



## **ESCUELA SUPERIOR DE INGENIERÍA**

### **INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS**

**ZYCARS: JUEGO DE CONDUCCIÓN 2D**

José Jesús Marente Florín

30 de agosto 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 agosto de 2011

Fdo: José Jesús Marente Florín



## **Agradecimientos**

Me gustaría dar las gracias a todos esos compañeros que he conocido durante la carrera y me han ayudado a lo largo la misma y desarrollo de este proyecto. Así como a mi familia, pareja y amigos por todo el apoyo que me han dado. También querría dar las gracias al tutor del proyecto Manuel Palomo por el apoyo, supervisión y consejos que me ha dado. Por último a David Nieto Rojas que ha realizado todo el diseño gráfico del proyecto.



## **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:

```
print "Hola mundo"
```

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

```
sudo apt-get install
```



# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Objetivos . . . . .	1
1.3. Estructura del documento . . . . .	1
<b>2. Descripción general del proyecto</b>	<b>3</b>
2.1. Descripción . . . . .	3
2.2. Características del videojuego . . . . .	3
2.2.1. Modos de juego . . . . .	3
2.2.2. Elementos de juego . . . . .	4
2.3. Colaboradores . . . . .	5
<b>3. Planificación</b>	<b>7</b>
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. Redacción de la memoria . . . . .	8
3.7. Diagrama de Gantt . . . . .	8
<b>4. Análisis</b>	<b>11</b>
4.1. Especificación de requisitos del sistema . . . . .	11
4.1.1. Requisitos de interfaces externas . . . . .	11
4.1.2. Requisitos funcionales . . . . .	17
4.1.3. Requisitos de rendimiento . . . . .	18
4.1.4. Restricciones de diseño . . . . .	18
4.1.5. Requisitos del sistema software . . . . .	18
4.2. Modelo de casos de uso . . . . .	19
4.2.1. Diagrama de los casos de uso . . . . .	19
4.2.2. Descripción de los casos de uso . . . . .	19
4.3. Modelo conceptual de datos . . . . .	26
4.3.1. Diagrama de clases conceptuales . . . . .	26
4.4. Modelo de comportamiento del sistema . . . . .	29
4.4.1. Diagramas de secuencia y contrato de las operaciones del sistema. . . . .	30
<b>5. Diseño</b>	<b>45</b>
5.1. Interfaz gráfica . . . . .	45
5.1.1. Diagrama de interacción entre interfaces . . . . .	45

5.2. Diagrama de clases de diseño . . . . .	46
<b>6. Implementación</b>	<b>51</b>
6.1. Carga desde ficheros . . . . .	51
6.2. Formato y carga de circuitos . . . . .	53
6.3. Colisiones . . . . .	56
6.3.1. Colisión con el escenario . . . . .	56
6.3.2. Colisiones entre objetos . . . . .	57
6.4. Inteligencia artificial . . . . .	58
6.4.1. Realización del recorrido. Algoritmo de búsqueda A* . . . . .	58
6.4.2. Lanzamiento de ítems. . . . .	60
<b>7. Pruebas</b>	<b>61</b>
7.1. Pruebas unitarias . . . . .	61
7.1.1. Análisis del código con Pylint . . . . .	62
7.2. Pruebas de integración . . . . .	62
7.3. Pruebas de jugabilidad . . . . .	62
7.4. Pruebas de interfaz . . . . .	63
<b>8. Conclusiones</b>	<b>65</b>
8.1. Resumen de objetivos . . . . .	65
8.2. Conclusiones personales . . . . .	65
8.3. Mejoras y ampliaciones . . . . .	66
<b>A. Herramientas utilizadas</b>	<b>67</b>
A.1. Lenguaje de programación . . . . .	67
A.2. Biblioteca gráfica . . . . .	68
A.3. Analizador de código: Pylint . . . . .	69
A.4. Sistema de control de versiones . . . . .	69
A.5. Documentación del código . . . . .	69
A.6. Redacción de la memoria . . . . .	70
A.7. Realización de diagramas: Dia . . . . .	70
A.8. Programa de edición de escenarios: Tiled . . . . .	70
<b>B. Manual de instalación</b>	<b>73</b>
B.1. Windows . . . . .	73
B.2. Linux: Ubuntu. Desde paquete Debian . . . . .	73
B.3. Linux: Ubuntu. Desde código fuente . . . . .	74
<b>C. Manual de usuario</b>	<b>75</b>
C.1. Menú principal . . . . .	75
C.2. Modos de juego . . . . .	75
C.2.1. Carrera rápida . . . . .	76
C.2.2. Campeonato . . . . .	76
C.2.3. Contrarreloj . . . . .	76
C.3. Menú de selección de personaje . . . . .	76
C.4. Menú de selección de circuito . . . . .	77
C.5. Pantalla de juego . . . . .	77
C.6. Menú de Opciones . . . . .	78
C.6.1. Sonido . . . . .	78

C.6.2. Pantalla . . . . .	79
C.6.3. Controles . . . . .	80
C.7. Ítems . . . . .	80
<b>D. Manual para añadir nuevos personajes</b>	<b>83</b>
D.1. Imágenes necesarias . . . . .	83
D.2. Añadir imágenes a los recursos del juego . . . . .	85
D.3. Creación del fichero del personaje . . . . .	85
D.4. Añadir al personaje para que sea seleccionable. . . . .	86
<b>E. Manual para añadir nuevos circuitos</b>	<b>89</b>
E.1. Descarga e instalación de Tiled . . . . .	89
E.2. Creando nuestro circuito . . . . .	89
E.2.1. Familiarización con la interfaz . . . . .	90
E.2.2. Tipos de tiles. . . . .	92
E.2.3. Añadiendo el fondo. Capa 1. . . . .	93
E.2.4. Añadiendo trazado circuito. Capa 2. . . . .	94
E.2.5. Añadiendo decorado. Capa 2. . . . .	94
E.2.6. Añadiendo elementos superiores. Capa 3. . . . .	95
E.2.7. Rellenado la capa checkpoints. . . . .	96
E.2.8. Rellenado la capa objetos. . . . .	100
E.2.9. Modificando el xml generado. . . . .	102
E.2.10. Añadiendo el mapa al juego. . . . .	102
<b>Bibliografia y referencias</b>	<b>103</b>
<b>GNU Free Documentation License</b>	<b>107</b>
1. APPLICABILITY AND DEFINITIONS . . . . .	107
2. VERBATIM COPYING . . . . .	108
3. COPYING IN QUANTITY . . . . .	108
4. MODIFICATIONS . . . . .	109
5. COMBINING DOCUMENTS . . . . .	110
6. COLLECTIONS OF DOCUMENTS . . . . .	111
7. AGGREGATION WITH INDEPENDENT WORKS . . . . .	111
8. TRANSLATION . . . . .	111
9. TERMINATION . . . . .	111
10. FUTURE REVISIONS OF THIS LICENSE . . . . .	112
11. RELICENSING . . . . .	112
ADDENDUM: How to use this License for your documents . . . . .	112



# Indice de figuras

2.1. Descripción: Logo de Zycars . . . . .	3
2.2. Descripción: Personaje de Zycars. . . . .	4
2.3. Descripción: Caja de ítem. . . . .	5
3.1. Planificación: Diagrama de Gantt 1/2. . . . .	9
3.2. Planificación: Diagrama de Gantt 2/2. . . . .	10
4.1. Análisis: Boceto del menú principal . . . . .	12
4.2. Análisis: Boceto del menú de opciones . . . . .	13
4.3. Análisis: Boceto de la pantalla de créditos . . . . .	13
4.4. Análisis: Boceto del menú de selección de personaje . . . . .	14
4.5. Análisis: Boceto del menú de selección de circuito . . . . .	14
4.6. Análisis: Boceto de ventana de juego . . . . .	15
4.7. Análisis: Boceto de menú de pausa . . . . .	16
4.8. Análisis: Boceto de la ventana de posiciones . . . . .	16
4.9. Análisis: Boceto de la ventana de tiempos . . . . .	17
4.10. Análisis: Diagrama de casos de uso . . . . .	20
4.11. Análisis: Diagrama de clases conceptuales 1 . . . . .	28
4.12. Análisis: Diagrama de clases conceptuales 2 . . . . .	29
4.13. Análisis: Diagrama de secuencia Menú principal (escenario principal) . . . . .	30
4.14. Análisis: Diagrama de secuencia Menú principal (escenario 2a) . . . . .	31
4.15. Análisis: Diagrama de secuencia Menú principal (escenario 2b) . . . . .	32
4.16. Análisis: Diagrama de secuencia Menú principal (escenario 2c) . . . . .	33
4.17. Análisis: Diagrama de secuencia Elegir personaje (escenario principal) . . . . .	34
4.18. Análisis: Diagrama de secuencia Elegir circuito (escenario principal) . . . . .	35
4.19. Análisis: Diagrama de secuencia Elegir campeonato (escenario principal) . . . . .	36
4.20. Análisis: Diagrama de secuencia Jugar carrera (escenario principal) . . . . .	37
4.21. Análisis: Diagrama de secuencia Pausar(escenario principal)	38
4.22. Análisis: Diagrama de secuencia Pausar (escenario *b)	39
4.23. Análisis: Diagrama de secuencia (escenario ) . . . . .	40
4.24. Análisis: Diagrama de secuencia Lanzar ítem (escenario principal) . . . . .	41
4.25. Análisis: Diagrama de secuencia Mover vehículo (escenario principal) . . . . .	41
4.26. Análisis: Diagrama de secuencia opciones (escenario principal) . . . . .	42
4.27. Análisis: Diagrama de secuencia Créditos(escenario principal) . . . . .	43
5.1. Diseño: Capturas de la interfaz del sistema . . . . .	45
5.2. Diseño: Diagrama de interacción . . . . .	46
5.3. Diseño: Diagrama de clases de diseño 1 . . . . .	47
5.4. Diseño: Diagrama de clases de diseño 2 . . . . .	48

6.1. Implementación: conjunto de tiles . . . . .	55
6.2. Implementación: Mapa de colisiones . . . . .	55
6.3. Implementación: Colisión con el escenario 1/2 . . . . .	57
6.4. Implementación: Colisión con el escenario 2/2 . . . . .	57
6.5. Implementación: Ejemplo del algoritmo A* . . . . .	59
6.6. Implementación: Segmentos de los coches dirigidos por el ordenador . . . . .	60
A.1. Herramientas utilizadas: Logo de python . . . . .	68
A.2. Herramientas utilizadas: Logo de pygame . . . . .	68
A.3. Herramientas utilizadas: Logo de L <sup>A</sup> T <sub>E</sub> X . . . . .	70
A.4. Herramientas utilizadas: Logo de Tiled . . . . .	70
A.5. Herramientas utilizadas: Captura del editor de mapas Tiled . . . . .	71
C.1. Manual de usuario: Menú principal . . . . .	75
C.2. Manual de usuario: Menú selección de personaje . . . . .	76
C.3. Manual de usuario: Menú selección de circuito . . . . .	77
C.4. Manual de usuario: Pantalla de juego . . . . .	78
C.5. Manual de usuario: Menú opciones - Audio . . . . .	79
C.6. Manual de usuario: Menú opciones - Pantalla . . . . .	79
C.7. Manual de usuario: Menú opciones - Controles . . . . .	80
C.8. Manual de usuario: Bola de ítem. . . . .	80
C.9. Manual de usuario: Misil. . . . .	81
C.10. Manual de usuario: Tres misiles. . . . .	81
C.11. Manual de usuario: Bola. . . . .	81
C.12. Manual de usuario: Chicle. . . . .	81
C.13. Manual de usuario: Macha de aceite. . . . .	82
C.14. Manual de usuario: Turbo. . . . .	82
D.1. Manual para añadir personajes: Ejemplo de coche con dimensiones de 42 x 18 píxeles. . . . .	83
D.2. Manual para añadir personajes: Ejemplo de imagen de corredor. . . . .	84
D.3. Manual para añadir personajes: Ejemplo de avatar de corredor. . . . .	84
E.1. Manual para añadir circuitos: interfaz de tiled. . . . .	90
E.2. Manual para añadir circuitos: herramientas capa de tiles. . . . .	90
E.3. Manual para añadir circuitos: herramientas capa de objetos. . . . .	91
E.4. Manual para añadir circuitos: Capas. . . . .	91
E.5. Manual para añadir circuitos: Tileset con los tipos de tiles reflejados. . . . .	92
E.6. Manual para añadir circuitos: Fondo circuito. . . . .	93
E.7. Manual para añadir circuitos: Trazado circuito. . . . .	94
E.8. Manual para añadir circuitos: Decorado circuito. . . . .	95
E.9. Manual para añadir circuitos: Capa 3 del circuito. . . . .	96
E.10. Manual para añadir circuitos: Checkpoints paso 1. . . . .	97
E.11. Manual para añadir circuitos: Checkpoints paso 2. . . . .	97
E.12. Manual para añadir circuitos: Checkpoints paso 3. . . . .	98
E.13. Manual para añadir circuitos: Checkpoints paso 4. . . . .	99
E.14. Manual para añadir circuitos: Checkpoints paso 5. . . . .	99
E.15. Manual para añadir circuitos: Checkpoints paso 6. . . . .	100
E.16. Manual para añadir circuitos: Cajas de ítems paso 1. . . . .	101
E.17. Manual para añadir circuitos: Cajas de ítems paso 2. . . . .	101
E.18. Manual para añadir circuitos: Puntos de control de la inteligencia artificial. . . . .	102

# **Capítulo 1**

## **Introducción**

### **1.1. Motivación**

Mi interés por el mundo de los videojuegos, desde muy pequeño, y tras haber cursado en la carrera la asignatura optativa de "Diseño de videojuegos", donde aprendí mucho relacionado con el desarrollo de estos, aumentó mi interés por este mundo y además el desarrollo de ellos. Por lo que desde entonces consideraba seriamente 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 fuera 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**

El objetivo del proyecto es realizar un videojuego de conducción en dos dimensiones con vista cénital<sup>1</sup>. Se podría decir que el juego tendrá tintes de juegos como Micro Machines, disponible para diversas plataformas, y del Mario Kart de Nintendo. En el siguiente capítulo se explica más detalladamente el objetivo concreto del proyecto.

Otro de los objetivos principales del proyecto, es la realización del 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 dedicar 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 nuevos personajes, como circuitos en los que competir.

### **1.3. Estructura del documento**

Este documento esta compuesto por las siguientes partes:

---

<sup>1</sup>Los elementos son vistos desde arriba

- **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 más 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.
- **Apéndices:**
  - **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 coches dirigidos por el ordenador. 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 personaje dirigidos por el ordenador 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 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 serán 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 tendrán 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.

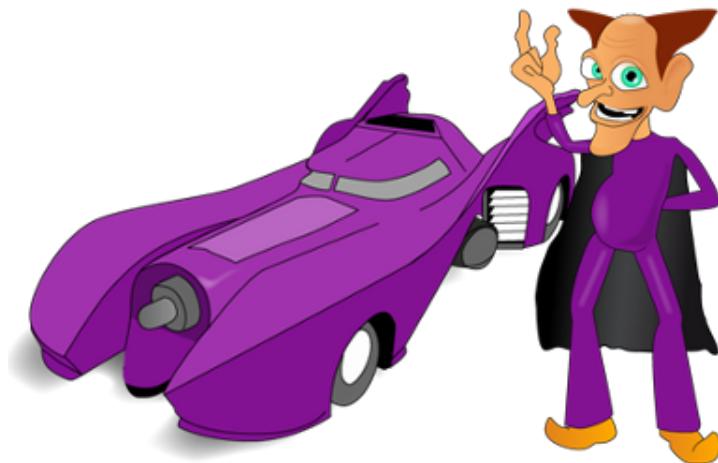


Figura 2.2: Descripción: Personaje de Zycars.

## Cajas de ítems

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 ítem que nos ayuden en la competición contra nuestros rivales.



Figura 2.3: Descripción: Caja de ítem.

## Tipos de ítems

Los ítems que podremos obtener a partir de la caja de ítems, 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** Estos nos darán la opción de aumentar nuestra velocidad durante un pequeño intervalo de tiempo.

## 2.3. Colaboradores

Todo el apartado del proyecto referente a la programación del mismo se ha realizado de forma individual. En cambio, otros apartados como el diseño gráfico, se ha contado con la colaboración de otra persona, y la música se ha obtenido de Internet, concretamente de Jamendo, la página de música libre publicadas bajo licencias Creative Commons. Los créditos de juego son los siguientes:

**Desarrollador** José J. Marente Florín

**Diseñador Gráfico** David Nieto Rojas

**Música** Bob Wizman, Pirato Ketchup, Los Cadaver, The Wavers, Zamalska



# 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 deliberación 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 siguientes:

- **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 realizaron 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 realizaron 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 más extensa de todas, como es comprensible. Y también la etapa que más subetapas contiene, las principales son las 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 compondrán 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 ítems u otros vehículos.
- **Ítems:** implementación del comportamiento y efecto que producirían cada uno de los ítems que están disponibles en el juego.
- **Inteligencia artificial:** 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 ítems.
- **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. Redacción de la memoria**

La redacción de la memoria se ha redactado conforme se iba avanzando en el desarrollo del proyecto. Pero tras la finalización de este, se le ha dedicado más tiempo a la finalización de la memoria.

### **3.7. Diagrama de Gantt**

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

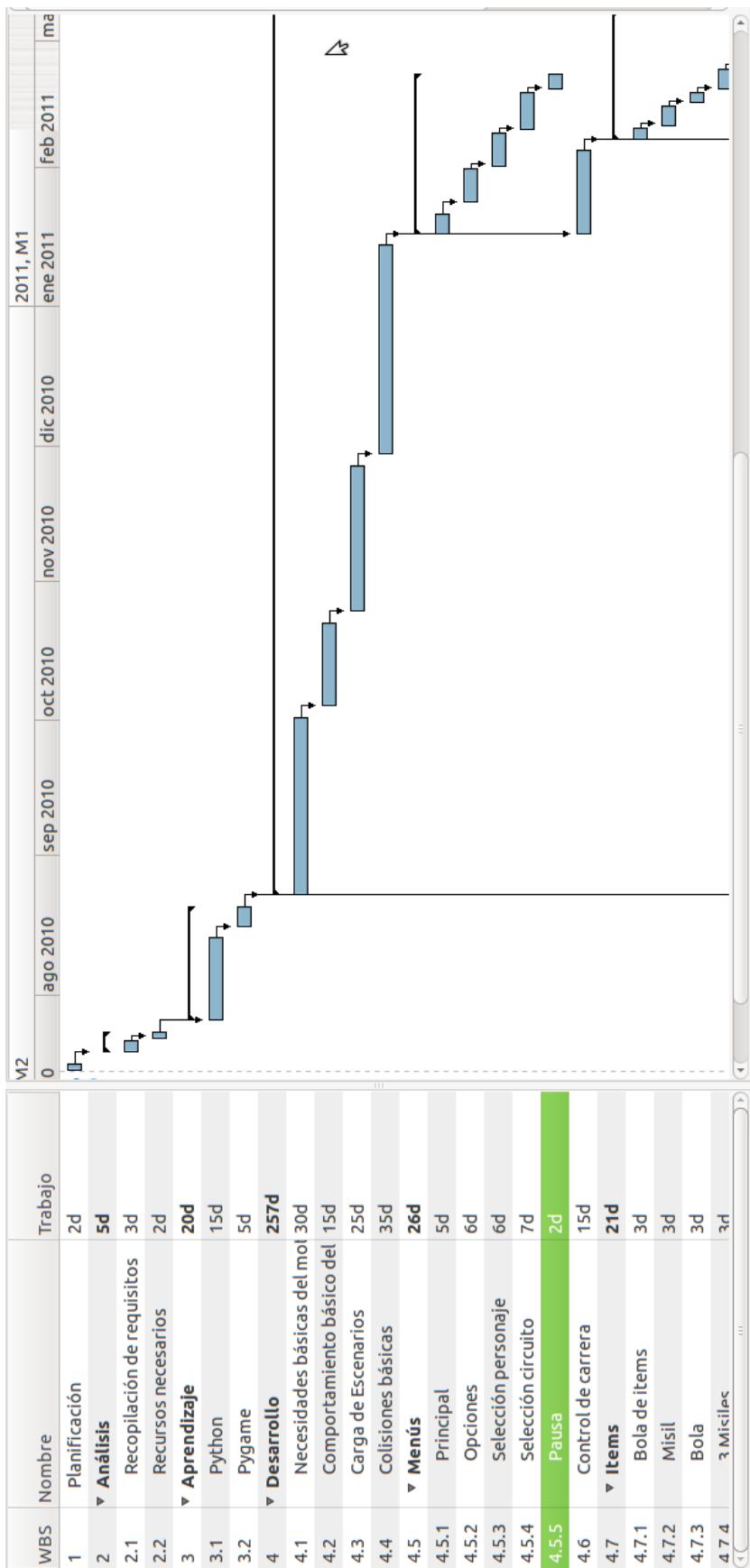


Figura 3.1: Planificación: Diagrama de Gantt 1/2.

