que indicaría de que tipo son cada uno de los tiles de la imagen anterior:

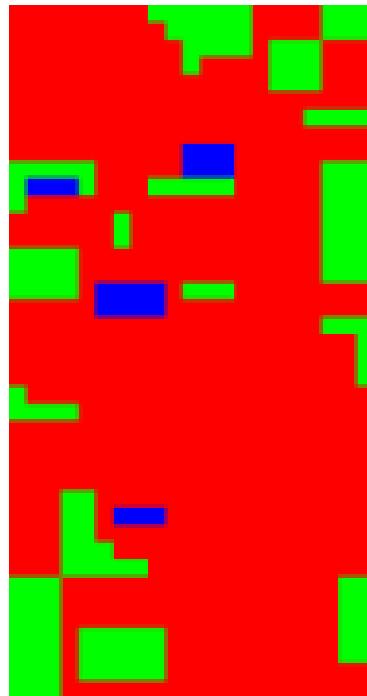


Figura 6.2: Implementación: Mapa de colisiones

Como se puede apreciar la segunda imagen tiene las mismas características que la superior, solo que los tiles que contiene son de un único color para indicar el tipo de los tiles. Los tipos de tiles que se eligieron añadir fueron los siguientes:

- **Rojo:** tiles completamente atravesables, su única función es decorativa.
- **Verde:** tiles colisionables, aquellos que no pueden ser atravesados, normalmente marcan el recorrido del circuito, también usados como obstáculos.
- **Azul:** tiles atravesables, pero realentizan de forma considerable al vehículo que pase sobre ellos.

6.3. Colisiones

La detección de las colisiones es una de las cosas más básicas de la mayoría de los juegos en la que los jugadores recorren mapas o niveles.

En Zycars debemos de gestionar varios tipos de colisiones, entre esos tipos estaría la colisión que se produciría entre cualquier vehículo y el escenario en el que se encuentre, así como la colisión entre dos objetos del juego, como podrían ser desde dos vehículos entre si, o algún vehículo por algún item lanzado por otro jugador.

Indicar que cada uno de los objetos que intervienen en el juego y puede colisionar con cualquier elemento, tienen un rectángulo asociado a su forma, de manera que sea más sencilla la detección de colisiones.

En las siguientes subsecciones se explicarán cuáles son las soluciones que se llevaron a cabo para la gestión de las colisiones en cada una de las situaciones posibles.

6.3.1. Colisión con el escenario

Como se comentó en el apartado dedicado a la carga y formato de los circuitos, cada uno de los circuitos tiene asociada una imagen que indica de qué tipo es cada uno de los tiles que nos podemos encontrar a lo largo del circuito.

Así que a la hora de cargar el circuito en el que vayamos a competir almacenábamos cada uno de los tiles que componían el circuito, así como el tipo que eran. Dada esta situación debemos ir comprobando si el jugador está atravesando algún tile colisionable. Si es el caso debemos corregir la posición del objeto con respecto al tile con el que estaba colisionando.

A la hora de realizar la corrección de la colisión, debemos tener en cuenta aspectos como, ángulo del vehículo, dirección del vehículo, así como el lado del tile por el que se produce la colisión, ya sea por la parte superior, inferior o alguno de los laterales. Según estos parámetros la colisión se corregirá en una dirección u otra.

A continuación se expone un ejemplo visual para su correcta comprensión:

- Se detecta que un vehículo colisiona con un tile colisionable:



Figura 6.3: Implementación: Colisión con el escenario 1/2

- Se corrige la colisión en función del ángulo y dirección del vehículo, así como el lado por el que colisionó el tile:



Figura 6.4: Implementación: Colisión con el escenario 2/2

En el caso de que se colisione con un tile que atravesable pero era del tipo que realentizaban la velocidad, únicamente se reducirá la velocidad del vehículo y no se corregirá su posición para este tile.

6.3.2. Colisiones entre objetos

En este caso hay varias posibilidades según que tipos de objetos han colisionado, en todas ellas la detección de la colisión se hace de forma similar, comprobamos si alguna de las caja de colisiones de los objetos en cuestión se superponen o no.

A continuación se exponen las distintas situaciones que pueden suceder:

- **Colisión vehículo-vehículo:** dos vehículos en carrera colisionan entre sí, en este caso debemos comprobar cual de los dos vehículos ha colisionado con el otro, es decir, cual se ha interpuesto. En ese caso corregiremos la posición de ese vehículo y produciremos algún tipo de rebote en función de la velocidad que llevara en ese momento.
- **Colisión vehículo-item:** en este caso se responderá a la colisión dependiendo del tipo de item con el que hemos colisionado.
 - Si el item es un misil o una bola, el item pasará a su estado de explosión, mientras que el vehículo parará a un estado de daño
 - Si el item es una mancha de aceite, el item no cambiará su estado, pero el coche pasará a un estado de descontrol durante unos instantes
 - Si el item es un chicle, se reducirá de forma considerable la velocidad del vehículo.

6.4. Inteligencia artificial

Otro de los aspectos más importantes de un videojuego de las características de *Zycars*, es la inteligencia artificial, ya que en dos de los tres modos de juegos disponibles el objetivo es obtener la mejor clasificación posible, por delante de los demás coches manejados por la inteligencia artificial.

Entre las habilidades que debe tener la inteligencia artificial deben ser:

- **Realización del recorrido:** la inteligencia artificial debe ser capaz de realizar los recorridos de los circuitos.
- **Lanzamiento de items:** también debe poder tirar los items que reciba de las bolas de items.

En los siguientes apartados se explicará de que forma se han afrontado los distintos problemas para obtener una inteligencia artificial que cumpla las expectativas básicas.

6.4.1. Realización del recorrido. Algoritmo de búsqueda A*

Esta es la parte más importante de la inteligencia artificial, ya que en un juego de conducción y carreras, lo mínimo que se espera es que la inteligencia artificial realice los recorridos de los circuitos disponibles, con el objetivo de vencer al jugador.

A lo largo de los circuitos existen unos puntos de control que cada uno de los vehículo de la inteligencia artificial debe pasar para realizar la vuelta al circuito, dichos puntos de control ocupan un tile.

Para ello, aprovechando que tenemos un circuito creado por tiles y que podemos saber en todo momento en el tile actual que se puede encontrar cualquiera de los competidores, se decidió que se implementaría el algoritmo de búsqueda A*.

Algoritmo de búsqueda A*

El objetivo del algoritmo A* es buscar el camino más corto y óptimo, en el caso de que exista, desde un nodo origen, hasta un nodo destino. A la hora de buscar dicho camino se tienen en cuenta factores como, el valor heurístico que poseen cada uno de los nodos, así como el coste real del recorrido.

Dicho algoritmo tiene la siguiente función de evaluación $f(n) = g(n) + h'(n)$, siendo $h'(n)$ el valor heurístico del nodo actual n, hasta el final y $g(n)$ el coste real del camino desde el origen al nodo actual.

7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

Figura 6.5: Implementación: Ejemplo del algoritmo A*

En el A* se tiene dos estructuras diferenciadas:

- Lista de abiertos: donde se encuentran los nodos por los que aún no se han pasado
- Lista de cerrados: donde se encuentran los nodos por los que ya se han pasado

El funcionamiento general del algoritmo es el siguiente, partiendo de un nodo en el que nos encontramos actualmente, obtenemos todos los nodos vecinos de este, comprobamos que no se encuentren en la lista de abiertos, ni en la lista de cerrado para evitar ciclos, todos aquellos que cumplan dichos requisitos se añaden a la lista de abiertos. En el caso de que alguno de los nodos se encuentren en la lista de abiertos, comparamos el valor de $f(n)$, en el caso de que sea menor lo sustituiremos. Una vez evaluado dicho nodo, pasamos a obtener de la lista de abiertos aquel nodo que tenga un $f(n)$ menor y comenzamos de nuevo todo el proceso anterior. En el momento en el que llegemos al nodo objetivo, detenemos la búsqueda y devolvemos el camino completo.

Dicho algoritmo se aplica en *Zycars*, de forma que el vehículo controlado por la inteligencia artificial, tiene todos los puntos de chequeo, en orden, que debe atravesar para recorrer el circuito completo. Así que en cada momento se comprobará el tile actual en el que se encuentra y se obtendrá el camino más óptimo y corto hasta el siguiente punto de chequeo, una vez llegado a este se hará una nueva consulta al A* para obtener el camino al próximo y así sucesivamente.

6.4.2. Lanzamiento de items.

La inteligencia artificial debe ser capaz de lanzar los items disponibles a lo largo del juego, según las distintas situaciones en la que se encuentre, en el momento que obtenga dicho item.

Por ejemplo, cuando obtenga un misil o una bala deberá lanzarlos en el momento que tenga algún oponente delante y a tiro. Pero si en cambio hemos obtenido como item una macha de aceite o un chicle, deberá lanzarlo en el momento que tenga un oponente detrás y lo más cerca posible para que no tenga tiempo para maniobrar y poder esquivar el obstáculo.

Como solución, se eligió una forma muy sencilla y eficiente a la hora de realizarlo. Para ello cada vehículo controlado por la inteligencia artificial, tiene tanto un segmento que va desde el centro del coche hacia unos píxeles por delante de la posición actual del vehículo, como otro segmento que también va desde el centro pero uno píxeles atrás de la posición del vehículo. Si se pudieran ver dichos segmentos, tendrían la siguiente forma:



Figura 6.6: Implementación: Segmentos de la inteligencia artificial

De forma que en el momento que tengamos un item que se lanza por la parte delantera del vehículo, se comprobará si alguno de los oponentes colisiona con la barra delantera de este. En el caso de que tengamos un item que se lanza por la parte trasera, haríamos la misma comparación pero con la barra trasera.