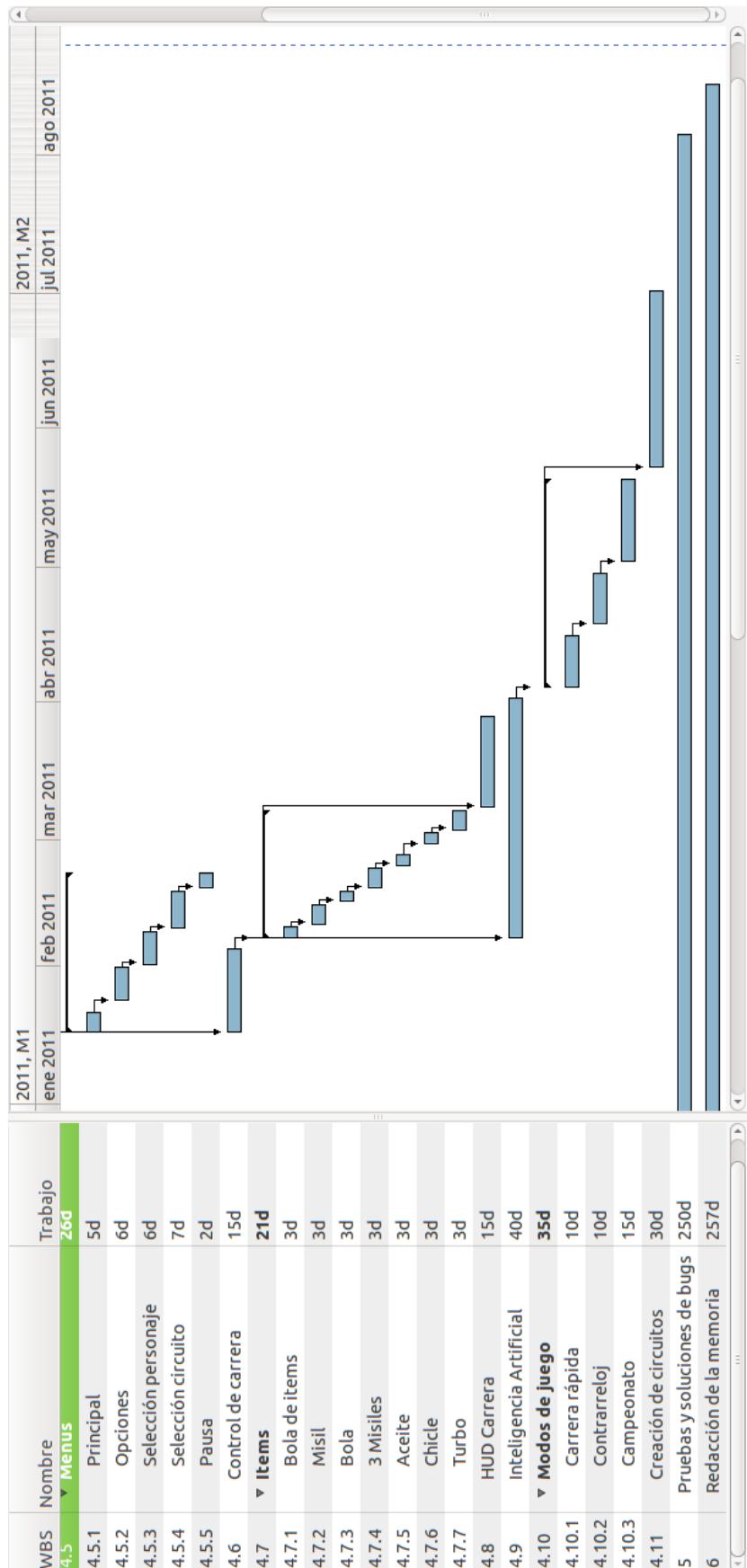


Figura 3.2: Planificación: Diagrama de Gantt 2/2.

# Capítulo 4

## Análisis

### 4.1. Especificación de requisitos del sistema

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 completa<sup>1</sup>. 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 (figura 4.1) 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
- Créditos
- Salir

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

---

<sup>1</sup>El modo de pantalla completa se podrá establecer a través del menú de opciones

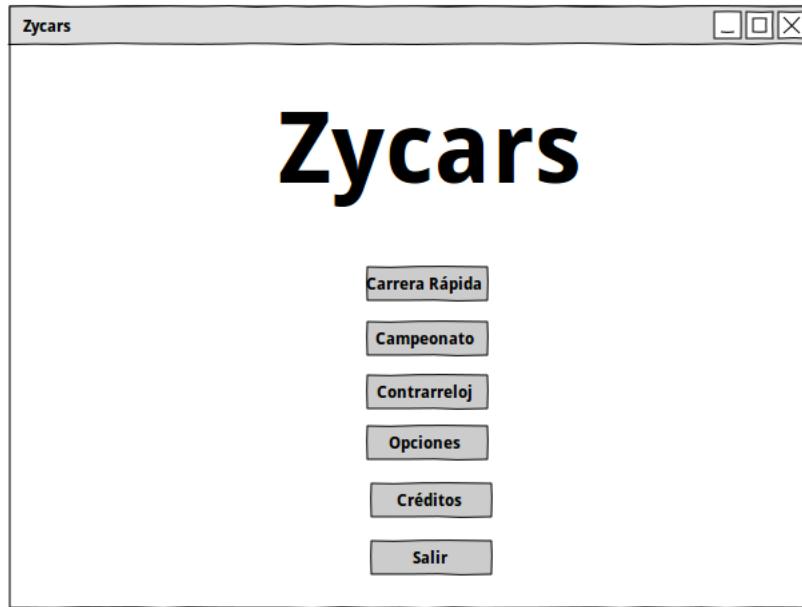


Figura 4.1: Análisis: Boceto del menú principal

**Ventana de opciones** Desde la ventana de opciones (figura 4.2) 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 ítems y pausa del juego.

**Ventana de créditos** En esta pantalla (figura 4.3) se mostrarán los creadores de *Zycars*. Tendremos a nuestra disposición un botón para volver al menú principal.

**Ventana de selección de personaje** Esta ventana (figura 4.4) 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.

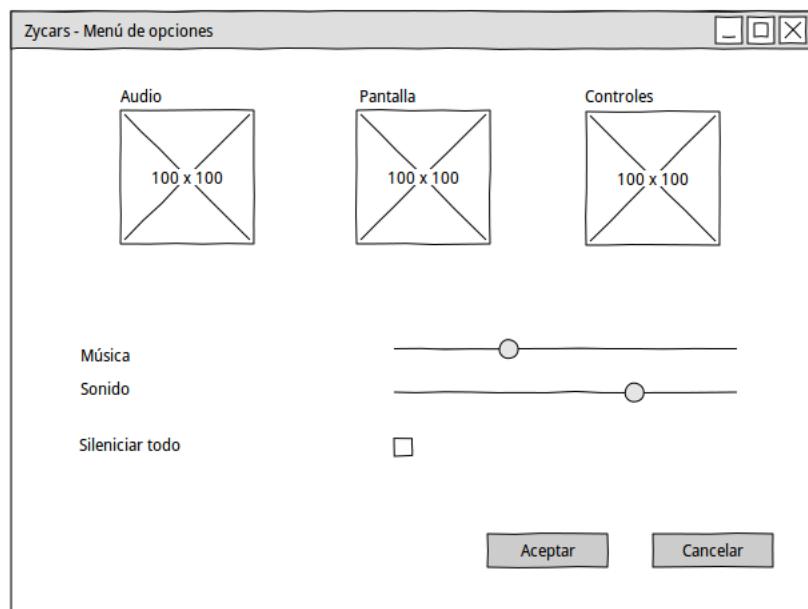


Figura 4.2: Análisis: Boceto del menú de opciones



Figura 4.3: Análisis: Boceto de la pantalla de créditos

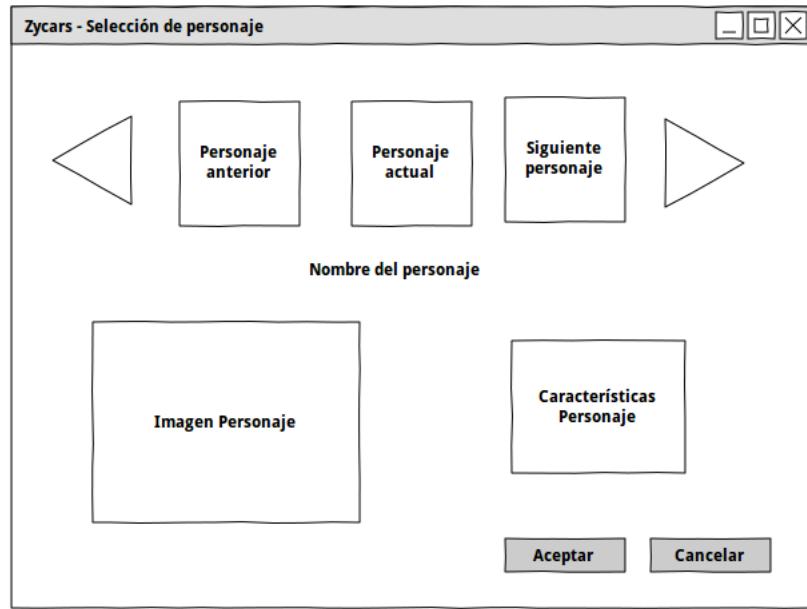


Figura 4.4: Análisis: Boceto del menú de selección de personaje

**Ventana de selección de circuito** Ventana (figura 4.5) 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.

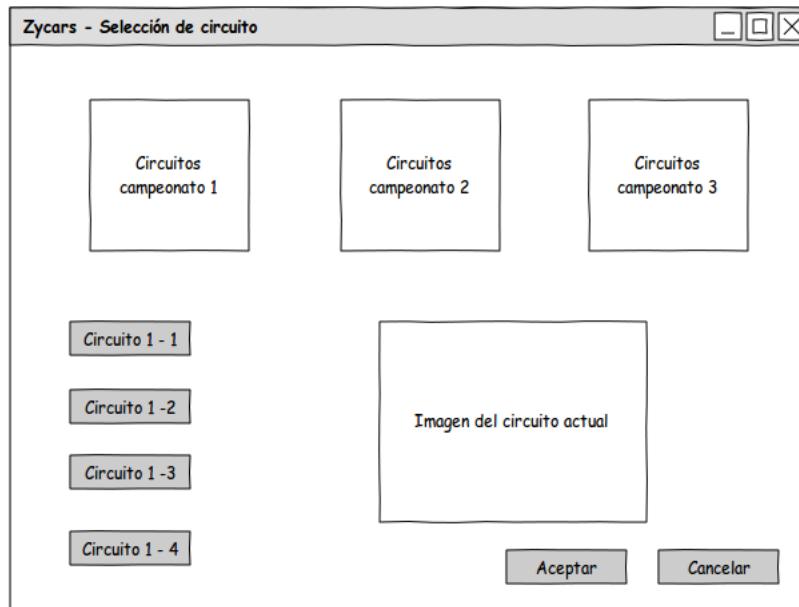


Figura 4.5: Análisis: Boceto del menú de selección de circuito

**Ventana de selección de campeonato** Ventana muy similar a la descrita anteriormente. En ella se nos

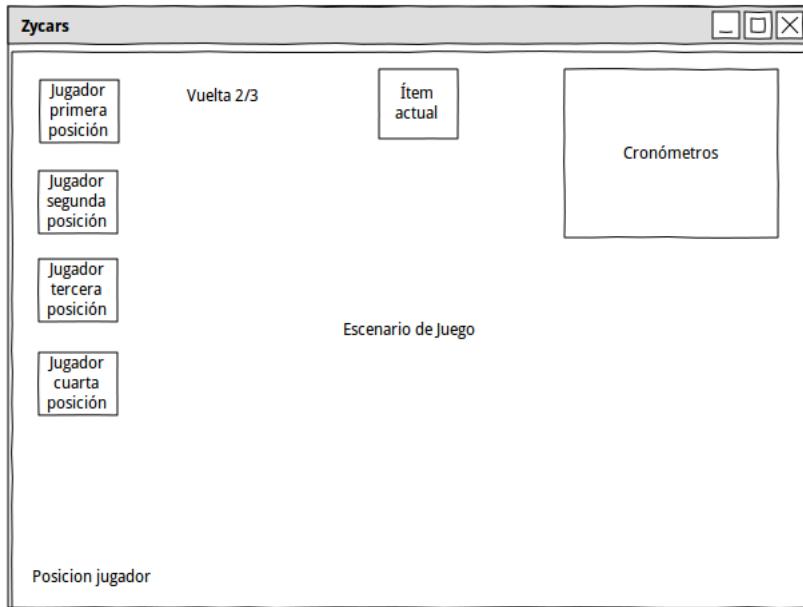


Figura 4.6: Análisis: Boceto de ventana de juego

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 (figura 4.6). 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, ítem 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 (figura 4.7). 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 (figura 4.8) 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.

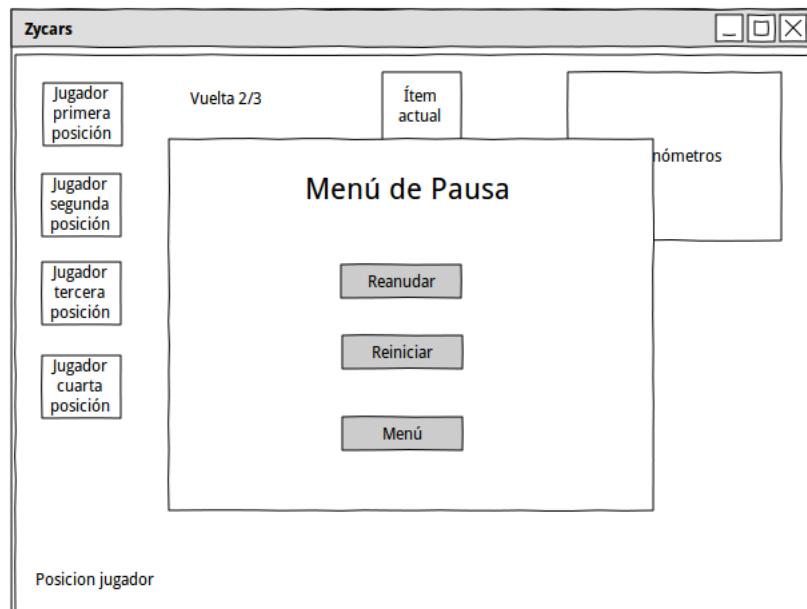


Figura 4.7: Análisis: Boceto de menú de pausa

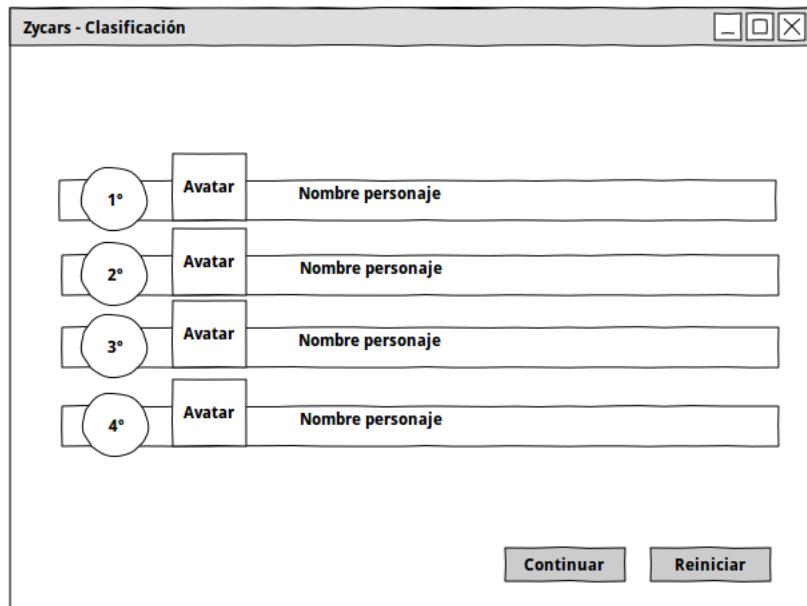


Figura 4.8: Análisis: Boceto de la ventana de posiciones

**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 (figura 4.9) 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 récord. Esta ventana nos permite continuar hacia el menú principal, así como reiniciar el circuito disputado.

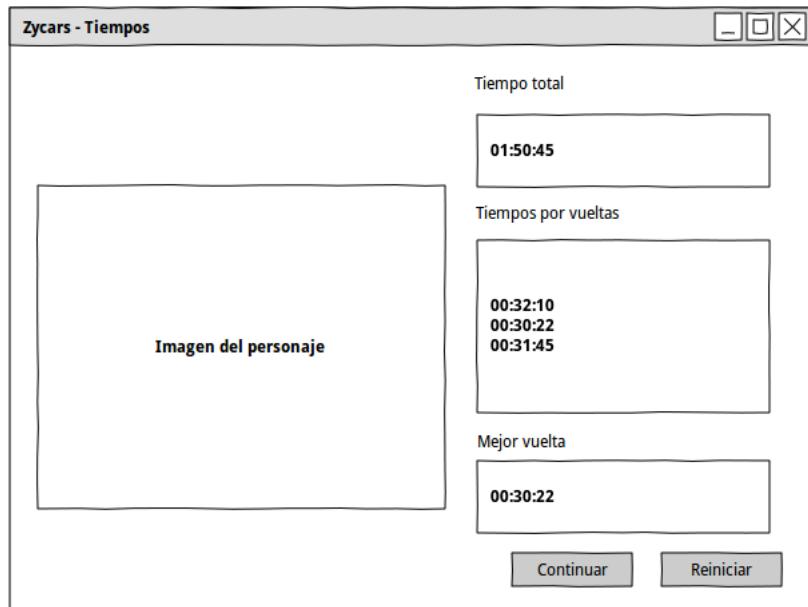


Figura 4.9: Análisis: Boceto de la ventana de tiempos

#### 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 coches dirigidos por el ordenador.
- 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 ítems durante cualquier carrera.
- Extender el juego mediante la incorporación de nuevos circuitos y personajes.