Capítulo 7

Pruebas y validaciones

Capítulo 8

Conclusiones

Apéndice A

Herramientas utilizadas

A.1. Lenguaje de programación

Una de las decisiones más importante que consideraba a la hora de desarrollar el proyecto, era la elección de un lenguaje de programación adecuado y que cumpliera todas las expectativas necesarias. En un principio se pensó en utilizar C++ por varias razones:

1. Lenguaje que hemos visto y aprendido a lo largo de toda la carrera, y con mas profundidad en la asignatura de Programación Orientada a Objetos. Por lo que ya se tenía una soltura y conocimientos previos que no se tenían con ningún otro lenguaje.
2. Lenguaje compilado por lo que su velocidad y eficiencia es mucho mayor que la de otros lenguajes. Aspectos muy a tener en cuenta a la hora de desarrollar un videojuego.

Aún así, vamos a desarrollar un juego en 2D por lo que quizás las exigencias tanto de eficiencia como velocidad no son tan altas, como las de un juego en 3D. También debía tener en cuenta que era la ocasión perfecta para poder aprender un nuevo lenguaje de programación, ya que no hay forma mejor de aprender uno que desarrollando un proyecto.

Tras evaluar varias posibilidades y todos sus virtudes y defectos, me decanté por elegir como lenguaje de programación para el desarrollo del proyecto, el lenguaje *Python*. A continuación se verán las distintas ventajas y desventajas de este lenguaje comparadas con C++:

1. Multiplataforma al igual que C++ por lo que no tendremos problemas para portar el proyecto a otras plataformas.
2. Lenguaje interpretado que a diferencia de C++, que es compilado, puede ser más lento que este.
3. Sintaxis muy limpia y que favorece a un código mucho más legible.
4. A la par de una sintaxis limpia, una gran facilidad para aprenderlo, también posee una documentación inmensa.
5. Conjunto de bibliotecas estandar muy amplia y muy bien documentadas.
6. Lenguaje multiparadigma, soporta orientación a objetos, programación imperativa y programación funcional.



Figura A.1: Herramientas utilizadas: Logo de python

Así que finalmente se decidió usar *Python* como lenguaje principal para el desarrollo de *Zycars* y cabe destacar que se han obtenido unos resultados muy satisfactorios y ha cumplido todas las expectativas esperadas.

A.2. Biblioteca gráfica

Tras la decisión final del lenguaje de programación que se usaría a lo largo de todo el desarrollo del proyecto, la siguiente decisión de gran importancia que se debía tomar, era la elección de la biblioteca gráfica que usariamos en *Zycars*.

En este caso se tuvo clara la elección desde el momento en el que se decidió usar *Python* como lenguaje principal, me decanté por la biblioteca gráfica *Pygame*. Dicha biblioteca es un wrapper de la biblioteca *SDL*, de C/C++, para *Python*, por lo que tiene todas las virtudes de dicha biblioteca:

1. Biblioteca multiplataforma compatible oficialmente con los sistemas Microsoft Windows, GNU/Linux, Mac OS y QNX.
2. Programada en C, por lo que tiene un gran rendimiento.
3. Muy completa, ya que permiten la manipulación tanto de imágenes 2D, como gestión de sonido y música, y también gestión de la entrada estandar del sistema. Todos los elementos necesarios para el desarrollo de videojuegos.
4. También se usó durante el desarrollo de la asignatura de Diseño de Videojuegos, por lo que se conocen todas sus características muy bien.



Figura A.2: Herramientas utilizadas: Logo de pygame

A parte de todas las virtudes que tiene heredadas de la *SDL*, *Pygame* tiene un nivel de abstracción más alto, por lo que el uso de esta se hace mucho más sencillo y llevadero, sin necesidad de realizar una abstracción propia, como se haría con la *SDL* al usar programación orientada a objetos.

A.3. Analizador de código: Pylint

Se creyó necesario que el código que se implementara siguiera un estandar uniforma y que estuviera exento de cualquier tipo de errores o signos de mala calidad. Para ello se usó la herramiento Pylint.

Pylint es una herramienta que analiza el código *Python* en busca de errores y señales de mala calidad. Es una herramienta de python que comprueba si un módulo cumple con un estándar de codificación.

Nos ofrece multitud de funcionalidades, como la comprobación de la longitud de las líneas de código, comprobación si los nombres de las variables están bien formadas de acuerdo a su estándar de codificación, o comprobar si las interfaces declaradas son realmente implementadas.

La gran ventaja de pylint es que es altamente configurable y personalizable, y es muy fácil escribir un pequeño plugin para añadir una característica personal.

A.4. Sistema de control de versiones

Todo el código y recursos de *Zycars* está alojado en el sistema que proporciona Google Code, que consiste en un entorno completo usando el sistema de control de versiones *subversion*.

Nos permite llevar todas las versiones y visualizar todos los cambios que se han producido durante todo el desarrollo del proyecto, entre los distintos archivos del mismo, así como poder volver a versiones anteriores en caso de necesidad y cualquier operación que permita cualquier sistema de control de versiones.