Los distintos tipos de jugadores son:

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

Existen tres modos de juego:

- **Carrera rápida:** consiste en la realización de un único circuito, compitiendo contra coches dirigidos por el ordenador.
- **Campeonato:** el jugador competirá contra coches dirigidos por el ordenador 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 más 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 ralentice 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. Requisitos del sistema software

La aplicación deberá cumplir los siguientes requisitos del sistema:

- Deberá ser multiplataforma, al menos en los siguientes 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 fácilmente 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 (figura 4.10), 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.

### 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 usuario inicia la aplicación
2. El sistema muestra el menú principal del juego en pantalla.
3. El usuario selecciona la opción **carrera rápida, campeonato o contrarreloj**.
4. 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 selección la opción **opciones**.

1. El sistema inicia las opciones

**2b** El usuario selección la opción **créditos**.

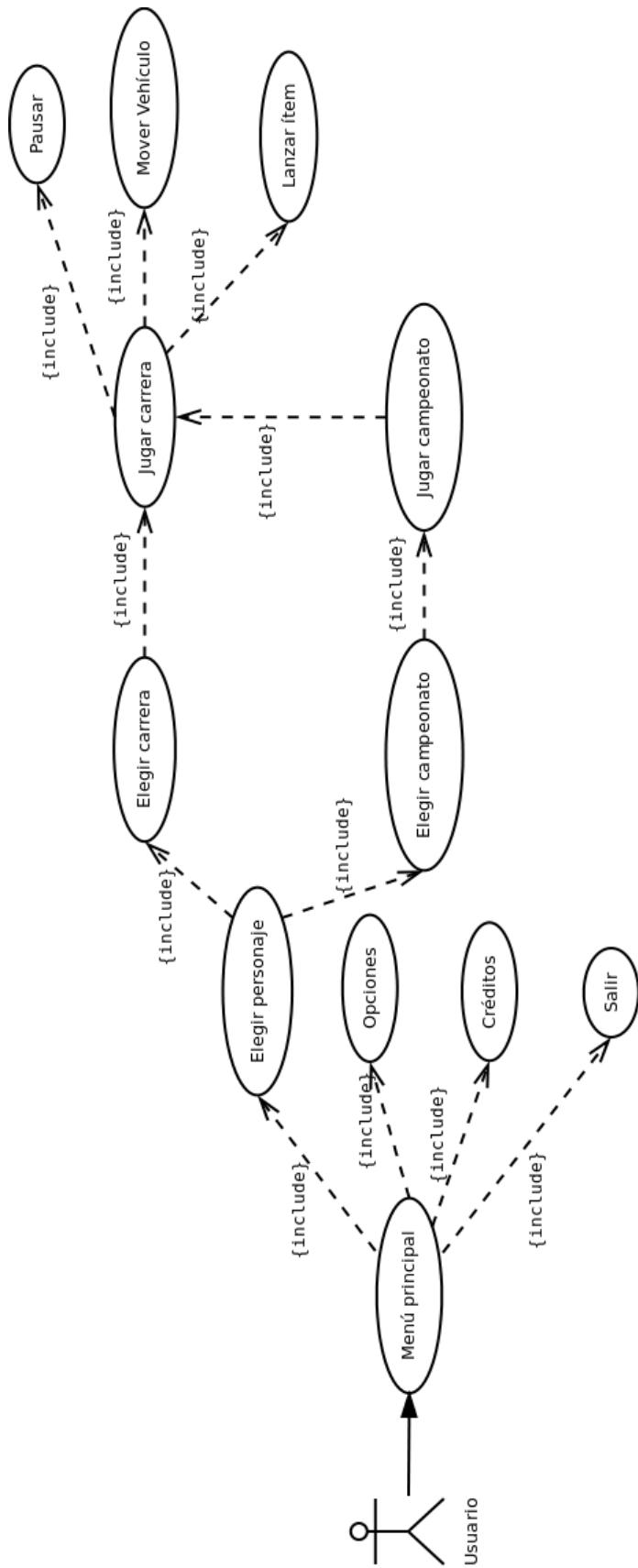


Figura 4.10: Análisis: Diagrama de casos de uso

1. El sistema inicia la opción créditos
- 2c** El usuario selección 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** Ninguna

**Postcondiciones** Se selecciona un jugador y se almacenará en la configuración de juego.

### Escenario principal

1. El usuario desea seleccionar al personaje que usará en el juego.
2. El sistema muestra la pantalla de elección de personaje y carga todos los personajes disponibles.
3. El usuario selecciona un personaje de la lista y pulsa la opción aceptar.
4. El sistema almacena en la configuración el jugador seleccionado.
5. El sistema pasa a selección de circuito

### 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

**5a** El sistema pasa a selección de campeonato.

### 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** y ya ha seleccionado un personaje.

**Postcondiciones** Se selecciona un circuito y se almacena en la configuración del juego.

### Escenario principal

1. El usuario desea seleccionar el circuito en el que competir.
2. El sistema muestra todos los circuitos disponibles (ícono e imagen) para cada campeonato.
3. El usuario selecciona un circuito y pulsa aceptar
4. El sistema almacena el circuito en la configuración del sistema.

5. El sistema accede a la pantalla de jugar 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.

#### **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** y ha elegido un personaje.

**Postcondiciones** Selecciona un campeonato y se almacena en la configuración del juego.

#### **Escenario principal**

1. El usuario desea seleccionar el circuito en el que competir.
2. El sistema muestra los campeonatos disponibles
3. El usuario selecciona un campeonato y pulsa sobre la opción aceptar.
4. El sistema almacena todos los circuitos del campeonato en la configuración del sistema.
5. El sistema inicia el modo jugar campeonato.

#### **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.

#### **Caso de uso: Jugar carrera**

**Caso de uso** Jugar carrera

**Descripción** El usuario juega una carrera

**Actores** Usuario

**Precondiciones** El usuario ha seleccionado un circuito o un campeonato.

**Postcondiciones** El usuario completa una carrera.

#### **Escenario principal**

1. El sistema carga el circuito, el jugador, inteligencia artificial y los ítems.
2. El sistema muestra la pantalla de juego.
3. El usuario y el sistema interactúan durante la carrera.
4. El usuario completa la carrera.

5. El sistema muestra las posiciones finales de la carrera.
6. El usuario pulsa continuar.
7. El sistema pasa al menú principal

#### **Extensiones — flujo alternativo**

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

#### **Caso de uso: Pausar**

**Caso de uso** Pausar

**Descripción** El usuario selecciona pausar el juego y puede reanudarlo, reiniciarlo o volver al menú principal.

**Actores** Usuario

**Precondiciones** Se está jugando una carrera

**Postcondiciones** Ninguna.

#### **Escenario principal**

1. El usuario pulsa el botón de pausa.
2. El sistema detiene todos los elementos del juego y muestra el menú de pausa.
3. El usuario pulsa la opción reanudar.
4. El sistema reanudar 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 reiniciar.

1. El sistema reinicia la carrera.

- \*b El usuario pulsa la opción menú.

1. El sistema vuelve al menú principal.

#### **Caso de uso: Lanzar ítem**

**Caso de uso** Salir

**Descripción** El usuario lanza el ítem que tenga actualmente.

**Actores** Usuario

**Precondiciones** Se esta jugando una carrera y el usuario a recogido un ítem.

**Postcondiciones** Se añade el ítem a la carrera.

#### **Escenario principal**

1. El usuario pulsa la opción de lanzar ítem.
2. El sistema comprueba que el usuario posee un ítem y añade el ítem a la carrera.

#### **Extensiones — flujo alternativo**

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

## **Caso de uso: Mover vehículo**

**Caso de uso** Mover vehículo

**Descripción** El usuario desplaza al vehículo por el circuito.

**Actores** Usuario

**Precondiciones** Se está jugando una carrera.

**Postcondiciones** Ninguna.

### **Escenario principal**

1. El usuario pulsa sobre una de las teclas de movimiento. W o Flecha hacia arriba, para mover el vehículo hacia adelante o S o Flecha hacia abajo, para moverlo marcha atrás.
2. El sistema mueve al vehículo teniendo en cuenta su orientación, velocidad y ángulo.
3. El sistema comprueba que no ha colisionado con ningún obstáculo u otro competidor.

### **Extensiones — flujo alternativo**

- \***a** El usuario cierra la ventana de la aplicación y sale de la aplicación
- 1a** El usuario pulsa las teclas de dirección a la vez que avanza. D o Flecha hacia la derecha, para girar a la derecha o, A o Flecha hacia la izquierda, para girar a la izquierda.
  1. El sistema gira al coche hacia la zona deseada.
- 3a** El sistema detecta que ha colisionado con un competidor o con algún obstáculo.
  1. El sistema corrige la posición del vehículo.

## **Caso de uso: Jugar campeonato**

**Caso de uso** Jugar campeonato

**Descripción** El usuario juega un campeonato.

**Actores** Usuario

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

**Postcondiciones** Se juega un campeonato.

### **Escenario principal**

1. El usuario desea jugar un campeonato
2. El sistema carga el circuito actual
3. El sistema y usuario interactúan en la carrera. Include Jugar carrera.
4. Una vez terminada la carrera el sistema muestra las posiciones del campeonato.
5. El usuario selección la opción continuar.
6. El sistema pasar al siguiente circuito.
7. Volveremos 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 las opciones de juego.
3. El usuario modifica las distintas opciones de juego.
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.

### Caso de uso: Créditos

**Caso de uso** Salir

**Descripción** Se muestran la pantalla de créditos donde se reflejan los creadores del juego.

**Actores** Usuario

**Precondiciones** Ninguna.

**Postcondiciones** Ninguna.

### Escenario principal

1. El sistema muestra la pantalla de créditos.
2. El usuario pulsa la opción volver.
3. El sistema vuelve al menú principal.

### Extensiones — flujo alternativo

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

### **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.

**Menú de pausa** Clase que gestiona el menú de pausa.

**Cursor** Cursor de los menús.

**Botón** Clase que representa el botón en los menús.

**Modo de juego** Clase virtual, con las necesidades básicas de los distintos 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 los coches dirigidos por el ordenador. Aspectos básico 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 ítem** Clase que representa las cajas que proporcionan ítems 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 dirigidos por el ordenador

**Jugador** Clase que representa al vehículo controlado por el jugador.

En las siguientes imágenes podemos ver los diagrama de clases asociado a los requisitos obtenidos. Por razones de espacio en el documento y para una mejor apreciación de las clases necesarias, se ha visto conveniente dividir en dos partes diferenciadas principalmente.

La primera figura muestra el primer diagrama las clases relacionadas con las pantalla de juego, como pueden ser menús y modos de juego.

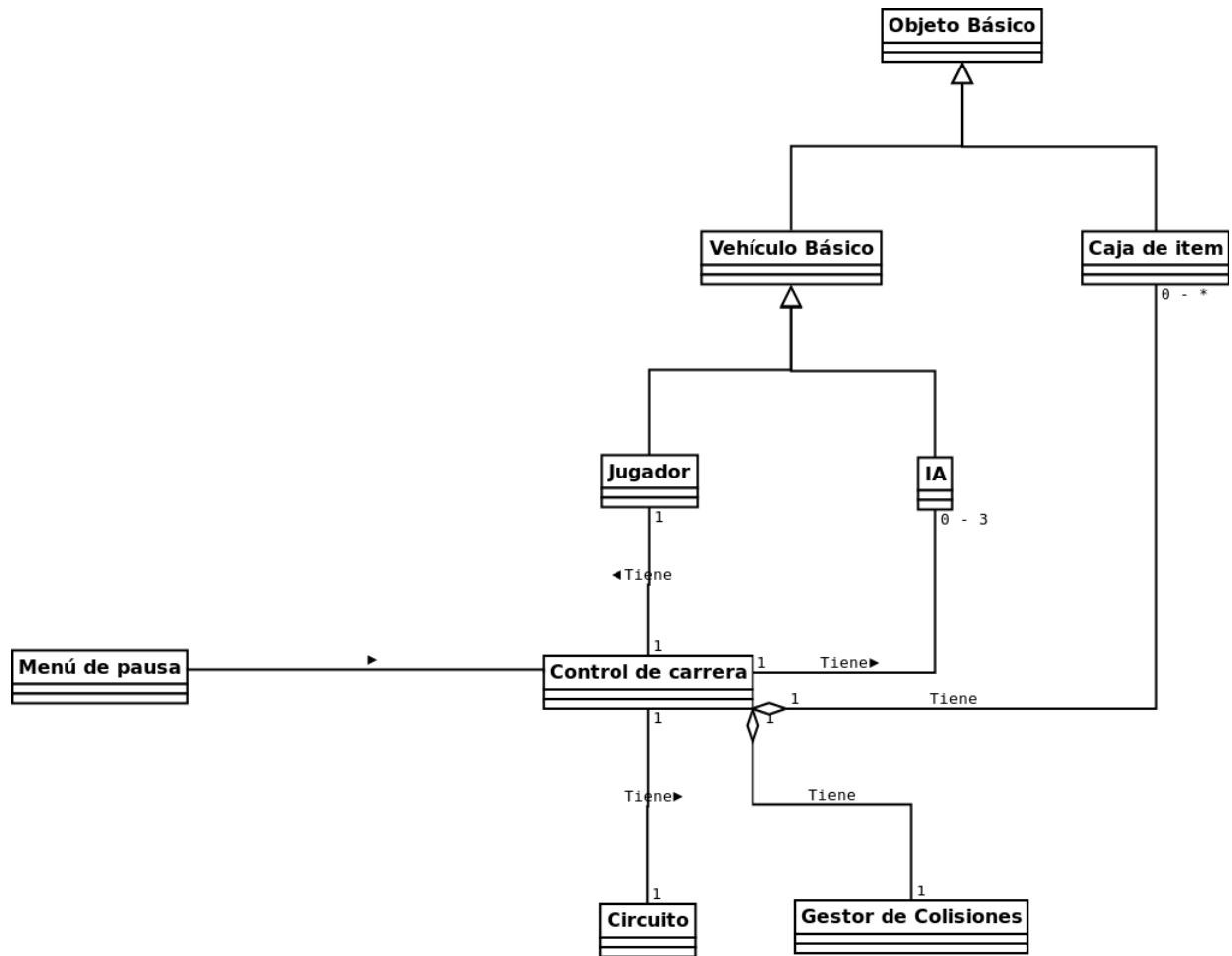


Figura 4.11: Análisis: Diagrama de clases conceptuales 1

En la segunda figura vemos el diagrama de clases referente a las clases principales que intervienen en la gestión de la pantalla de juego.

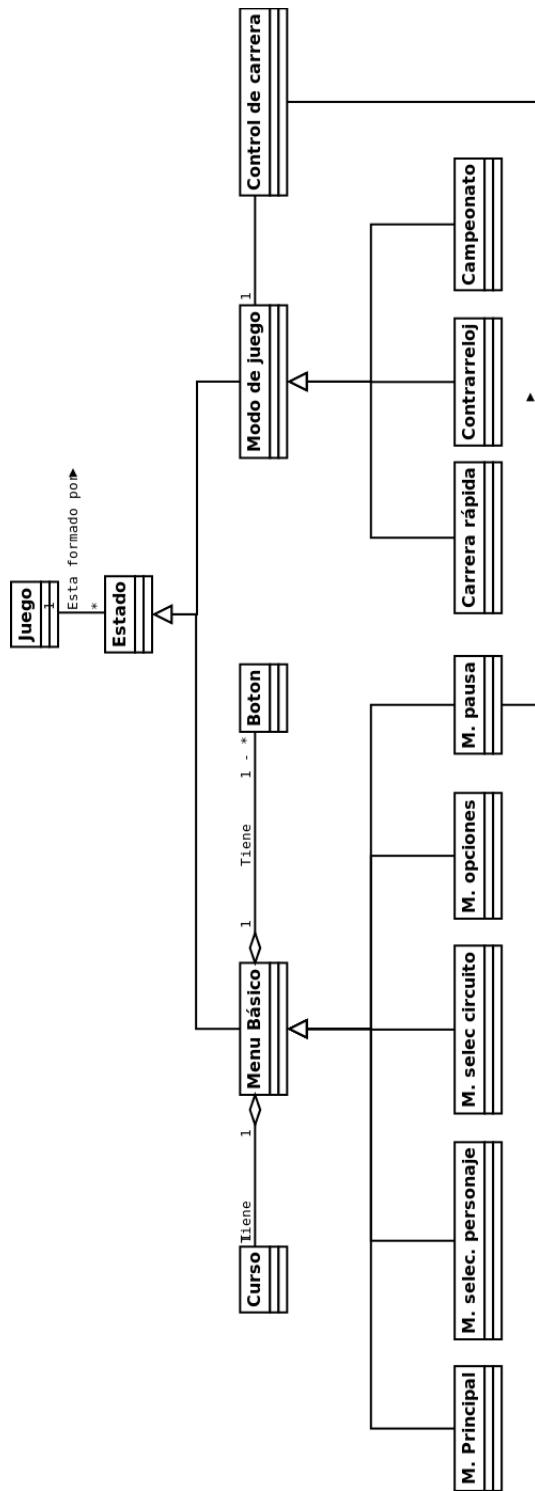


Figura 4.12: Análisis: Diagrama de clases conceptuales 2

#### 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: muestran 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 secuencia y contrato de las operaciones del sistema.

No todos los posibles diagramas de secuencia aparecerán, nos centraremos en los más importantes, los que implican algún tipo de cambio en el sistema.

##### Caso de uso: Menú principal(escenario principal)

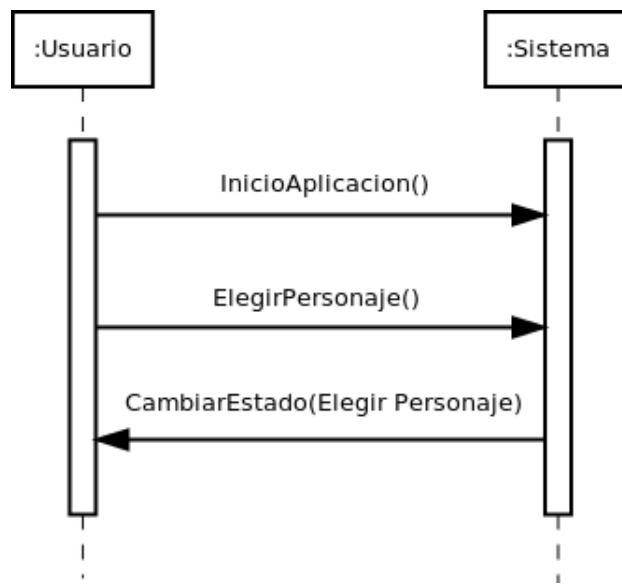


Figura 4.13: Análisis: Diagrama de secuencia Menú principal (escenario principal)

##### Operación InicioAplicacion()

**Actores** Jugador, sistema.

**Responsabilidades** inicia la aplicación y muestra el menú principal.

**Precondiciones** Ninguna

##### Postcondiciones

- El sistema inicia todos los subsistemas necesarios para la correcta ejecución de la aplicación.
- El sistema crea un objeto con el Menú principal

##### Operación ElegirPersonaje()

**Actores** Jugador, Sistema.

**Responsabilidades** salir del menú principal y acceder a la pantalla de elección de personaje.

##### Precondiciones

- Existe un objeto del menú principal.

#### **Postcondiciones**

- Se destruye el objeto del menú principal.

#### **Caso de uso: Menú principal(escenario 2a)**

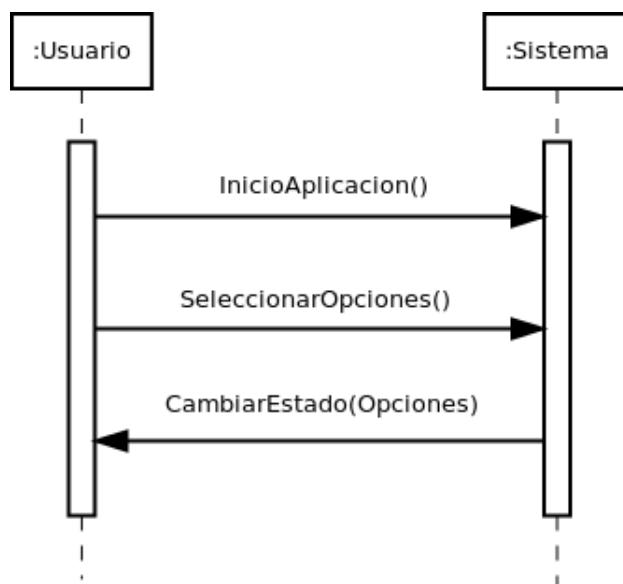


Figura 4.14: Análisis: Diagrama de secuencia Menú principal (escenario 2a)

**Operación** SeleccionarOpciones()

**Actores** Jugador, Sistema

**Responsabilidades** salir del menú principal y entrar en el menú de opciones.

#### **Precondiciones**

- Existe un objeto del menú principal.

#### **Postcondiciones**

- Se destruye el objeto del menú principal.

### Caso de uso: Menú principal(escenario 2b)

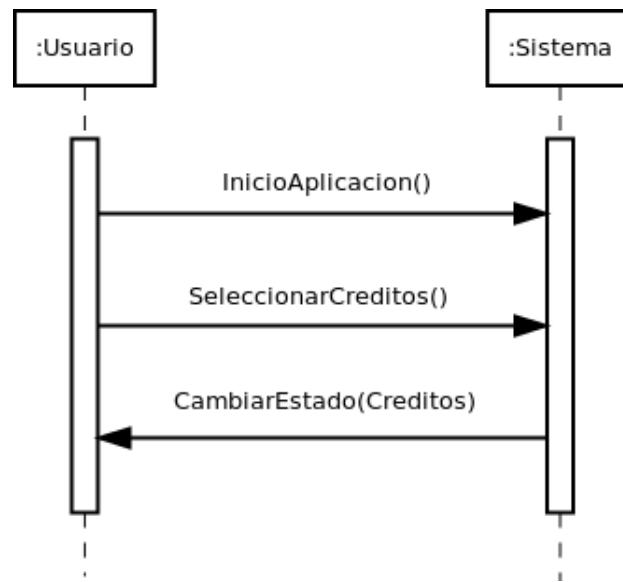


Figura 4.15: Análisis: Diagrama de secuencia Menú principal (escenario 2b)

**Operación** `SeleccionarCréditos()`

**Actores** Jugador, sistema.

**Responsabilidades** salir del menú principal y entrar en la pantalla de créditos.

**Precondiciones**

- Existe un objeto del menú principal.

**Postcondiciones**

- Se destruye el objeto del menú principal.

### Caso de uso: Menú principal(escenario 2c)

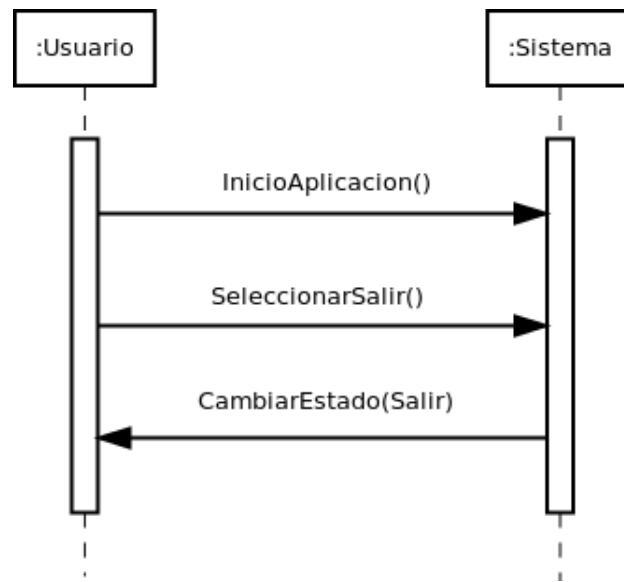


Figura 4.16: Análisis: Diagrama de secuencia Menú principal (escenario 2c)

**Operación** SeleccionarSalir()

**Actores** Jugador, sistema.

**Responsabilidades** Salir de menú principal y salir de la aplicación

**Precondiciones**

- Existe un objeto del menú principal.

**Postcondiciones**

- Se destruye el objeto del menú principal.

## Caso de uso: Elegir personaje (escenario principal)

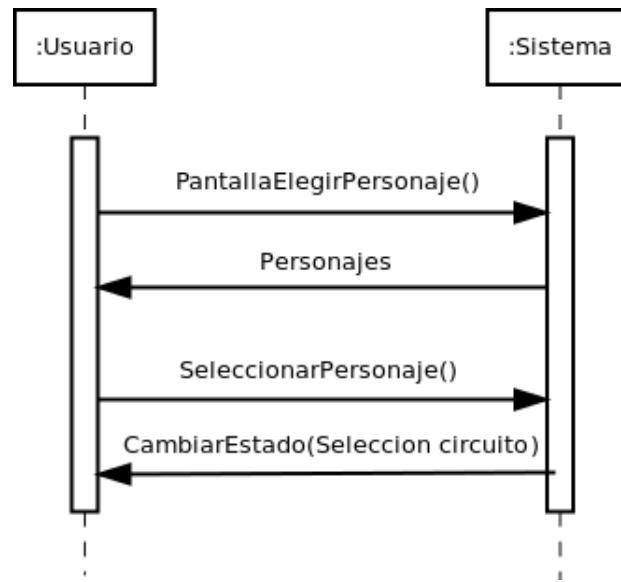


Figura 4.17: Análisis: Diagrama de secuencia Elegir personaje (escenario principal)

**Operación** PantallaElegirPersonaje()

**Actores** Jugador, sistema.

**Responsabilidades** carga y muestra la pantalla de elección de personaje

**Precondiciones** Ninguna

**Postcondiciones**

- Crea un objeto de la clase Menú personaje.

**Operación** SeleccionarPersonaje()

**Actores** Jugador, sistema.

**Responsabilidades** marca un personaje como seleccionado

**Precondiciones**

- Existe un objeto del menú de personaje.
- Existe el personaje seleccionado.

**Postcondiciones**

- Se marca el personaje como seleccionado.
- Se destruye el objeto del menú de personaje.

### Caso de uso: Elegir circuito (escenario principal)

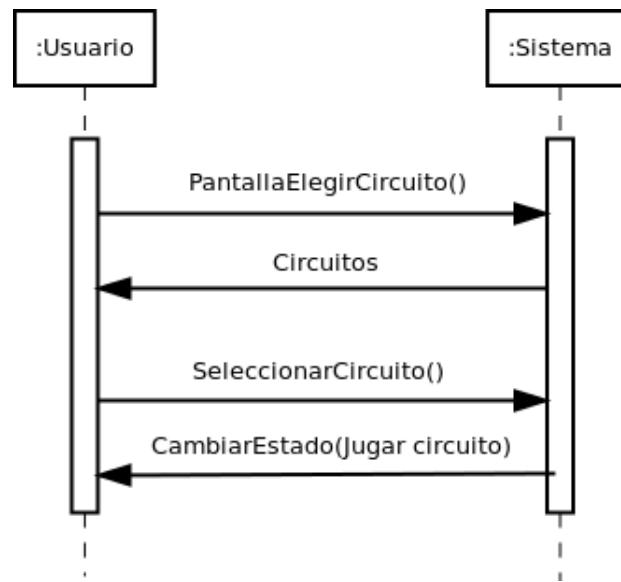


Figura 4.18: Análisis: Diagrama de secuencia Elegir circuito (escenario principal)

**Operación** PantallaElegirCircuito()

**Actores** Jugador, sistema.

**Responsabilidades** crea y muestra la pantalla de elección de circuito.

**Precondiciones** Ninguna.

**Postcondiciones**

- Crea un objeto de la clase menú de circuito.

**Operación** SeleccionarCircuito()

**Actores** Jugador, sistema.

**Responsabilidades** marca un circuito como seleccionado.

**Precondiciones**

- Existe el circuito seleccionado.

**Postcondiciones**

- Se marca el circuito como seleccionado.
- Se destruye el objeto de menú de circuito.

### Caso de uso: Elegir campeonato (escenario principal)

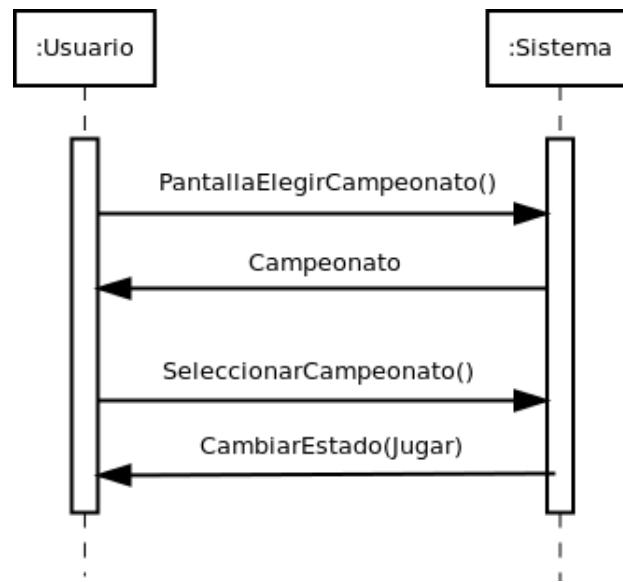


Figura 4.19: Análisis: Diagrama de secuencia Elegir campeonato (escenario principal)

**Operación** PantallaElegirCampeonato()

**Actores** Jugador, sistema.

**Responsabilidades** crea y muestra la pantalla de selección de campeonato

**Precondiciones** Ninguna.

**Postcondiciones**

- Crea un objeto de la clase menú campeonato.

**Operación** SeleccionarCampeonato()

**Actores** Jugador, sistema.

**Responsabilidades** selecciona un campeonato.

**Precondiciones**

- Existe el campeonato seleccionado.

**Postcondiciones**

- Se marca el campeonato seleccionado.
- Se destruye el objeto de menú campeonato.

### Caso de uso: Jugar carrera (escenario principal)

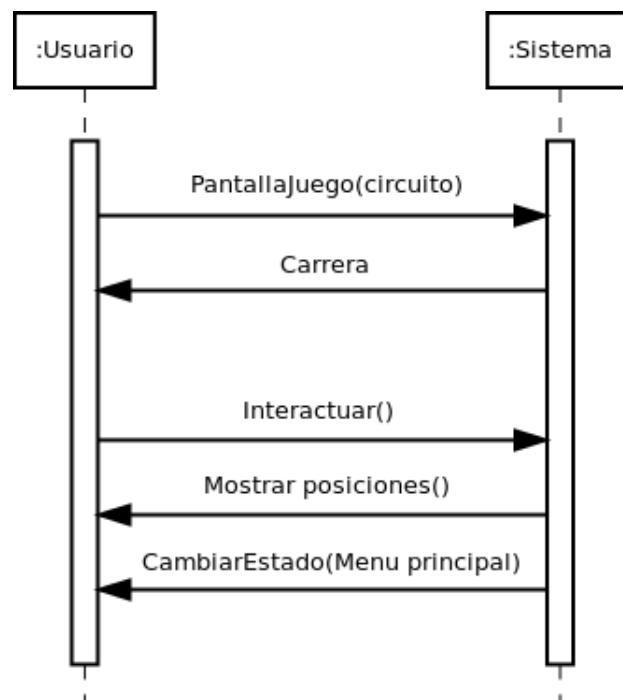


Figura 4.20: Análisis: Diagrama de secuencia Jugar carrera (escenario principal)

**Operación** PantallaJuego()

**Actores** Jugador, sistema.

**Responsabilidades** carga y muestra la pantalla de juego con el circuito seleccionado, inicia el juego.

**Precondiciones** Ninguna

**Postcondiciones**

- Se crea un objeto de Control de juego.
- Se crea un objeto de Circuito
- Se crea un objeto de Jugador.
- Se crea un objeto de Gestor de colisiones.
- Se crea un objeto de IA por cada competidor de carrera.
- Se crea un objeto de Caja de ítem por cada caja del circuito.

**Operación** Interactuar()

**Actores** Jugador, sistema.

**Responsabilidades** permite al jugador interactuar con el mundo 2D y los elementos que este posee.

**Precondiciones**

- Existe un objeto de Control de juego.
- Existe un objeto de Jugador.

**Postcondiciones** Ninguna.

### Caso de uso: Pausar (escenario principal)

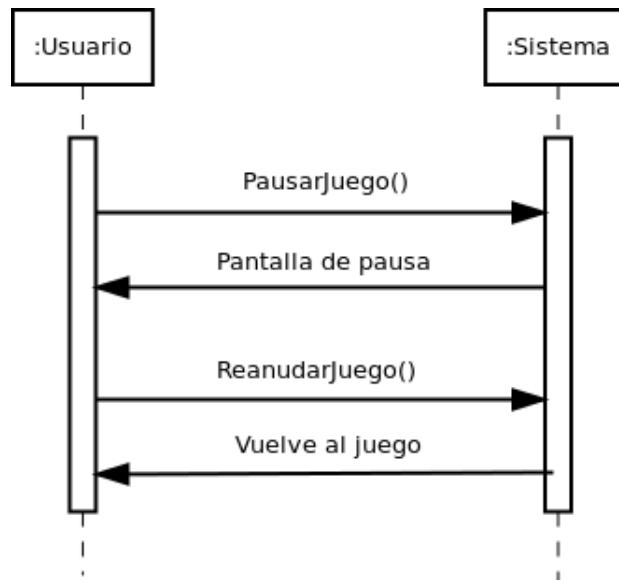


Figura 4.21: Análisis: Diagrama de secuencia Pausar(escenario principal)

#### Operación PausarJuego()

**Actores** Jugador, sistema.

**Responsabilidades** pausa el juego y detiene todos los elementos de este.

#### Precondiciones

- Existe un objeto de Control juego.
- Se está jugando una carrera.

**Postcondiciones** ■ Se crea un objeto de Menú de pausa.

#### Operación ReanudarJuego()

**Actores** Jugador, sistema.

**Responsabilidades** reanuda la partida y quita el menú de pausa.

#### Precondiciones

- Existe un objeto de Control juego.
- La carrera estaba pausada

#### Postcondiciones

- Se destruye el objeto del menú de pausa.

### Caso de uso: Pausar (escenario \*b)

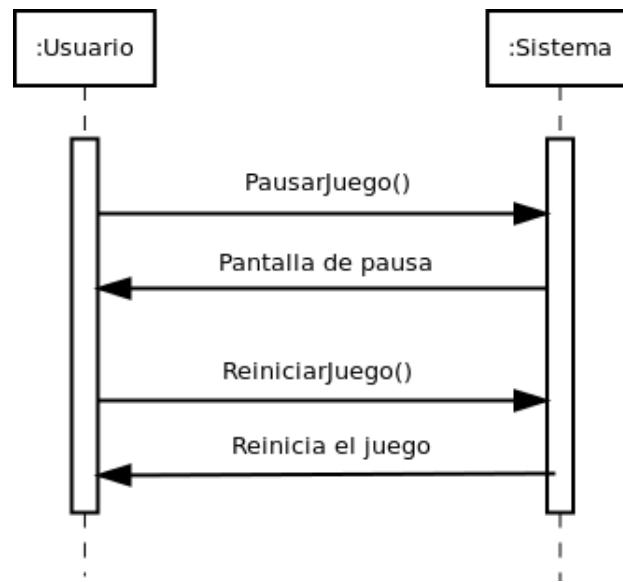


Figura 4.22: Análisis: Diagrama de secuencia Pausar (escenario \*b)

**Operación** ReiniciarJuego()

**Actores** Jugador, sistema.

**Responsabilidades** reinicia la carrera que se estaba jugando.

**Precondiciones**

- Existe un objeto de Control juego.
- La carrera estaba pausada

**Postcondiciones**

- Se destruye el objeto del menú de pausa.
- Se reinicia la carrera.

### Caso de uso: Pausar (escenario \*c)

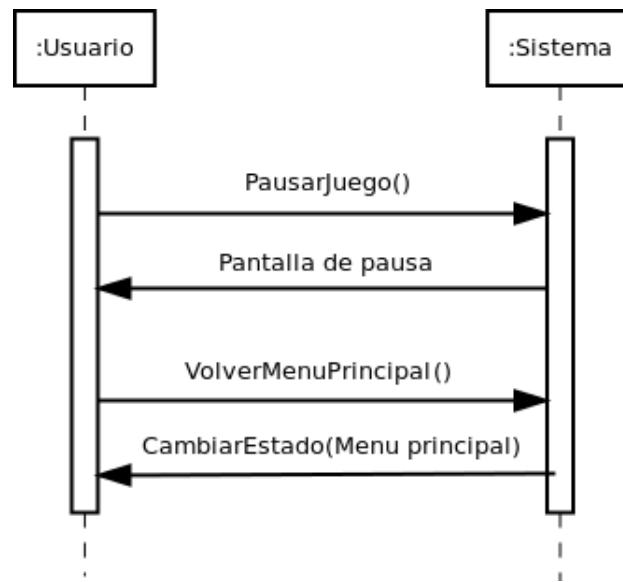


Figura 4.23: Análisis: Diagrama de secuencia (escenario )

#### Operación VolverMenuPrincipal()

**Actores** Jugador, sistema.

**Responsabilidades** se para la carrera jugada y se vuelve al menú principal.

#### Precondiciones

- Existe un objeto de Control juego.
- La carrera estaba pausada

#### Postcondiciones

- Se destruye el objeto del menú de pausa.