Se evaluaron otros sistema de control de versiones como pueden ser GIT o mercurial¹, pero finalmente me decanté por subversión, ya que en el proyecto yo era el único desarrollador y quizás, los otros dos sistema están mas pensados para proyecto con un número de desarrolladores más altos.

A.5. Documentación del código

Para la documentación del código me decanté por usar *Doxxygen*, esta permite la documentación sencilla y legible de todo el código, generando la documentación en varios formatos como puede ser *HTML* o *PDF*.

Para python existe la herramienta *Doxypy*, que nos permite usar la convención de comentarios de *Python* y adaptarlos a *Doxxygen*, por lo que nos ahorra trabajo y sigue la normativa de código *Python*.

También comentar que dicha herramienta se usó en proyectos anteriores con resultados muy satisfactorios.

¹Google Code también permite la gestión de proyecto mediante Mercurial

A.6. Redacción de la memoria.

Para la completa realización de la memoria se ha usado L^AT_EX. Es un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas.



Figura A.3: Herramientas utilizadas: Logo de L^AT_EX

L^AT_EX es un sistema de composición de textos que está formado mayoritariamente por órdenes (macros) construidas a partir de comandos de TeX. L^AT_EX es una herramienta práctica y útil pues, a su facilidad de uso, se une toda la potencia de TeX.

A.7. Realización de diagramas: Dia

Para la realización de todos los diagramas necesarios que aparecen a lo largo de toda la memoria se ha usado el creador de diagramas Dia.

Dia es un programa de creación de diagramas en GNU/Linux, Mac OS X, Unix y Windows, bajo la licencia GPL. Puede ser utilizado para dibujar diferentes tipos de diagramas. Actualmente cuenta con herramientas para dibujar diagramas entidad relación, diagramas UML, diagramas de flujo, diagramas de red, y muchos otros diagramas.

A.8. Programa de edición de escenarios: Tiled

Como se comenta en el capítulo referente a la implementación del proyecto, se dejó claro que los circuito y niveles que aparecerán en el juego estarán formados por tiles. Dicho formato se podría usar ficheros de texto plano indicando la posición de los tiles, pero se eligió una opción muchísimo más cómoda y con resultado mucho mejor.



Figura A.4: Herramientas utilizadas: Logo de Tiled

Para ello se eligió el programa de edición de mapas *Tiled*, el mismo es un editor de mapas de tiles de propósito general. Está hecho para ser fácil de usar, lo suficientemente flexible para trabajar con distintos motores de juegos, como RPG, carreras... *Tiled* es software libre y escrito en C++, usando la librerías gráficas QT. Las principales características son las siguientes

- Generación de XML con la información de todo el mapa.
- Soporta mapas ortogonales e isométricos.
- Objetos personalizados pueden ser colocados con precisión de píxeles.
- Agregar propiedades personalizadas a los tiles, capas u objetos del mapa.
- Redimensionar tamaño de mapas sin problemas.
- Soporta entrada y salida de plugins para abrir y guardar archivos en formatos personalizados.

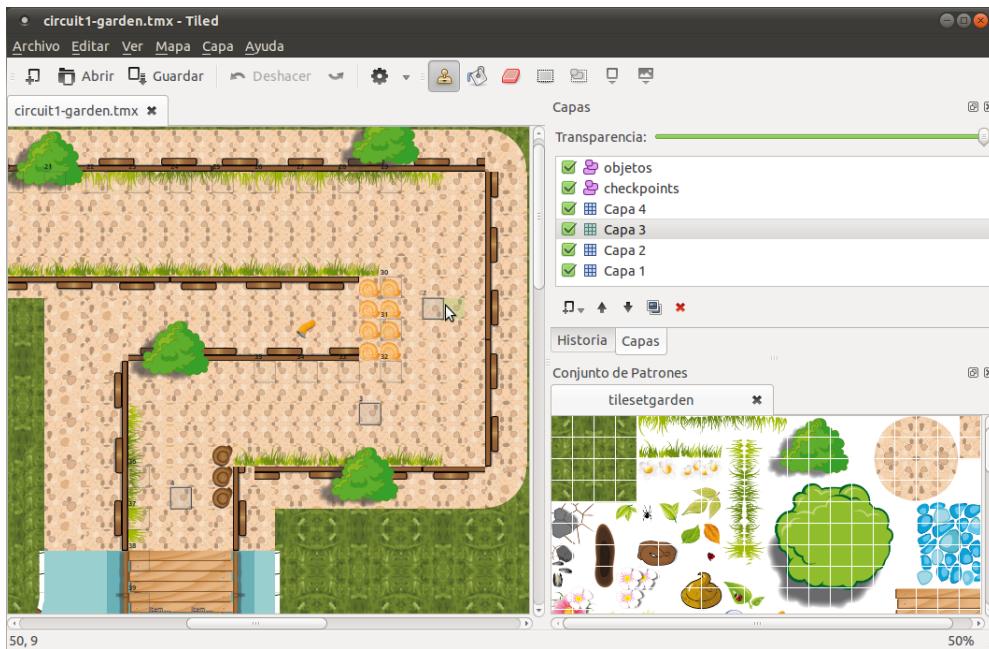


Figura A.5: Herramientas utilizadas: Captura del editor de mapas Tiled

Apéndice B

Manual de instalación

B.1. Linux: Ubuntu. Desde código fuente.