### Caso de uso: Lanzar ítem (escenario principal)

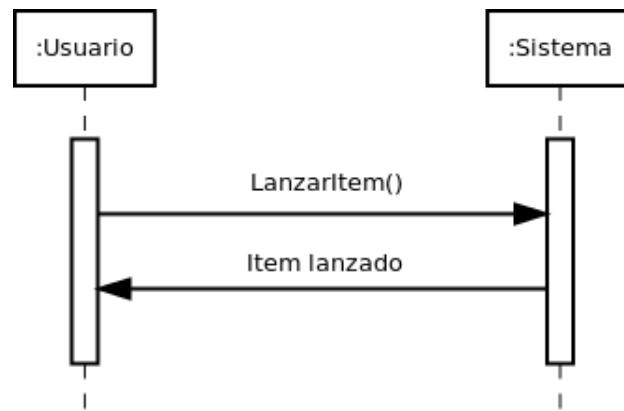


Figura 4.24: Análisis: Diagrama de secuencia Lanzar ítem (escenario principal)

**Operación** LanzarItem()

**Actores** Jugador, sistema.

**Responsabilidades** hace que el jugador lance el ítem que tiene en ese momento.

**Precondiciones**

- Existe un objeto de Control juego.
- Existe un objeto de jugador.
- El jugador posee un ítem.

**Postcondiciones**

- Se crea un objeto ítem.
- Se añade el objeto ítem a Control de juego.

### Caso de uso: Mover vehículo (escenario principal)

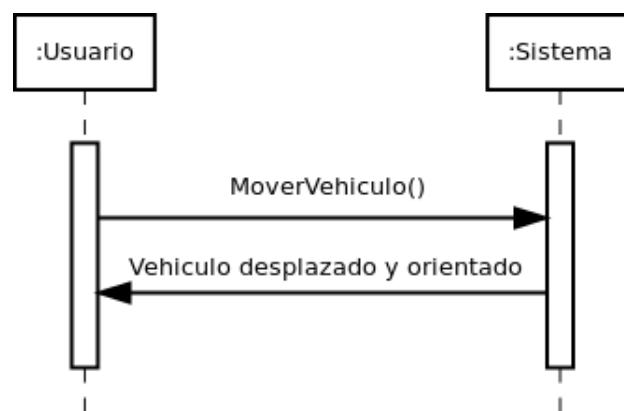


Figura 4.25: Análisis: Diagrama de secuencia Mover vehículo (escenario principal)

**Operación** MoverVehiculo()

**Actores** Jugador, sistema.

**Responsabilidades** mueve al personaje por el mundo 2D.

#### Precondiciones

- Existe un objeto de Control juego.
- Existe un objeto de jugador.

#### Postcondiciones

- Se modifica la posición del objeto jugador.

**Caso de uso: Opciones (escenario principal)**

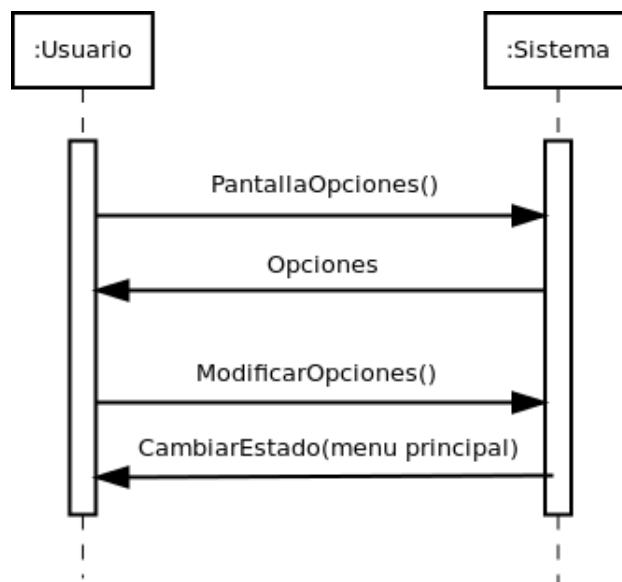


Figura 4.26: Análisis: Diagrama de secuencia opciones (escenario principal)

**Operación** PantallaOpciones()

**Actores** Jugador, sistema.

**Responsabilidades** se crea y muestra la pantalla de opciones.

**Precondiciones** Ninguna.

#### Postcondiciones

- Se crea un objeto de menú de opciones

**Operación** ModificarOpciones

**Actores** Jugador, sistema.

**Responsabilidades** se modifican las opciones y se vuelve al menú principal.

### Precondiciones

- Existe un objeto de menú de opciones

### Postcondiciones

- Se aplican los cambios de opciones.
- Se destruye el objeto de menú de opciones.

### Caso de uso: Créditos (escenario principal)

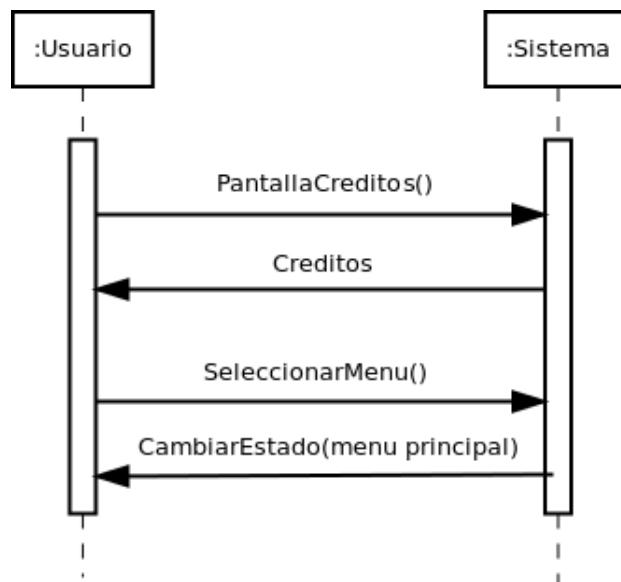


Figura 4.27: Análisis: Diagrama de secuencia Créditos(escenario principal)

#### Operación PantallaCréditos()

Actores Jugador, sistema.

Responsabilidades crea y muestra la pantalla de créditos

Precondiciones Ninguna.

#### Postcondiciones

- Se crea un objeto de Menú de créditos.

#### Operación SeleccionarMenu()

Actores Jugador, sistema.

Responsabilidades destruye la pantalla de créditos y vuelve el menú principal.

#### Precondiciones

- Existe un objeto de Menú de créditos.

#### Postcondiciones

- Se destruye el objeto de Menú de créditos.



# 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 más 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 erróneos 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 gráficas que se han desarrollado para la aplicación.

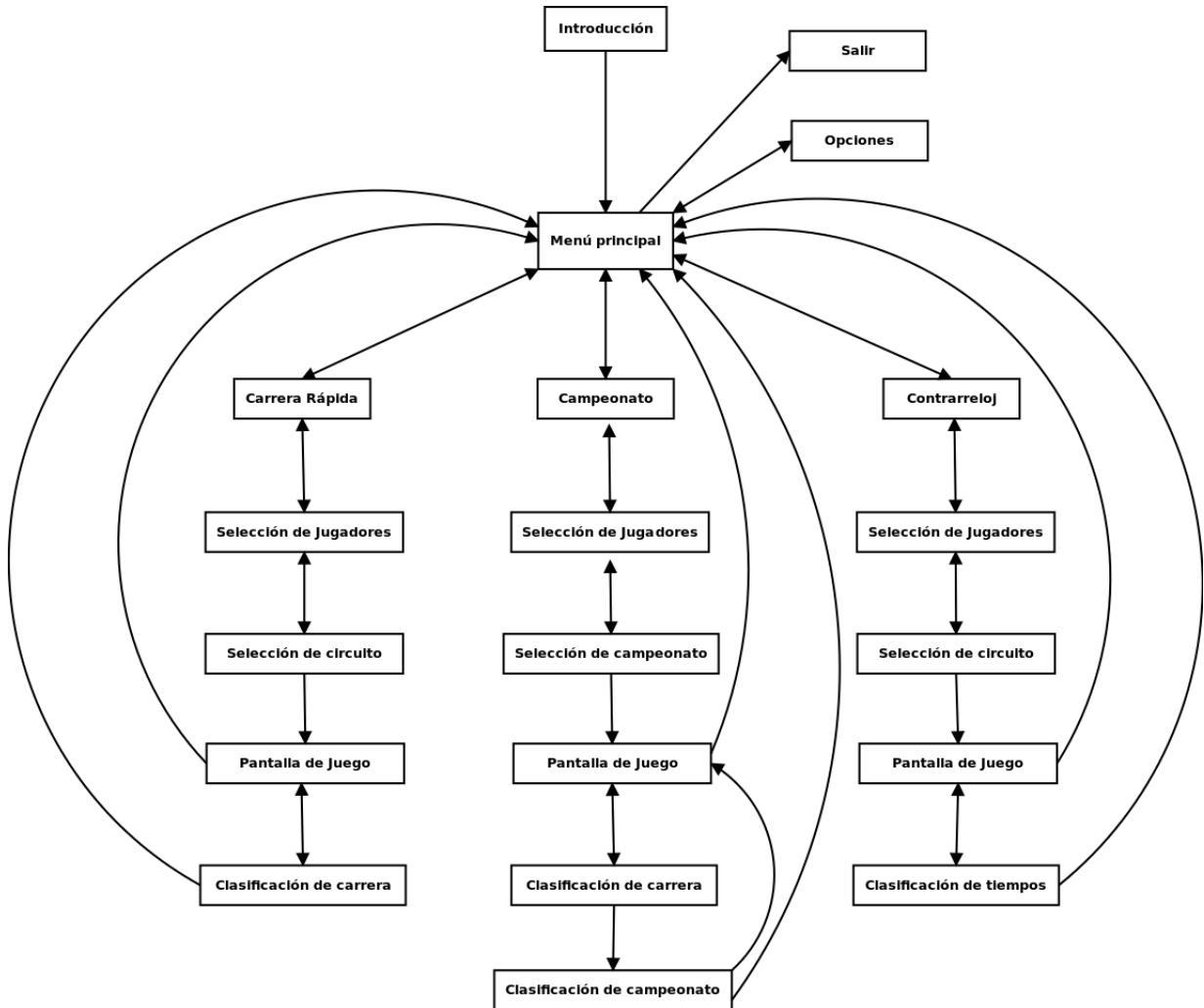


Figura 5.2: Diseño: Diagrama de interacción

## 5.2. Diagrama de clases de diseño

Dividiremos el diagrama de clases de diseño en dos partes, para verlo con una mayor claridad, dicha separación se basará en el cometido de las clases que interfieran.

En primer lugar mostraremos todas las clases, encargada de inicializar el juego, y la gestión de los distintos estados que podemos encontrar en este. Como pueden ser todos los menús en los que podremos interactuar en el sistema, mostrando la jerarquía de los mismos. Y también la clases encargadas de gestionar los tres modos de juego existentes, carrera rápida, contrarreloj y campeonato.

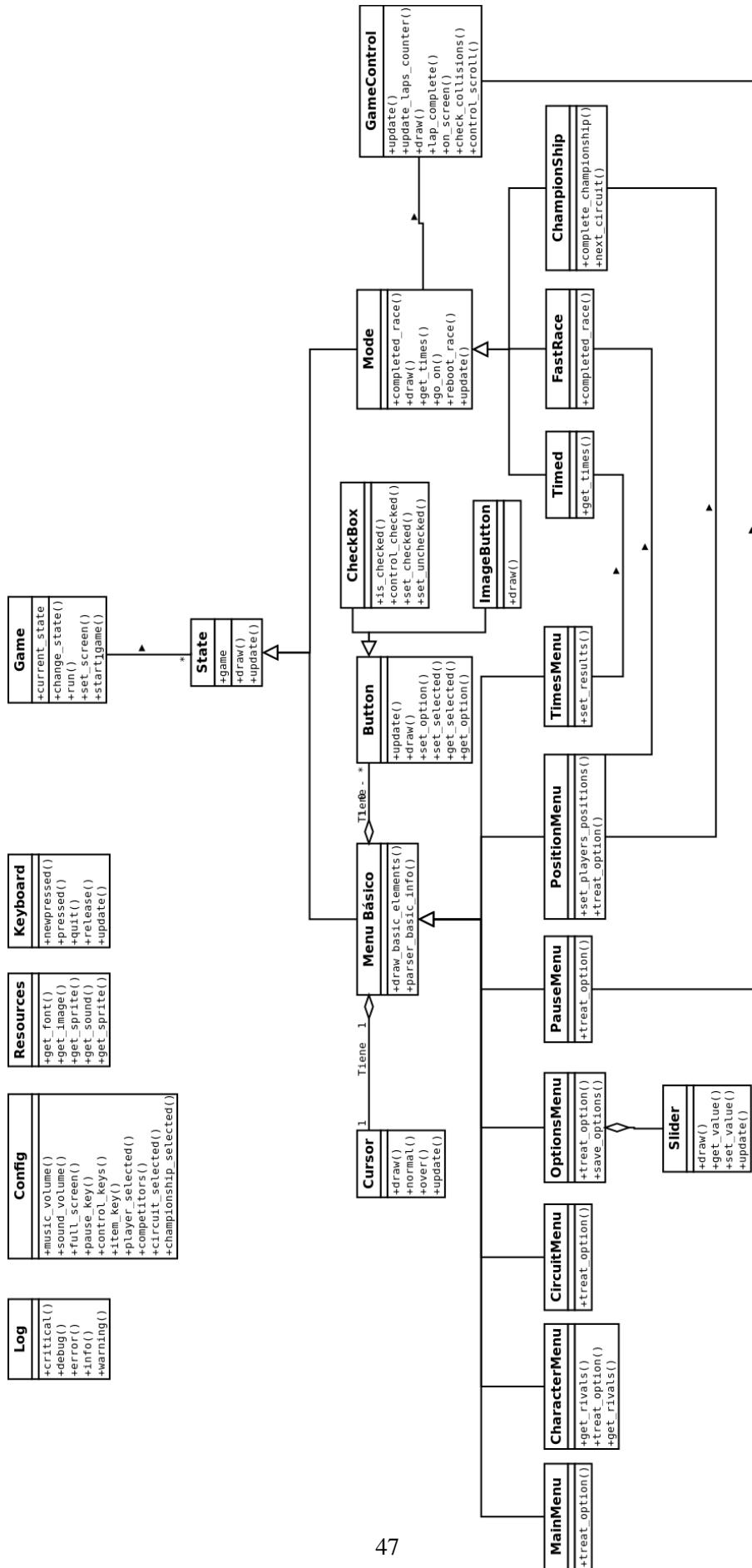


Figura 5.3: Diseño: Diagrama de clases de diseño 1

En el segundo diagrama mostraremos todas las clases relacionadas con el sistema de juego. Podremos observar los distintos objetos que intervienen en el juego, así como la jerarquía de estos. Y también todas las clases necesarias para el correcto funcionamiento del juego.

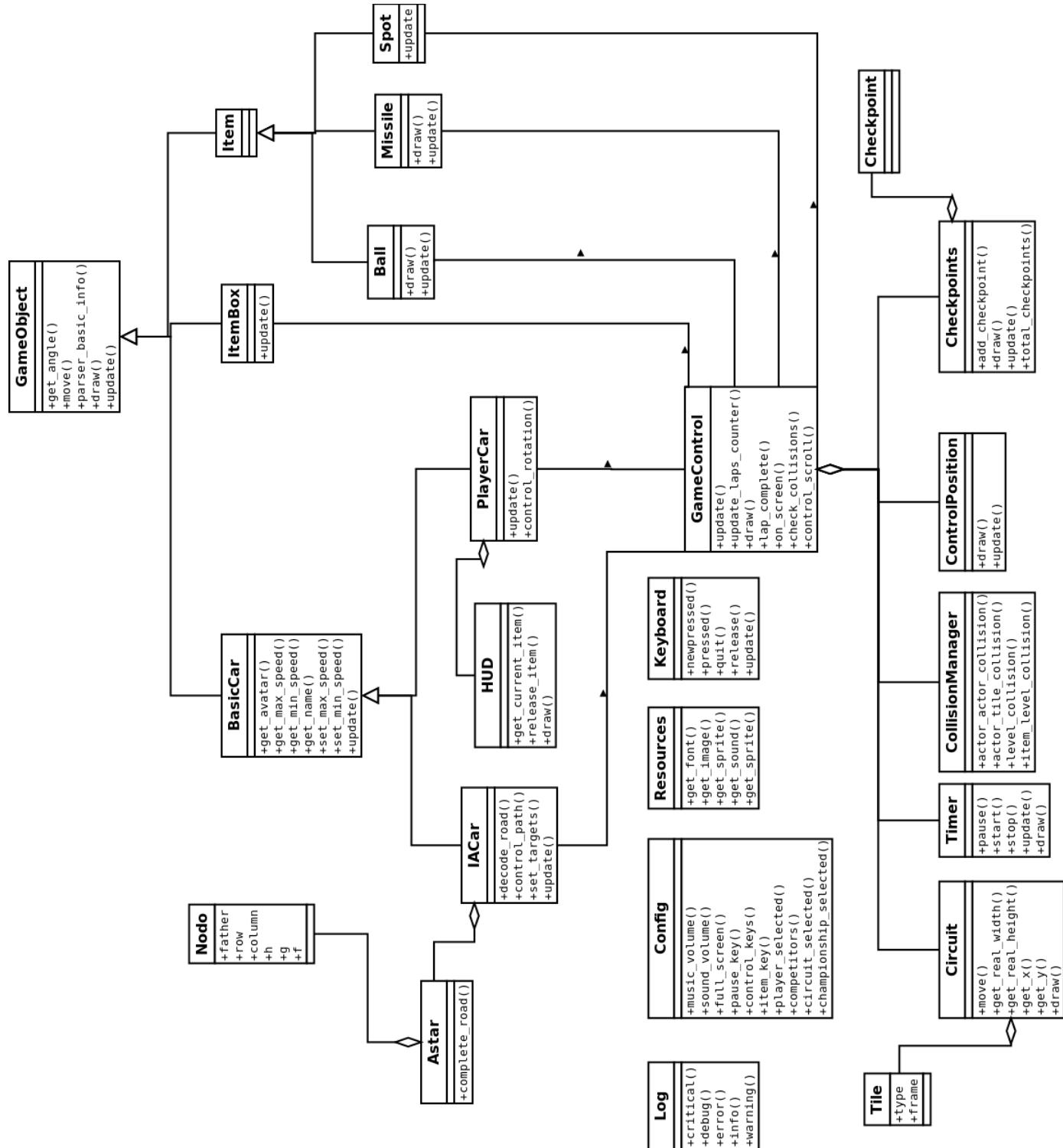


Figura 5.4: Diseño: Diagrama de clases de diseño 2

En ambos diagramas de clases de pueden apreciar un conjuntos de clases que no están relacionadas con ninguna otra del diagrama. Estas clases a las que nos referimos siguen un patrón Singleton, que permite que cualquier clase pueda acceder a ella desde cualquier parte del sistema. Sólo existirá una única instancia de cada una de las clases Singleton, que se creará en la primera llamada que se realice a ellas.



# 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.

También esta disponible toda la documentación del proyecto generada con doxygen. Se puede consultar en el siguiente enlace:

<http://zycars.googlecode.com/svn/doc/doxygen/html/index.html>

### 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 <sup>1</sup>.

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:

```
1 <mainmenu background="background_menu" cursor="cursor.xml"
2   music='la maryjane - bob wizman.ogg'
3     <title text="Menu Principal" font="cheesebu" size="30" r="0" g="0" b="0"
4       "
5         x="15" y="15"/>
6         <images>
7           <image image_code="separator_line" x="0" y="240"></image>
```

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

```

7     <image image_code="logo_menu" x="130" y="5"></image>
8 </images>
9 <options>
10    <option xml_file="menu/mainoption2.xml" font="cheesebu" text=""
11        Carrera Rapida"
12    center="True" x="400" y="295"></option>
13    <option xml_file="menu/mainoption1.xml" font="cheesebu" text=""
14        Campeonato"
15    center="True" x="400" y="335"></option>
16    <option xml_file="menu/mainoption2.xml" font="cheesebu" text=""
17        Contrarreloj"
18    center="True" x="400" y="400"></option>
19    <option xml_file="menu/mainoption1.xml" font="cheesebu" text=""
20        Opciones"
21    center="True" x="400" y="440"></option>
22    <option xml_file="menu/mainoption2.xml" font="cheesebu" text=""
23        Creditos"
24    center="True" x="400" y="505"></option>
25    <option xml_file="menu/mainoption1.xml" font="cheesebu" text="Salir
26        "
27    center="True" x="400" y="545"></option>
28 </options>
29 </mainmenu>

```

Como se puede 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:

```

1 <car name_character="Camaro Bun" sprite_code="yellow" max_speed="6.1"
2 min_speed="3" acceleration="0.5" desacceleration="0.08" rotation_angle="0.35"
3 avatar="avatar1" racer_image='car1'>
4 <!--NORMAL, NOACTION, RUN, FORWARD, REVERSE, DAMAGED, ERASE, YAW-->
5   <animations>
6     <animation name="normal" frames="0" delay="1"/>
7     <animation name="noaction" frames="0" delay="1"/>
8     <animation name="foward" frames="0" delay="1"/>
9     <animation name="run" frames="0" delay="1"/>
10    <animation name="reverse" frames="0" delay="1"/>
11    <animation name="damaged" frames="0" delay="1"/>
12    <animation name="erase" frames="0" delay="1"/>
13    <animation name="yaw" frames="0" delay="1"/>
14    <animation name="fall" frames="0" delay="1"/>
15    <animation name="turbo" frames="0" delay="1"/>
16   </animations>
17 </car>

```

Como se puede observar, se pueden modificar los aspectos más básico del comportamiento de los personajes, como pueden ser la velocidad máxima de este, velocidad en marcha atrás, aceleración, des-aceleración y ángulo 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, indicáramos 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*<sup>2</sup>, 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, dicha imagen sería demasiado grande y ocuparía bastante memoria. Mientras que con esta opción tendríamos una imagen pequeña con el conjunto necesario de tiles para el circuito en cuestión.

El formato de los XML generados por el programa son de la siguiente forma:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <map version="1.0" orientation="orthogonal" width="88" height="74"
   tilewidth="45" tileheight="45">
3  <properties>
4    <property name="ancho_meta" value="5"/>
5    <property name="ancho_pista" value="5"/>
6    <property name="collision_map" value="collisionmapgarden.png"/>
7    <property name="grado_coche" value="270"/>
8    <property name="tileset_alto" value="35"/>
9    <property name="tileset_ancho" value="21"/>
10   </properties>
11   <tileset firstgid="1" name="tilesetgarden" tilewidth="45" tileheight="45">
12     <image source="tilesetgarden.png" width="945" height="1575"/>
13   </tileset>
14   <layer name="Capa 1" width="88" height="74">
15     <data>
16       <tile gid="1"/>
17       <tile gid="2"/>
```

<sup>2</sup>Se comenta su uso y características en el apéndice relacionado con las herramientas utilizadas

```

18      ....
19    </data>
20  </layer>
21  <layer name="Capa 2" width="88" height="74">
22    <data>
23      <tile gid="0"/>
24      <tile gid="0"/>
25      ...
26  </layer>
27  <layer name="Capa 3" width="88" height="74">
28    <data>
29      <tile gid="74"/>
30      <tile gid="75"/>
31      ...
32    </data>
33  </layer>
34  <objectgroup name="checkpoints" width="88" height="74" opacity="0.2">
35    <object name="1" type="Horizontal" x="180" y="1035" width="45" height="45"
36      "/>
37    <object name="1" type="check" x="450" y="225" width="45" height="45"/>
38    <object name="2" type="check" x="2205" y="405" width="45" height="45"/>
39  </objectgroup>
</map>

```

Una de las únicas cosas que no proporcionaba el programa era poder indicar cuales 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 cargáramos el circuito que necesitáramos en ese momento y con ello el conjunto de tiles relacionado, se comprobaría que 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

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

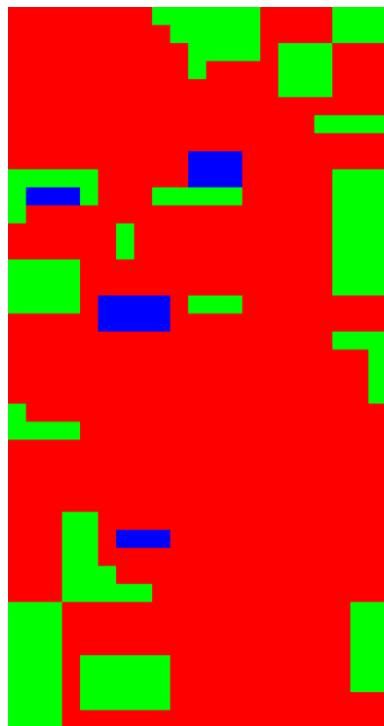


Figura 6.2: Implementación: Mapa de colisiones

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 ralentizan 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 ítem lanzado por otro jugador.

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

En el siguiente apartado se explicará cuales 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 asociado una imagen que indica de que 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 esta 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 colisiona del tile:



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

En el caso de que se colisione con un tile que no es atravesable pero es del tipo que ralentizaban 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 un rebote en función de la velocidad que llevara en ese momento.
- **Colisión vehículo-ítem:** en este caso se responderá a la colisión dependiendo del tipo de ítem con el que hemos colisionado.
  - Si el ítem es un misil o una bola, el ítem pasará a su estado de explosión, mientras que el vehículo pasará a un estado de daño
  - Si el ítem es una mancha de aceite, el ítem no cambiará su estado, pero el coche pasar a un estado de descontrol durante unos instantes
  - Si el ítem es un chicle, se reducirá de forma considerable la velocidad del vehículo.

## 6.4. Inteligencia artificial

Otro de los aspectos más importante 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 el ordenador.

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 ítems:** también debe poder tirar los ítems que reciba de las bolas de ítems.

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 vehículos 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 diferencias:

- 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 lleguemos 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 punto, se hará una nueva consulta al A\* para obtener el camino al próximo y así sucesivamente.

#### **6.4.2. Lanzamiento de ítems.**

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

Por ejemplo, cuando obtenga un misil o una bola deberá lanzarlos en el momento que tenga algún oponente delante y a tiro. Pero si en cambio hemos obtenido como ítem 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 los coches dirigidos por el ordenador

De forma que en el momento que tengamos un ítem 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 ítem que se lanza por la parte trasera, haríamos la misma comparación pero con la barra trasera. En el momento que tenga lugar una de las colisiones, el personaje lanzaría automáticamente el ítem disponible.

# **Capítulo 7**

## **Pruebas**

El diseño de casos de pruebas para un videojuego es algo complicado, no sólo para *Zycars*, si no para la mayoría de los distintos tipos de videojuegos existentes. Esto se debe a que estamos en un escenario simulando como interactúan muchísimos elementos entre sí. Pero las pruebas son algo necesario si deseamos un software con una calidad aceptable.

Todos los módulos que componen la aplicación han sido probados individualmente, como pueden ser aquellos módulos encargados de la gestión de colisiones, la inteligencia artificial, gestión de las carreras.

También se realizaron pruebas de integración, ya que había módulos, una vez probado en solitario, debían realizar distintas acciones junto con otros módulos. Como puede ser la búsqueda A\* o el módulo de configuración.

Otras pruebas que se realizaron fueron las de jugabilidad, tanto yo como personas ajenas al desarrollo de proyecto probaron el mismo ofreciendo sus opiniones sobre aspectos que deberían ser modificados o errores que aparecían a lo largo de la ejecución del juego.

Se hicieron pruebas sobre la interfaz a medida que se implementaban nuevos menús de juego, donde se probaban la reacción de esto ante situaciones para las que no estaban pensado su uso.

En definitiva, la organización de los casos de pruebas fue la siguiente:

1. Tras finalizar la implementación de cada módulo, se realizaban pruebas unitarias sobre estos.
2. A medida que distintos módulo que anteriormente probados individualmente, debían colaborar entre ellos, se llevaban a cabo pruebas de integración.
3. Con las distintas versiones jugables se realizaban pruebas de jugabilidad.
4. Pruebas de interfaz sobre los distintos menús que se implementaban.

### **7.1. Pruebas unitarias**

Estas pruebas se realizaron junto a la fase de implementación, conforme se implementaban nuevos módulos necesarios para la aplicación se realizaban pruebas individuales sobre estos módulos. De esta forma se buscaban todos los caminos posibles que podría dar cada módulo, teniendo en cuenta aquellos que fueran más predispuesto a fallos.

De esta forma todas las sentencias se ejecutaban como mínimo una vez y los posibles fallos se encontraban de una forma más sencilla. Por lo que también eran más fácil localizar donde estaba el problema y afrontar la solución de este.

### 7.1.1. Análisis del código con Pylint.

Una vez que se comprobaba el correcto funcionamiento del módulo implementado, se le pasaba el analizador de código pylint, con el fin de que el código siguiera un estándar uniforme y que estuviera exento de cualquier tipo de errores o signos de mala calidad.

Pylint proporciona una nota de 1 a 10 al código analizado. Se intentó que todos y cada uno de los módulos tuvieran al menos una nota mínima de 7. Finalmente se obtuvo una nota de 8.25 en todo el código conjunto de la aplicación.

## 7.2. Pruebas de integración

Conforme aparecían nuevos módulos, cuya implementación era necesaria y a su vez estos requerían el uso de otro módulos que posteriormente habían sido probados individualmente, se realizaban pruebas de integración entre dichos módulos.

Conforme se avanzaba en el desarrollo de *Zycars* se realizaban pruebas de integración a mayor escala. No solo entre módulos del mismo sistema, sino entre varios sistemas del juego.

En este apartado los principales problemas se encontraron a la hora de integrar todas las pantallas del juego, como pueden ser todos los menús, los modos de juego y la pantalla de juego en sí. Pero la resolución de los errores que aparecían no tuvieron gran dificultad.

## 7.3. Pruebas de jugabilidad

Cada vez que estaban disponibles nuevas versiones jugables del juego, se pedían la ayuda a personas, totalmente ajena al desarrollo de la aplicación, que probaran las distintas demos disponibles. De esta forma cada uno de los colaboradores daba su opinión sobre distintos aspectos como la jugabilidad, dificultad, respuesta del juego o nivel de la inteligencia artificial. Tras recopilar la información que todos ellos proporcionaron, se procedía a realizar los ajustes necesarios a los distintos parámetros requeridos.

Entre los distintos aspectos a probar que se le recomendaban a los colaboradores que hicieran especial hincapié, son los siguiente:

- **Colisiones con escenario:** se les pedía que comprobarán que era imposible atravesar los objetos colisionables que se pueden encontrar a lo largo de los circuitos.
- **Colisiones con otros competidores:** comprobación de las colisiones con los otros competidores de la carrera y la reacción de los coches.
- **Control de carreras:** correcto funcionamiento del sistema de control de carrera, que se controlaran correctamente las posiciones de los personajes, el control de las vuelta y la consecución correcta de esta.
- **Inteligencia artificial:** los coches controlados inteligencia artificial hacen correctamente el recorriendo y lanzan ítems cuando lo ven oportuno.

- **Modo carrera rápida:** la carrera se completa correctamente y muestra la posición de los distintos jugadores tal y como terminó la carrera.
- **Modo contrarreloj:** Los tiempos se controlan bien y al concluir la carrera se muestra los tiempos obtenido y si se ha superado algún tiempo.
- **Modo campeonato:** se pasan correctamente de una carrera a otra y el control de los puntos acumulados por carrera se realiza correctamente.

La mayor parte de las personas que probaron el juego, opinaron que este tenía la dificultad necesaria para que el juego fuera un reto competir contra coches dirigidos por el ordenador, pero no hasta el punto de que fuera imposible vencer a esta. Aun así aconsejaron adaptar las velocidad de los distintos vehículos disponibles, para que la diferencia entre estos fuera menor y la posibilidad de adelantamientos fuera mayor. Tras realizar los cambios propuesto, los colaboradores se mostraron más contentos con los resultados.

## 7.4. Pruebas de interfaz

Conforme se realizaban las distintas pantallas referentes a los menús que podemos encontrar en el juego, se realizaban pruebas exhaustivas sobre estas. Sobre todo se comprobaban que las interfaces fueran intuitivas y claras. Así como probar a cambiar valores erróneos de elementos que podríamos encontrar en el menú de opciones o a la hora de seleccionar las vueltas de carrera.

Se hizo especial hincapié en las siguientes pantallas:

- **Menú principal:** comprobar si se accedía correctamente a las distintas opciones disponibles.
- **Menú de opciones:** se modificaban las opciones sin problemas y sin poder añadir valor errores, tras aceptar los cambios, estos se aplicaban correctamente.
- **Menú de selección de personaje:** se elegía el personaje deseado correctamente.
- **Menú de selección de circuito:** se elegía correctamente el circuito deseado y la modificación del número de vueltas se hacia correctamente.

Se comprobó que la interfaz era bastante sólida y no era necesario ningún manual de usuario para poder navegar sobre ella. El sistema de opciones no admite valores erróneos.



# **Capítulo 8**

## **Conclusiones**

En esta sección se comentarán las distintas conclusiones que se han obtenido tras la finalización del proyecto *Zycars*.

### **8.1. Resumen de objetivos**

En primer lugar comentar que es el primer proyecto de estas características al que me enfrento en solitario. Es evidente que su realización no me ha dejado indiferente. No ha sido fácil construir una idea clara sobre lo que se quería hacer. Así como solucionar los distintos problemas que han ido apareciendo a lo largo del desarrollo de este.

También decir que el proyecto me ha ocupado bastante más tiempo del esperado en un principio. Tuve muchos problemas y alguna que otra duda en algunas fases del desarrollo de proyecto, que me tuvieron bloqueado durante un tiempo hasta encontrar la solución más adecuada para estos. A pesar de todo, estoy muy satisfecho con el resultado que se ha obtenido.

Se puede decir que el proyecto goza de buena calidad. Se ha intentado hacer un software sencillo, intuitivo, fácil de manejar y entretenido para el jugador. Algo esencial para un juego de estas características, en el que se busca que cualquier persona pueda echar algún rato de su tiempo libre y qué menos que disfrute durante ese tiempo.

### **8.2. Conclusiones personales**

Durante el desarrollo del proyecto se han aprendido muchísimas cosas: como hacer distintas ramas de desarrollo, plantear y crear calendarios, usar las herramientas adecuadas, hacer decisiones importantes para el desarrollo de este, documentación del código, organización, etc. Ya que durante la carrera se han realizado distintas prácticas y trabajos de complejidad, pero nada con el tamaño y duración que requiere un Proyecto de fin de carrera. Una vez finalizado este creo que tengo la experiencia necesaria para afrontar otro proyecto con buenos resultados.

Entre las distintas herramientas, *L<sup>A</sup>T<sub>E</sub>X* es una de las que he aumentado mis conocimientos durante la realización de la memoria y gracias al compañero Pablo Recio por la plantilla facilitada para la realización de la memoria del proyecto, que sin duda ha evitado muchos problemas.

Puedo decir que he aprendido un nuevo lenguaje de programación, como es *Python*, ya que, que mejor forma de aprender un nuevo lenguaje, que realizar un proyecto con este.

He aprendido a usar con bastante soltura la biblioteca *Pygame*, gracias tanto a la documentación de la página oficial, como a la traducción disponible en Loserjuegos.

En definitiva, este proyecto me ha hecho madurar como persona y estudiante. He aprendido a buscar bibliografía, opiniones en otras personas, compartir ideas, seguir un horario, cumplir una fechas de entrega y enfrentarme a un proyecto de estas características.

### 8.3. Mejoras y ampliaciones

Las posibles mejoras y ampliaciones que se podrían añadir al proyecto en futuras versiones, se comentan a continuación:

- **Modo de dos jugadores:** añadir un nuevo modo de juego que nos permitiera jugar contra otra persona en el mismo ordenador. De forma que la pantalla quedaría dividida en dos.
- **Modo en red:** también sería una buena idea añadir un modo de juego en el que pudiésemos jugar en red contra otros oponentes. Este modo sería más conveniente que el modo de dos jugadores, ya que dos personas jugando en un mismo ordenador puede llegar a ser incomodo.
- **Soporte para varias resoluciones:** añadir soporte para varias resoluciones sería algo muy cómodo para aquellas personas con pantalla muy pequeñas, como pueden ser los usuarios de netbooks, o también para persona con grandes resoluciones que desean una ventana de juego mayor.
- **Grabación de las mejores vueltas para repetirlas:** implementar una opción que grabara la vuelta más rápida de cada uno de los circuitos, almacenándolas en un fichero, y así poder visualizarlas posteriormente.

## 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 más 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 estándar 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 usaríamos 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 estándar 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 estándar uniforme y que estuviera exento de cualquier tipo de errores o signos de mala calidad. Para ello se usó la herramienta 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 mercurial<sup>1</sup>, pero finalmente me decanté por subversión, ya que en el proyecto yo era el único desarrollador y quizás, los otros dos sistema están pensados para proyectos con un número de más altos desarrolladores.

### 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.

---

<sup>1</sup>Google 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<sup>A</sup>T<sub>E</sub>X. 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<sup>A</sup>T<sub>E</sub>X

L<sup>A</sup>T<sub>E</sub>X es un sistema de composición de textos que está formado mayoritariamente por órdenes (macros) construidas a partir de comandos de TeX. L<sup>A</sup>T<sub>E</sub>X 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 aparecen 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.

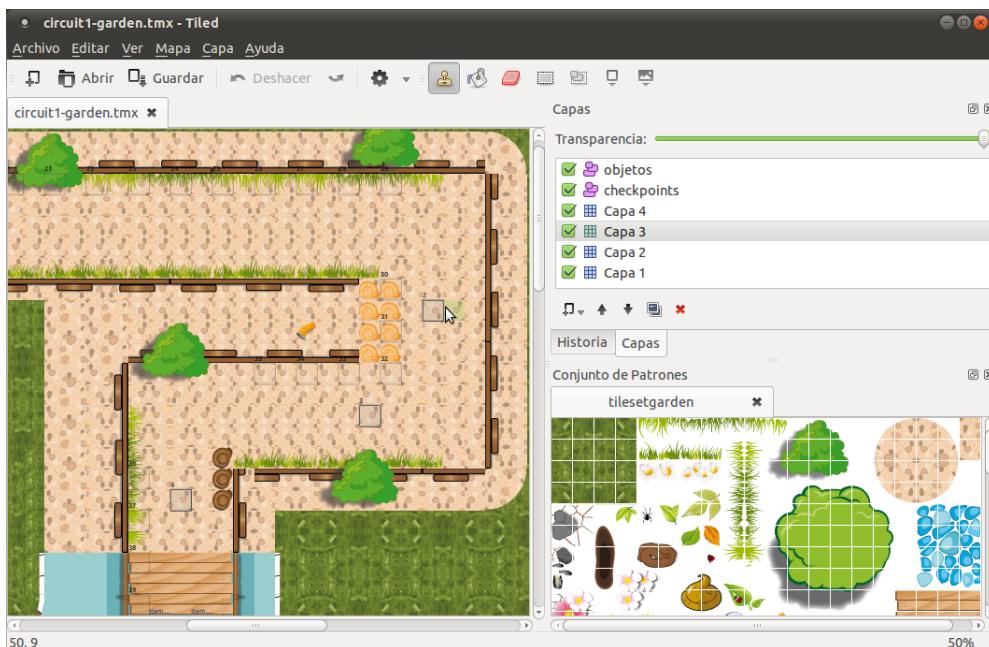


Figura A.5: Herramientas utilizadas: Captura del editor de mapas Tiled



## Apéndice B

# Manual de instalación

### B.1. Windows.

Para jugar a *Zycars* en el sistema operativo Windows no es necesario la instalación de ningún programa auxiliar, lo único que necesitaremos descargar será la versión correspondiente a Windows, llamada **zycars\_1.0\_win.zip** y descomprimirla. La descargaremos del siguiente enlace:

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

Tras descomprimir el archivo, accedemos a la carpeta generada llamada "zycars\_1.0\_win" y hacemos doble click sobre el archivo **zycars.exe** para comenzar a jugar.

### B.2. Linux: Ubuntu. Desde paquete Debian.

Para poder realizar la instalación de la aplicación desde el paquete Debian, debemos descargarnos el fichero Debian llamado **zycars\_1.0-1\_all.deb**. 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.

En el caso que no encontremos el juego instalado en la ruta anterior, lo podremos encontrar en Aplicaciones ->Otras ->zycars.

Si por algún motivo no podemos encontrar el enlace directo al juego en las secciones comentadas anteriormente, siempre podremos abrir una terminal y ejecutar el juego. De la siguiente forma:

**zycars**

### B.3. 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
```

## 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 coches dirigidos por el ordenador 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 ítems con los que hacer frente a nuestros competidores.