Para poder ejecutar *Zycars* desde el código fuente, será necesario la instalación de varias dependencias, para el correcto funcionamiento de la aplicación.

La primera de las dependencias a instalar será *Pygame*, que es la biblioteca principal con la que se ha desarrollado la aplicación. Para instalar, abrimos una terminal y ejecutamos el siguiente comando:

```
sudo apt-get install python-pygame
```

Una vez instalado *Pygame*, la siguiente dependencia que instalaremos será *Subversion* para poder obtener la versión más reciente del proyecto del repositorio del mismo. Para instalar subversion ejecutamos la siguiente orden en una terminal:

```
sudo apt-get install subversion
```

Tras instalar *Subversion*, hacemos checkout del repositorio del proyecto. Para ello ejecutamos en la terminal:

```
svn checkout http://zycars.googlecode.com/svn/trunk/ zycars
```

Con esto hemos obtenido la versión más reciente del código de la aplicación. Ahora accedemos a la carpeta generada anteriormente:

```
cd zycars/
```

Damos permisos de ejecución al fichero principal.

```
chmod +x run_test.py
```

Tras esto ya podremos jugar sin ningún problema haciendo doble click sobre **run_test.py** o ejecutando en la terminal:

```
./run_test.py
```

B.2. Linux: Ubuntu. Desde paquete debian.

Para poder realizar la instalación de la aplicación desde el paquete debian, debemos descargarnos el fichero debian para la arquitectura concreta de nuestro Sistema Operativo. Descargamos el fichero desde el siguiente enlace:

<http://code.google.com/p/zycars/downloads/list>

Una vez completada la descarga del fichero, hacemos doble click sobre este, y nos indicará si es necesario la instalación de algún paquete. Cuando ya estén instaladas todas las dependencias hacemos click en instalar y esperamos a la finalización de la instalación.

Para comenzar a jugar nos vamos a Aplicaciones ->Juegos ->Zycars.

B.3. Windows.

Para jugar a *Zycars* en el sistema operativo Windows no es necesario la instalación de ningún programa auxiliar, lo único que necesitaremos descargarnos será versión correspondiente a Windows y descomprimir. La descargaremos del siguiente enlace:

<http://code.google.com/p/zycars/downloads/list>

Tras descomprimir el archivo, accedemos a la carpeta generada llamada "zycars" debemos hacer doble click sobre el archivo **zycars.exe** para comenza a jugar.

Apéndice C

Manual de usuario

C.1. Menú principal

Desde el menú principal se podrá acceder a los distintos modos de juego disponibles en *Zycars*, así como las opciones del juegos y la información sobre los desarrolladores del proyecto.



Figura C.1: Manual de usuario: Menú principal

Debe usar el ratón para seleccionar la opción que deseé.

C.2. Modos de juego

En *Zycars* hay disponibles tres modos de juegos, en los que competiremos solos o contra la máquina en función del objetivo que tengamos que lograr.

C.2.1. Carrera rápida

El modo carrera rápida consiste en competir contra la inteligencia artificial en una única carrera, con el objetivo de mejorar nuestras habilidades y acostumbrarse a los controles del juego. A lo largo del circuito podremos obtener distintos items con los que hacer frente a nuestros competidores.

C.2.2. Campeonato

En el modo Campeonato competiremos contra la inteligencia artificial a lo largo de cuatro circuitos, en los que obtendremos una puntuación en relación a la posición que hayamos obtenido al concluir la carrera, 4 puntos para el ganador, 2 puntos para el segundo clasificado, 1 punto para el tercero y 0 puntos para el ultimo en concluir la carrera. El competidor que mas puntos haya conseguido al concluir el campeonato, será el ganador del mismo. En este modo también encontraremos items durante las distintas carreras.

C.2.3. Contrarreloj

En este modo de juego, el modo contrarreloj, el objetivo será batir los distintos records de tiempo que tienen cada uno de los circuitos, podremos mejorar tanto el tiempo general de la carrera, como el tiempo obtenido en la vuelta mas rápida. Tendremos un máximo de 3 vueltas para mejorar los tiempos. En este modo de juego no encontraremos items, ya que no tendremos ningún oponente al que tengamos que batir.

C.3. Menú de selección de personaje

Una vez seleccionado un modo de juego, pasaremos al menú de selección de personaje. En este menú se nos mostrarán todos los personajes disponibles en Zycars, así como el coche que cada uno de ellos conduce y las distintas características que poseen los coches.



Figura C.2: Manual de usuario: Menú selección de personaje

Con el ratón podremos navegar sobre los distintos personajes pulsando sobre las flechas rojas. Pulsemos en el botón aceptar, para elegir el personaje seleccionado. Si queremos volver al menú principal, pulsaremos sobre el botón cancelar.