### C.2.2. Campeonato

En el modo Campeonato competiremos contra coches dirigidos por el ordenador 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 más puntos haya conseguido al concluir el campeonato, será el ganador del mismo. En este modo también encontraremos ítems durante las distintas carreras.

### C.2.3. Contrarreloj

En este modo de juego, el modo contrarreloj, el objetivo será batir los distintos récords 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 más rápida. Tendremos un máximo de 3 vueltas para mejorar los tiempos. En este modo de juego no encontraremos ítems, 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. Pantalla de juego

Una vez comenzemos una carrera en cualquiera de los modos de juego seleccionados, pasaremos a la pantalla de juego.



Figura C.4: Manual de usuario: Pantalla de juego

En la imagen anterior podemos ver las distintas partes de las que se compone la interfaz de la pantalla de juego. A continuación se describen cada una de ellas:

- **Posición actual del jugador:** en la parte inferior izquierda, encontramos la posición actual del jugador en carrera.
- **Posición de todos los jugadores:** en la parte superior derecha podemos ver los avatares de todos los personajes que intervienen en la carrera, ordenados según la posición de estos.
- **Contador de vueltas:** justo a la derecha de la posición de los jugadores encontramos el contador de vueltas, que nos indica la vuelta actual y el total de la carrera.
- **Ítem actual:** en la zona superior central de la ventana encontramos un cuadro donde aparecerá el ítem que tengamos en ese momento.
- **Tiempos de carrera:** los tiempos de carrera como pueden ser, mejor vuelta, tiempo total, vuelta actual y mejor tiempo total, los podemos encontrar en la parte superior derecha de la pantalla.

## C.6. 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.6.1. 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.5: 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.6.2. Pantalla

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



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

### C.6.3. 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 ítems y pausar el juego.



Figura C.7: Manual de usuario: Menú opciones - Controles

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

### C.7. Ítems

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.8: Manual de usuario: Bola de ítem.

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

- **Misil:** este ítem 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.9: Manual de usuario: Misil.

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



Figura C.10: Manual de usuario: Tres misiles.

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



Figura C.11: 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.12: 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.13: Manual de usuario: Macha de aceite.

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



Figura C.14: Manual de usuario: Turbo.

## 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 más 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.

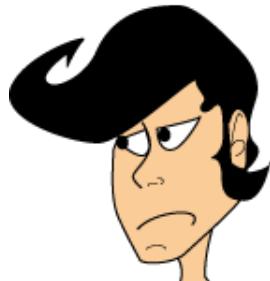


Figura D.3: Manual para añadir personajes: Ejemplo de avatar de corredor.

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:

```
1 <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:

```
1 <image code="character_image" name="character/character_image.png" alpha="True"/>
2 <image code="big_avatar" name="character/big_avatar.png" alpha="True"/>
3 <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 crea 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:

```
1 <car name_character="" sprite_code="" racer_image="" avatar="" max_speed=""
2 min_speed="" acceleration="" desaceleration="" rotation_angle="">
3 <!--NORMAL, NOACTION, RUN, FORWARD, REVERSE, DAMAGED, ERASE, YAW-->
4 <animations>
```

```

5   <animation name="normal" frames="0" delay="1"/>
6   <animation name="noaction" frames="0" delay="1"/>
7   <animation name="foward" frames="0" delay="1"/>
8   <animation name="run" frames="0" delay="1"/>
9   <animation name="reverse" frames="0" delay="1"/>
10  <animation name="damaged" frames="0" delay="1"/>
11  <animation name="erase" frames="0" delay="1"/>
12  <animation name="yaw" frames="0" delay="1"/>
13  <animation name="fall" frames="0" delay="1"/>
14  <animation name="turbo" frames="0" delay="1"/>
15  </animations>
16 </car>

```

Sólo debemos preocuparnos de las dos primeras líneas en las que deberemos llenar 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.
- **aceleration**: Aceleración del vehículo, valor entre 0.1 y 1.
- **desaceleration**: 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:

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

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

```
1 <character image="" name='' image_car=''
```

```
2 path_xml="cars/" speed=''
```

```
3 aceleration=''
```

```
4 rotation=''/>
```

Los distintos parámetros 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.
- **aceleration**: 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.



## Apéndice E

# Manual para añadir nuevos circuitos

En el siguiente manual se explicarán los distintos pasos que hay que llevar a cabo para crear y añadir nuevos circuitos a *Zycars* por cualquier persona sin necesidad de tener conocimientos sobre programación.

### E.1. Descarga e instalación de Tiled

Para la creación de los circuitos de *Zycars* se usa el programa de creación de mapa de tiles *Tiled*.

#### Descarga

Para descargarlo, nos vamos a la página principal de la aplicación: <http://www.mapeditor.org/>. En la parte derecha de la página podemos encontrar las distintas versiones de la aplicación para los sistemas operativos Windows y MAC OS X.

Si deseamos usar la aplicación en alguna distribución GNU/Linux deberemos bajarnos los fuentes de esta, pinchando en el enlace **Tiled Qt 0.7.0 source**.

#### Instalación

Para la instalación de *Tiled* en windows, únicamente deberemos ejecutar el archivo ejecutable que previamente hemos descargado previamente y seguir los pasos que se nos indica en el gestor de instalación.

Si deseamos ejecutar la aplicación en GNU/Linux, deberemos descomprimir el archivo descargado y acceder a la carpeta generada, dentro de esta encontraremos un archivo README, donde estarán todos los pasos necesarios para la correcta instalación y ejecución de la aplicación en GNU/Linux.

### E.2. Creando nuestro circuito

Una vez instalado *Tiled*, ya podremos empezar a crear nuestros circuitos. En primer lugar ejecutamos el programa, una vez hecho esto abrimos un de las plantillas ofrecidas para los 3 distintos tipos de temas disponibles en *Zycars*. Dichos archivos los encontraremos en *zycars/xml/circuits/*, y tendrán los siguientes nombres: *plantilla\_playa.tmx*, *plantilla\_jardin.tmx* y *plantilla\_cocina.tmx*.

Este tutorial se realizará usando como plantilla la referente a la playa, todos los pasos descritos aquí serán los mismo que habrá que llevar a cabo con cualquiera de las otras dos plantillas.

### E.2.1. Familiarización con la interfaz

Una vez abierta la plantilla que se desee, en nuestro caso la referente a la playa, podremos ver toda la interfaz y los componentes de esta, a continuación podemos ver una captura:

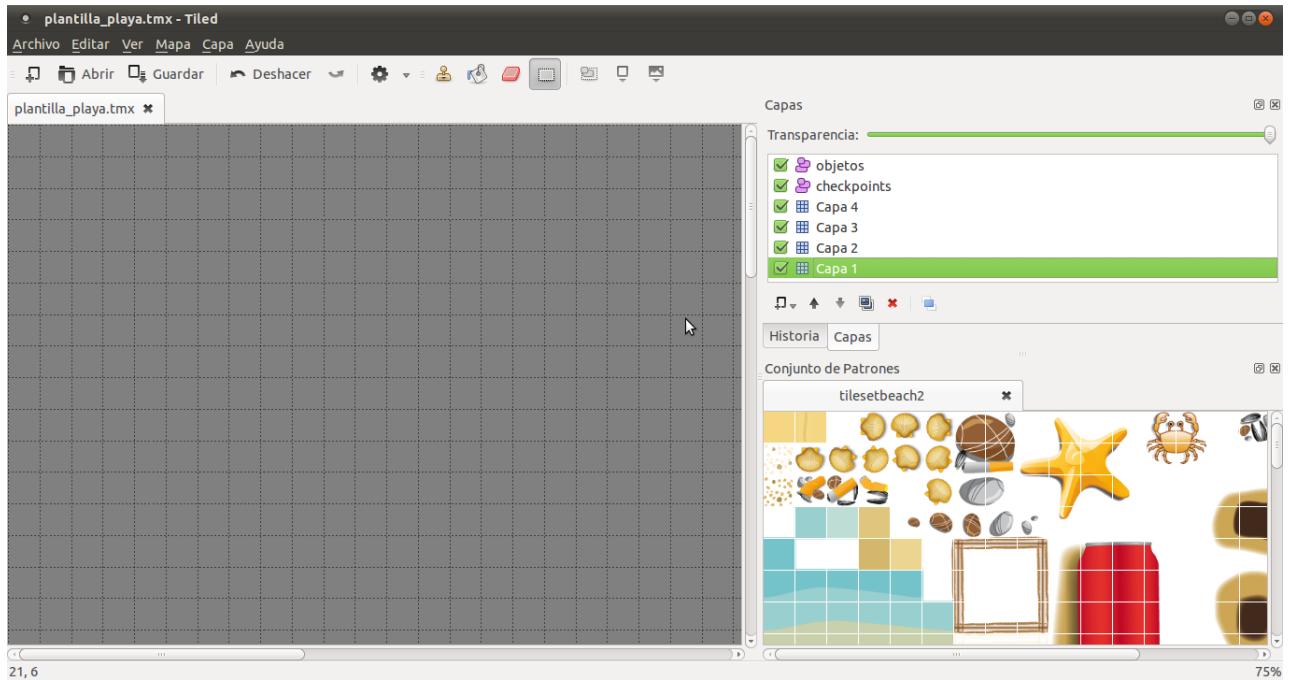


Figura E.1: Manual para añadir circuitos: interfaz de tiled.

Podemos diferenciar 4 zonas claramente que se describirán en los siguientes apartados.

#### Zona superior – Barra de herramientas

Aquí tendremos las distintas herramientas en referencia a la capa que nos encontremos en ese momento.

Si estamos en una capa de tiles, tendremos herramientas para poner tiles, llenar con tiles, borrar tiles y selección de un conjunto de tiles, de izquierda a derecha.

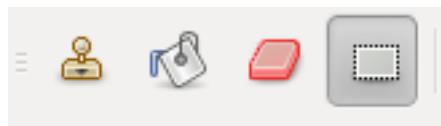


Figura E.2: Manual para añadir circuitos: herramientas capa de tiles.

Si la capa es de objetos, tendremos las herramientas de seleccionar objetos, insertar objetos e insertar objetos de patrones, de izquierda a derecha.



Figura E.3: Manual para añadir circuitos: herramientas capa de objetos.

### Zona izquierda – Lienzo

Esta será la zona donde iremos dibujando el circuito con los tiles disponibles en el tileset. Podremos hacer uso de las capas para poder situar elementos por encima de otros y crear sensación de lejanía.

Podremos usar zoom in y zoom out con la rueda del ratón a la vez que pulsamos la tecla cntrl.

### Zona derecha-superior – Capas

En esta otra parte de la interfaz, tendremos las distintas capas que dispondremos para crear el circuito. Se pueden añadir el número de capas que se desee, pero en la realización de circuito para Zycars, sólo usaremos las que se ven en la imagen a continuación:



Figura E.4: Manual para añadir circuitos: Capas.

Podemos diferenciar entre dos tipos de capas, las capas de tiles, donde dibujaremos los tiles que tenemos en el tileset y la capa de objetos, donde marcaremos donde se situarán alguno objetos necesarios de carreras, como pueden ser las cajas de ítems o los puntos de control de la carrera.

El papel de cada una de ellas es el siguiente:

- **Capa 1:** capa que sirve para situar el fondo del circuito
- **Capa 2:** capa donde podremos los tiles atravesables formando el recorrido del circuito y tiles de decoración.
- **Capa 3:** aquí podremos tiles que queramos que se vean por encima de los jugadores.
- **Capa 4:** sin uso actualmente.
- **Checkpoints:** capa de objetos donde podremos los puntos de control del circuito.
- **Objetos:** capa de objetos donde podremos las cajas de ítems y los puntos de la inteligencia artificial.

## Zona derecha-inferior – Tileset

En esta zona tendremos el tileset, es decir, la imagen que contiene el conjunto de tiles con el que crearemos el circuito. Al igual que en el lienzo, podremos hacer zoom in y zoom out con la ayuda de la rueda del ratón mientras pulsamos la tecla cntrl.

### E.2.2. Tipos de tiles.

Antes de comenzar deberemos diferenciar claramente los distintos tipos de tiles que existen en Zycars:

- **Atravesables:** son los tiles usados para añadir decoración al circuito, como fondo u otros elementos.
- **Colisionables:** aquellos tiles que el jugador no podrá atravesar, se usarán para hacer el trazado del circuito y añadir obstáculos en este.
- **Realentizadores:** tiles por los que al pasar por encima ralentizarán considerablemente la velocidad del jugador.

Para diferencia bien cada uno de los distintos tipos, en la carpeta zycars/multimedia/image/, tenemos una imagen de guia, sólo y únicamente sirve de guia, para saber de que tipo son los distintos tiles del tileset. En el caso de la playa, que es el que nos ocupa, la imagen se llama mixedbeach.png, en el caso del jardín y la cocina mixedgarden.png y mixedkitchen.png, respectivamente. A continuación un ejemplo de este tipo de imagen:

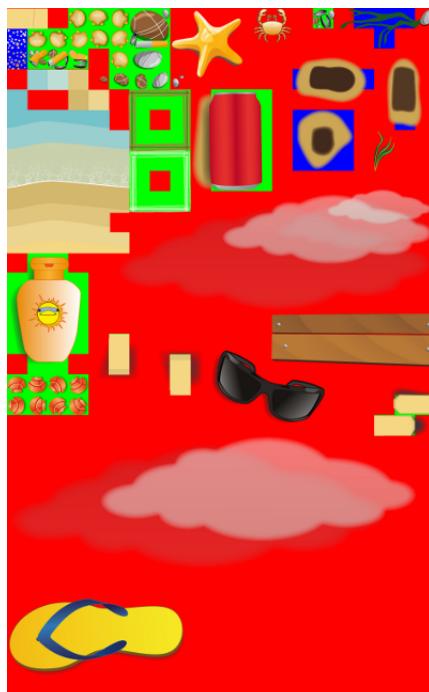


Figura E.5: Manual para añadir circuitos: Tileset con los tipos de tiles reflejados.

Como se puede ver en la imagen, tenemos 3 tipos de colores, para los tres tipos de tiles:

- **Rojo:** indican los tiles atravesables

- **Verdes:** indican los tiles colisionables
- **Azules:** indican los tiles que ralentizan

Una vez aclarado los distintos tipos de tiles que encontraremos para cada circuito, ya estamos preparados para comenzar la creación de circuitos.

### E.2.3. Añadiendo el fondo. Capa 1.

En primer lugar nos centraremos en la capa 1, en esta capa podremos el fondo del escenario, es conveniente que no pongamos ningún tile colisionables, nos centraremos únicamente en los tiles atravesables.

Por lo que en primer lugar deberemos seleccionar la capa 1 en ventana de capas, para indicar que vamos a comenzar a pintar en esa capa. Una vez hecho esto, podremos empezar a llenar el fondo del circuito. Para ellos tenemos muchos tiles con aspecto de arena, de distintos tonos para diferenciar entre arena mojada y arena seca.

El resultado que obtendríamos sería el siguiente:

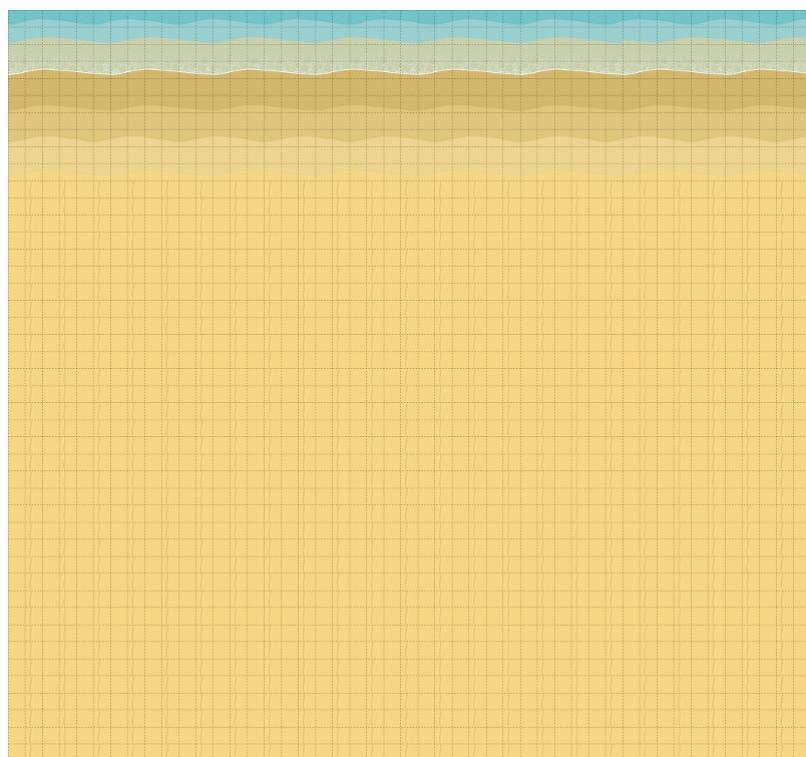


Figura E.6: Manual para añadir circuitos: Fondo circuito.