C.4. Menú de selección de circuito

Una vez seleccionado el personaje con el que deseamos competir, pasaremos al menú de selección de circuito. En este menú se nos muestran los distintos campeonatos que posee el juego, así como los circuitos que componen cada uno de los campeonatos.



Figura C.3: Manual de usuario: Menú selección de circuito

Si nos encontramos en el modo carrera rápida o en el modo contrarreloj, deberemos seleccionar algún circuito de todos los disponibles, una vez elegido, pulsaremos aceptar, en el caso de que queramos volver al menú de selección de personaje pulsaremos sobre el botón cancelar.

Si estamos en el modo campeonato, podremos ver todos los circuitos que componen cada uno de los campeonatos, al pulsar sobre el botón aceptar, indicaremos que seleccionamos el campeonato actual. Si pulsamos el botón cancelar volveremos al menú de selección de personaje.

Podremos elegir, en la parte derecha del menú, el número de vueltas que queremos que realicen en cada una de las carreras. Esta opción no estará disponible en el modo campeonato, ya que en este modo siempre habrá que dar 3 vueltas al circuito.

C.5. Menú de Opciones

En el menú de opciones, podremos modificar distintos apartados como sonido, características de pantalla y controles del juego. Una vez realizados los cambios y deseamos que se apliquen debemos pulsar

el botón aceptar, si por el contrario deseamos volver al menú principal si que se aplique ninguno de los cambios realizados, debemos pulsar sobre el botón cancelar.

C.5.1. Sonido

En este menú podremos seleccionar y modificar tanto el volumen de los efectos de sonido que se encuentran en el juego, así como el volumen de la música que escuchamos a lo largo de las distintas pantallas y circuitos.



Figura C.4: Manual de usuario: Menú opciones - Audio

Como podemos ver, hay dos slider para la regulación del sonido y la música. También hay un checkbox, que nos permitirá silenciar todo, tanto los efectos de sonido como la música.

C.5.2. Pantalla

En este apartado solo dispondremos de una única opción. Esta opción nos permitira indicar si deseamos el juego a pantalla completa o si por el contrario lo deseamos al tamaño original de 800x600 píxeles.



Figura C.5: Manual de usuario: Menú opciones - Pantalla

C.5.3. Controles

En esta sección del Menú de opciones podemos modificar que controles deseamos a la hora de manejar el vehículo. Los controles que podemos modificar son los de dirección, lanzamiento de los items y pausar el juego.



Figura C.6: Manual de usuario: Menú opciones - Controles

Debemos pulsar sobre las flechas para modificar los controles que queremos usar.

C.6. Items

Durante las carreras en las que compitamos contra la máquina, a lo largo de los circuitos encontraremos unas bolas que nos proporcionarán distintos elementos con los que podremos atacar a nuestros oponentes, dejar obstáculos o aumenten nuestra velocidad durante un periodo de tiempo.



Figura C.7: Manual de usuario: Bola de item.

Los distintos item que podemos conseguir tras atravesar la bola de item se describen a continuación:

- **Misil:** este item proporciona un único misil al jugador, el cual podremos lanzar a nuestros competidores, en caso de que el misil colisione con algún jugador, este perderá el control durante unos instantes. En el caso de que el misil colisione con algún objeto colisionable explotará.



Figura C.8: Manual de usuario: Misil.

- **Misil x 3:** este item nos proporciona 3 misiles que tienen las mismas características que el misil normal, introducido anteriormente.



Figura C.9: Manual de usuario: Tres misiles.

- **Bola:** este item tiene las mismas características que un misil, la única diferencia existente es que al colisionar con algun objeto no explotará, si no que rebotará. Sólo explotará en el caso de que colisione con algún jugador.



Figura C.10: Manual de usuario: Bola.

- **Chicle:** este ítem nos proporciona un chicle que al lanzarlo en el circuito, se pegará al asfalto de forma permanente. Cualquier jugador que pase por encima de él, decrementará su velocidad.



Figura C.11: Manual de usuario: Chicle.

- **Mancha de aceite:** este ítem nos proporciona una mancha de aceite que al lanzarla quedará en el circuito y cualquier jugador que pase por encima, perderá el control del vehículo durante unos instantes.



Figura C.12: Manual de usuario: Mancha de aceite.

- **Turbo:** este ítem nos permitirá doblar nuestra velocidad durante unos instantes.



Figura C.13: Manual de usuario: Trubo.

Apéndice D

Manual para añadir nuevos personajes

En *Zycars* es posible añadir nuevos personajes por cualquier persona, siguiendo unos pasos muy sencillos, sin necesidad de tocar una sola linea de código.

A continuación se detallan los distintos pasos que debemos llevar a cabo para añadir correctamente cualquier personaje que deseemos al juego.

D.1. Imágenes necesarias

En primer lugar deberemos recopilar todas las imágenes necesarias para las distintas pantallas en las que sale el personaje, a continuación se explican las características que deben tener esas imágenes.

Antes de comenzar, dejar claro que todas las imágenes deben tener formato ".png", con cualquier otro formato se pueden tener problemas a la hora de ejecutar el juego.

Vehículo del jugador

Si duda la más importante de todas, ya que representará el vehículo que manejaremos en cualquiera de las carreras si elegimos al nuevo personaje añadido.

Por su puesto debe representar la imagen de un coche visto desde arriba, la imagen debe estar en horizontal con la parte delantera del coche mirando hacia la derecha. La dimensiones que debe tener esta imagen son 47 píxeles de ancho y 24 píxeles de alto como máximo, cualquier imagen que tenga menor medida que las anterior dadas, no habrá ningún problema. Aunque hay que tener cuidado de no añadir una imagen demasiado pequeña, para que se pueda ver correctamente. A continuación se muestra un ejemplo para que quede mas claro:



Figura D.1: Manual para añadir personajes: Ejemplo de coche con dimensiones de 42 x 18 píxeles.

Una vez que tenemos la imagen debemos añadirla a *Zycars/multimedia/images/cars/*.

Imagen de corredor.

Una vez añadido la imagen del vehículo del jugador, el siguiente paso es añadir la imagen que representará al jugador, hasta ahora la imagen de todos los corredores han tenido las siguientes características. En la imagen aparecen el corredor de cuerpo completo, junto con el vehículo que manejan. Aunque no es necesario que la imagen tenga esos componentes en concreto, pero se aconseja que sea así para que no desentoné con la imagen del resto de los corredores.

Otra cosa que debemos tener en cuenta es el tamaño de la imagen, en este caso la imagen debe tener unas medidas exactas y esta no puede ser ni mayor ni menor, de se así se podría obtener malos resultados. El tamaño que debe tener la imagen es de 403 píxeles de ancho por 246 píxeles de alto. A continuación un ejemplo:



Figura D.2: Manual para añadir personajes: Ejemplo de imagen de corredor.

La imagen debemos añadirla a la carpeta Zycars/multimedia/image/character.

Avatares del personaje

Los avatares los usaremos para representar al jugador en el menú de selección, y también al jugador en carrera. Para ello son necesarios dos avatares, ambas imágenes son idénticas y lo único que las diferencia son el tamaño que deben tener. La imagen debe mostrar el rostro claro del corredor.

Para el avatar del menú de selección de personaje debe tener el tamaño de entre 150 y 90 píxeles de ancho y 149 píxeles de alto. Dichas medidas se deben cumplir de forma exhaustiva.

Para el avatar de carrera la imagen debe tener un tamaño de 67 y 42 píxeles de ancho y 69 píxeles de alto. Un ejemplo del avatar de un corredor sería el siguiente.

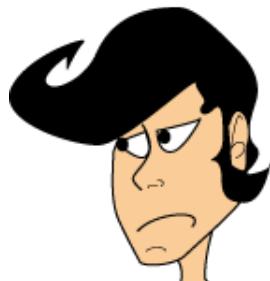


Figura D.3: Manual para añadir personajes: Ejemplo de avatar de corredor.

La imagen debemos añadirla a la carpeta Zycars/multimedia/image/character.

D.2. Añadir imágenes a los recursos del juego

Una vez que hemos reunido todas las imágenes necesarias del nuevo corredor que deseamos añadir, el siguiente paso será indicar dichas imágenes como recursos del juego para que se puedan usar sin problemas durante la ejecución de este.

Para ello debemos abrir el archivo llamado resources.xml que se encuentra en Zycars/xml, dicho archivo es muy extenso, ya que contiene todos los recursos que hace uso el juego, ya sean imágenes, sprites o sonidos.

En primer lugar añadiremos la imagen que representa al vehículo, buscamos <sprites>, y justo tras de ella añadimos la siguiente linea:

```
<sprite code="new_car" name="cars/new_car.png" rows="1" columns="1" alpha="True"/>
```

Suponiendo que la imagen del vehículo del corredor se llama "new_car.png". El atributo code, indica el código para acceder a dicho recurso, por lo que no se debe repetir.

El siguiente paso será añadir las imágenes referentes al personaje, como los avatares y el del personaje completo. Para ello en el fichero resources.xml buscamos la etiqueta <image> y añadimos las siguientes líneas tras esta:

```
<image code="character_image" name="character/character_image.png" alpha="True"/>
<image code="big_avatar" name="character/big_avatar.png" alpha="True"/>
<image code="small_avatar" name="character/small_avatar.png" alpha="True"/>
```

Suponiendo que la imagen completa del personaje se llama "character_image.png", el nombre del avatar grande sea "big_avatar.png" y el nombre del avatar pequeño sea "small_avatar.png".

Una vez hecho esto ya hemos añadido todos los recursos necesarios que necesita el personaje. Debemos tener cuidado de nuevo con el código de cada una de las imágenes y que no estén repetidas en ningún otro recurso.

D.3. Creación del fichero del personaje

El siguiente paso que deberemos llevar a cabo será crear el fichero del personaje donde indicaremos las principales características de este como puede ser la velocidad, ángulo de giro, etc. La plantilla para dicho fichero será la siguiente:

```
<car name_character="" sprite_code="" racer_image="" avatar="" max_speed=""
min_speed="" acceleration="" desacceleration="" rotation_angle="">
<!--NORMAL, NOACTION, RUN, FORWARD, REVERSE, DAMAGED, ERASE, YAW-->
<animations>
    <animation name="normal" frames="0" delay="1"/>
    <animation name="noaction" frames="0" delay="1"/>
```

```

<animation name="foward" frames="0" delay="1"/>
<animation name="run" frames="0" delay="1"/>
<animation name="reverse" frames="0" delay="1"/>
<animation name="damaged" frames="0" delay="1"/>
<animation name="erase" frames="0" delay="1"/>
<animation name="yaw" frames="0" delay="1"/>
<animation name="fall" frames="0" delay="1"/>
<animation name="turbo" frames="0" delay="1"/>
</animations>
</car>
```

Sólo debemos preocuparnos de las dos primeras líneas en las que deberemos rellenar los siguientes atributos:

- **name_character**: En este atributo deberemos poner el nombre que tendrá nuestro personaje.
- **sprite_code**: Aquí deberemos poner el código de la imagen que representará al vehículo del personaje.
- **racer_image**: En este otro deberemos poner el código de la imagen que representa al jugador completo.
- **avatar**: Debemos poner el código del avatar que representará al jugador en carrera, es decir, el del avatar pequeño.
- **max_speed**: Velocidad máxima del personaje, su valor debe estar entre 5.7 y 6.5.
- **min_speed**: Velocidad marcha atrás del personaje, valor entre 2.8 y 3.4.
- **acceleration**: Aceleración del vehículo, valor entre 0.1 y 1.
- **desacceleration**: Desaceleración del vehículo cuando no se acelera y ni se da marcha atrás. Valor entre 0.05 y 0.1.
- **rotation_angle**: Ángulo de rotación, valor recomendado entre 0.2 y 0.4.

Una vez completado el fichero del personaje, deberemos guardarlo con extensión ".xml" y almacenarlo en el directorio zycars/xml/cars.

D.4. Añadir al personaje para que sea seleccionable.

Tras todos los pasos anteriores, el último paso para poder manejar y disfrutar del nuevo personaje que hemos añadido se explica a continuación.

Nos vamos al fichero que se encuentra en zycars/xml/menu/chartermenu.xml y lo abrimos. Dentro de este buscamos la siguiente linea:

```
<characters normal_image='normal_box2' selected_image='selected_box2'>
```

Justo detrás de esta linea deberemos añadir otra linea que tenga la siguiente forma:

```
<character image="" name='' image_car='' path_xml="cars/"  
speed=''' acceleration=''' rotation=''' />
```

Los distintos parametros que debemos completar se explican a continuación:

- **image**: Código de la imagen del avatar grande que representa al personaje.
- **name**: Nombre del personaje, debe coincidir con el que añadimos en el fichero del personaje.
- **image_car**: Código de la imagen que representa al jugador completa.
- **path_xml**: Añadir el nombre del fichero con las características del jugador.
- **speed**: Velocidad del jugador para que se vea en el menú.
- **acceleration**: Aceleración del jugador para que se vea en el menú.
- **rotation**: Rotación del jugador para que se vea en el menú.

Una vez seguido todos los pasos anteriores, ya podremos disfrutar del nuevo jugador que hemos añadido.

Bibliografía

- [1] Página oficial sobre el lenguaje de programación *Python*.
<http://www.python.org/>.
- [2] Pilgrim, Mark. *Dive into Python*. Apress, 2004. 413p. ISBN:978-1590593561.
- [3] Larman, Craig. *Applying UML and Patterns*, 3^a Edición. Prentice Hall, 2004. 736p. ISBN:978-0131489066.
- [4] Página oficial de la herramienta para documentar código *Doxxygen*.
<http://www.stack.nl/~dimitri/doxygen/>
- [5] Lambert M. Surhone; Mariam T. Tenroe Y Susan F. Henssonow (Ed). *Doxxygen*. Betascript Publishing, 2010. 168p. ISBN:978-3639910025.
- [6] Guía para la generación de la memoria del Proyecto Fin de Carrera.
http://osl2.uca.es/wikiformacion/index.php/LaTeX_para_Proyecto_Fin_de_Carrera.
- [7] Página oficial sobre la herramienta *BOUML*.
<http://bouml.free.fr/index.html>
- [8] Foguel, Karl. *Producing Open Source Software*, 1^a edición. O'Reilly Media, 2005. 304p. ISBN:978-0596007591.
- [9] Página oficial de la herramienta de edición de imágenes *GIMP*.
<http://www.gimp.org/>
- [10] Peck, Akkana. *Beginning GIMP : from novice to professional*, 2^a edición. Apress, 2008. 584p. ISBN:978-1-4302-1070-2

GNU Free Documentation License

Version 1.3, 3 November 2008
Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with … Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.