Una vez hecho el fondo de nuestro circuito, podremos pasar al siguiente paso, en el siguiente paso nos centraremos en realizar el trazado de nuestro circuito, donde competirán los distintos jugadores.

#### E.2.4. Añadiendo trazado circuito. Capa 2.

Para comenzar con el trazado del circuito, debemos seleccionar la capa 2. Antes de comenzar debemos fijarnos bien cuales de los tiles son colisionables, ya que el trazado siempre se realizara con tiles colisionables para que los jugadores no se puedan salir del circuito.

En el caso concreto de la playa, tenemos distintos tiles para realizar el trazado, como conchas tanto amarillas, como rojas o las líneas marrones y blancas del tileset. Una vez claro los tiles disponibles, podemos comenzar a dibujar el trazado.

Se recomienda que todo el trazado del circuito tenga el mismo ancho, el todos los circuitos que ya existen en *Zycars*, se usa normalmente un ancho de 5 tiles para las pistas. Debemos tener claro el ancho máximo de la pista, ya que posteriormente deberemos señalarlo en las opciones del circuito.

Un ejemplo de un posible trazado se muestra a continuación:

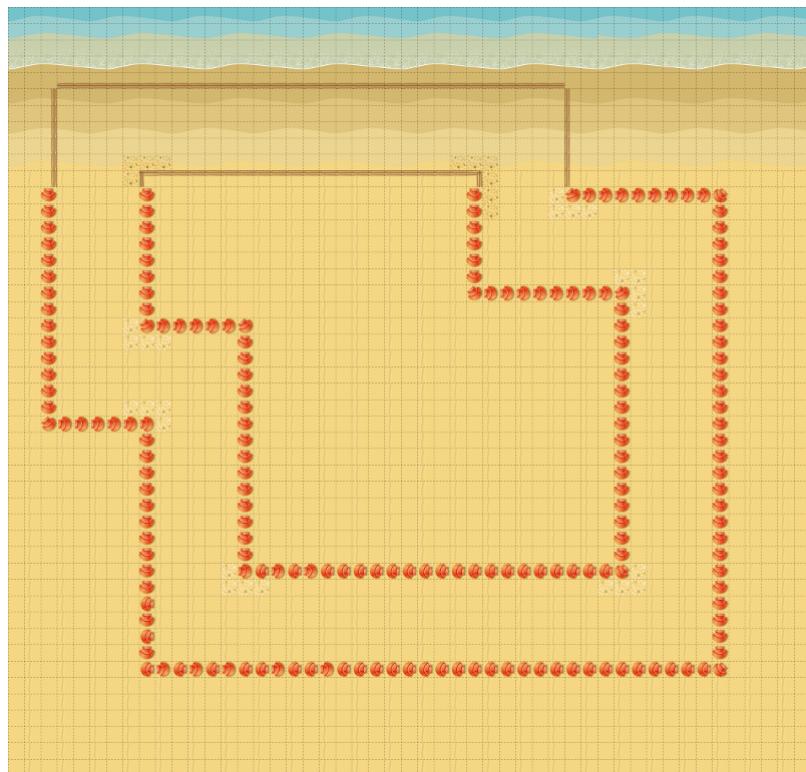


Figura E.7: Manual para añadir circuitos: Trazado circuito.

Bueno tras este paso, ya tenemos lo más esencial de cualquier circuito que se precie. En el siguiente paso añadiremos elementos de decoración por todo el mapa y obstáculos, para hacer más atractivo el trazado del circuito y más ameno a la hora de competir en él.

#### E.2.5. Añadiendo decorado. Capa 2.

En esta sección y como se ha comentado en la sección anterior, nos dedicaremos a añadir obstáculos en el circuito, como pueden ser tiles que ralenticen la velocidad de los jugadores u objetos colisionables en

medio del trazado del circuito. Como podemos ver en el tileset que estamos usando, tiles realentizadores son aquellos que parecen manchas de humedad o los tiles que tiene un conjunto de pequeñas piedras.

También añadiremos objetos de decoración fuera del circuito, para hacer al escenario más agradable y vistoso. Tiles de adorno, tenemos las gafas de sol, la chancla o por ejemplo la lata de refresco.

Para añadir el decorado y obstáculos seguiremos en la capa 2, con cuidado de no reemplazar los tiles que representan el trazado. Un ejemplo de resultado obtenido sería el siguiente:

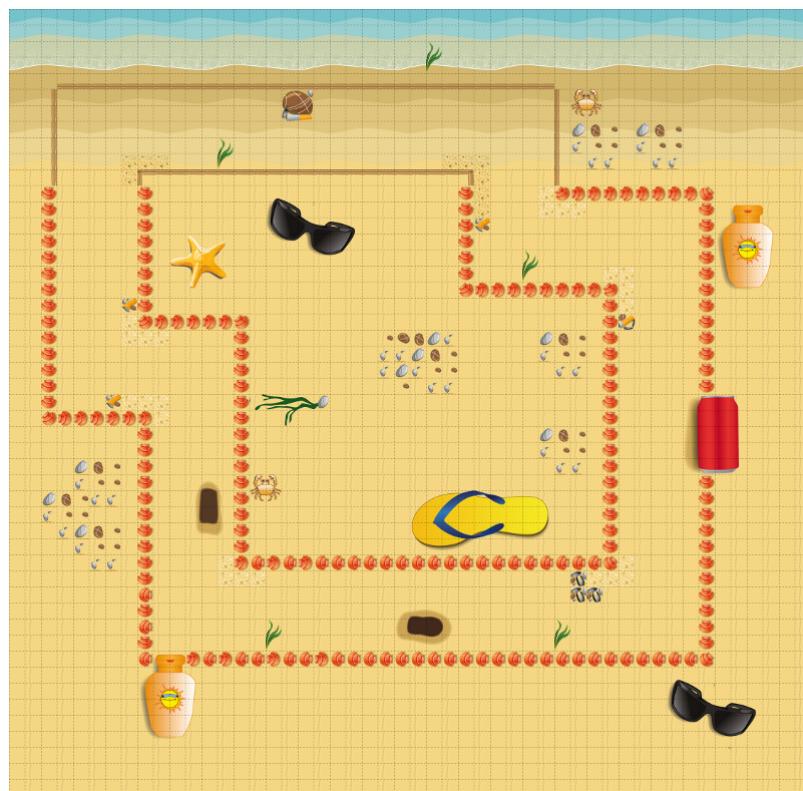


Figura E.8: Manual para añadir circuitos: Decorado circuito.

#### E.2.6. Añadiendo elementos superiores. Capa 3.

El último paso en el que añadiremos tiles al circuito, será en esta sección, donde añadiremos, si lo deseamos, aquellos elementos que deseamos que se vean por encima de los jugadores.

En este tileset en cuestión, tenemos tiles que representan nubes, por lo que son lo ideal para añadir en esta capa 3. El resultado quedaría de la siguiente forma.

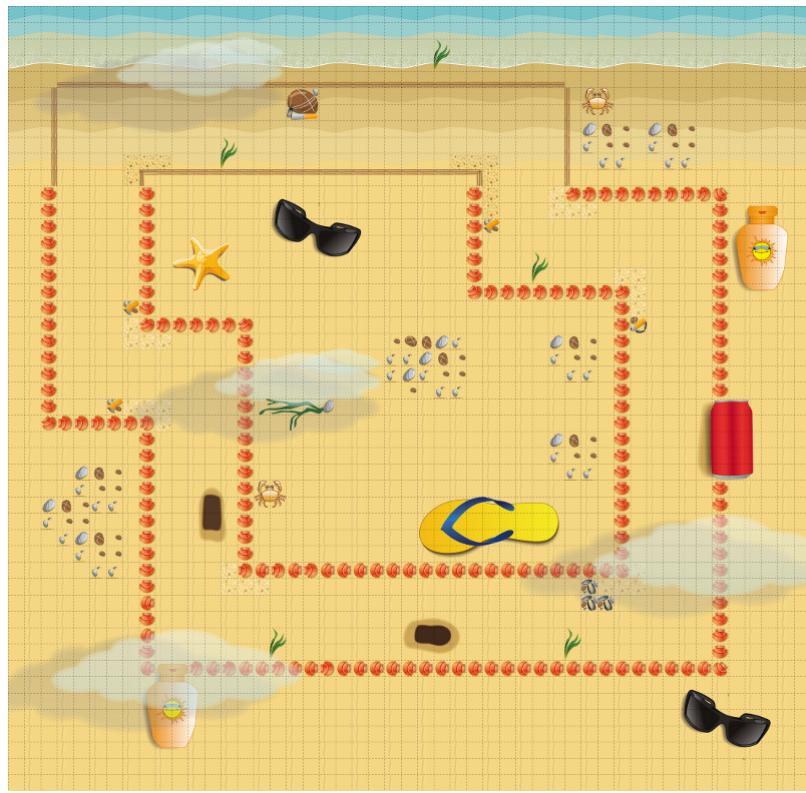


Figura E.9: Manual para añadir circuitos: Capa 3 del circuito.

Una vez añadido los elementos de la tercera capa, hemos concluido la parte de creación y decoración del circuito. En las secciones que vienen a continuación añadiremos todos los elementos necesarios para el correcto funcionamiento del juego en el nuevo circuito. Por ello, en las siguientes secciones sólo trabajaremos con las capas de objetos.

#### E.2.7. Rellenado la capa checkpoints.

Como hemos comentado en la sección anterior, en las secciones que viene a continuación nos centraremos en añadir todos los objetos necesarios en el circuito. Trabajaremos con las capas de objetos y la de checkpoints disponible en la plantilla. Centrándonos en primer lugar en la capa de checkpoints.

Este puede ser uno de los pasos más importantes y trabajosos a la hora de crear un circuito. En este paso, indicaremos donde se situará la meta y todos los puntos de control del circuito, que controlen que las vueltas se realizan de forma correcta.

#### Indicando la salida.

En primer lugar indicaremos donde estará la salida y meta del circuito, para ello seleccionamos la capa de checkpoints y situamos la meta donde queramos. Una aclaración si la meta va a ser horizontal se deberá colocar el objeto en la parte izquierda del trazado, si en cambio es vertical, se deberá colocar en la parte superior del trazado. En este ejemplo la meta será horizontal, su colocación será la siguiente:

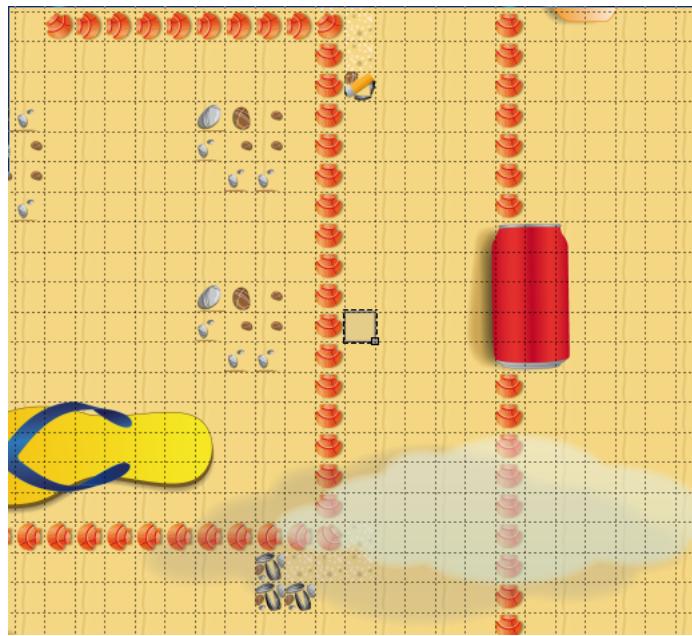


Figura E.10: Manual para añadir circuitos: Checkpoints paso 1.

Una vez que hemos añadido el objeto le damos el tamaño de un tile y pulsamos con el botón derecho del ratón sobre él y seleccionamos las propiedades del objeto. Ahora deberemos indicar que orientación que tiene en el atributo tipo, pondremos "GoalV" si es vertical o "GoalH" si es horizontal. En el atributo nombre pondremos el valor 0, para indicar que es la salida. En nuestro caso al ser la salida horizontal el resultado es el siguiente:



Figura E.11: Manual para añadir circuitos: Checkpoints paso 2.

Una vez añadido la salida, tendremos que indicar el ancho máximo de la pista, ancho de la pista en la meta y la orientación en la que saldrán los jugadores, si hacia arriba(270)<sup>1</sup>, abajo(90), izquierda(180) o derecha(0). En este caso a ser la salida horizontal, los jugadores sólo podrán salir hacia arriba o abajo. Por lo que en la parte superior de Tiled seleccionamos Mapa y propiedades del mapa y completamos las propiedades necesarias, que son ".ancho\_meta"uyo valor será 5, ".ancho\_pista"que también será 5 y "grado\_coche"que deberá ser en nuestro caso 90 o 270. Para este ejemplo se ha elegido que los jugadores saldrán hacia arriba por lo que daremos el valor 270.

Nombre	Valor
ancho_meta	5
ancho_pista	5
collision_map	collisionmapbeac...
grado_coche	270
music	
tileset_alto	34
tileset_ancho	21
<nueva propiedad>	

**Aceptar** **Cancelar**

Figura E.12: Manual para añadir circuitos: Checkpoints paso 3.

### Añadiendo puntos de control.

Una vez que hemos indicado donde se encontrará la salida y en que orientación saldrán los jugadores, deberemos añadir todos los puntos de control necesario para el control correcto de las vueltas.

En este caso a la salir los jugadores hacia arriba empezaremos a añadir objetos por encima de la meta. Esto puntos de control deberán cumplir las mismas reglas de la meta, a la izquierda del trazado cuando sean horizontales y en la parte superior cuando sean verticales. Por encima de la meta iremos añadiendo objetos, poniendo en el tipo "Horizontalz el nombre el número que corresponda, partiendo que el primero añadido será el 1. Quedando de la siguiente forma:

---

<sup>1</sup>Entre paréntesis se indica el valor que tendremos que añadir en función de la orientación

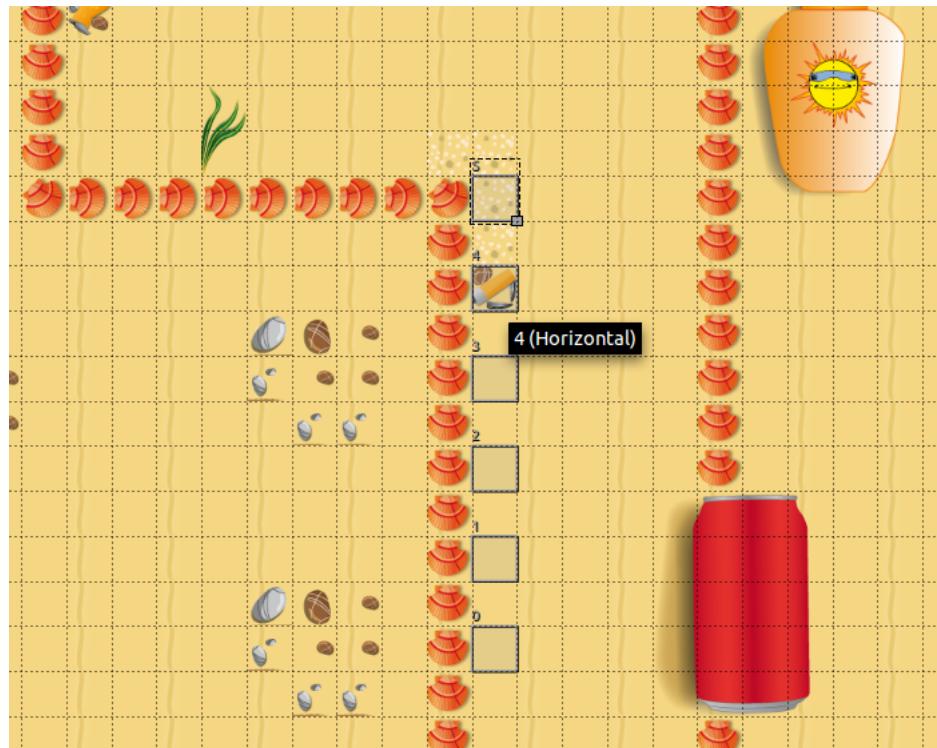


Figura E.13: Manual para añadir circuitos: Checkpoints paso 4.

Para los puntos de control verticales, un ejemplo de como quedaría:

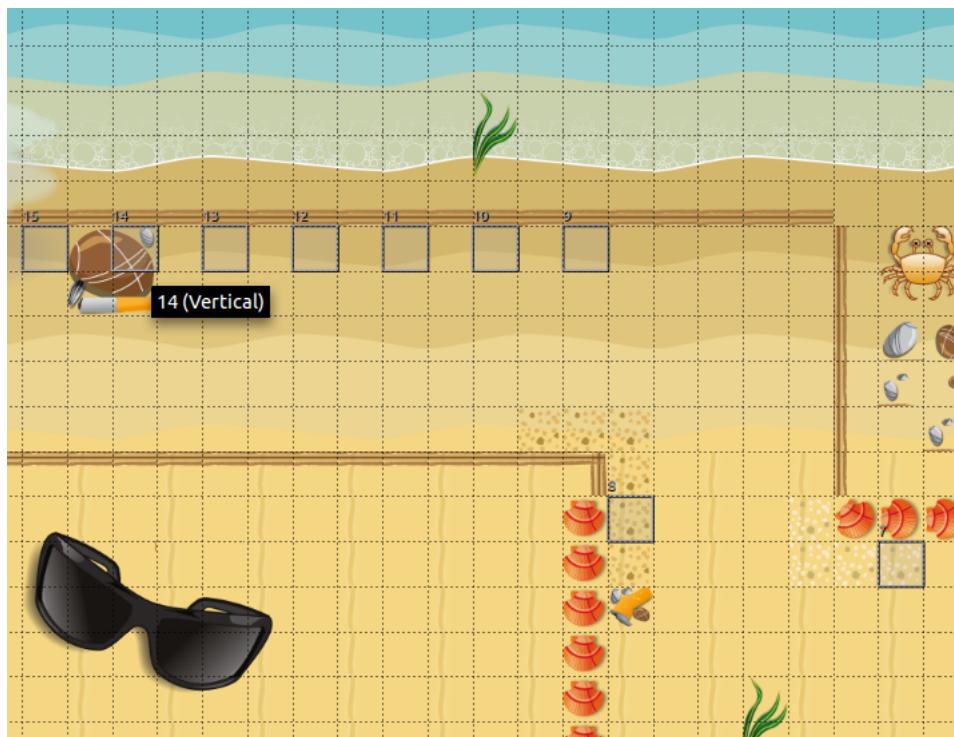


Figura E.14: Manual para añadir circuitos: Checkpoints paso 5.

Siguiendo los mismos pasos, deberemos añadir los puntos de control a lo largo de todo el trazado completo, siguiente la orientación de este hasta llegar de nuevo a la salida de este. Quedando como resultado la siguiente imagen:

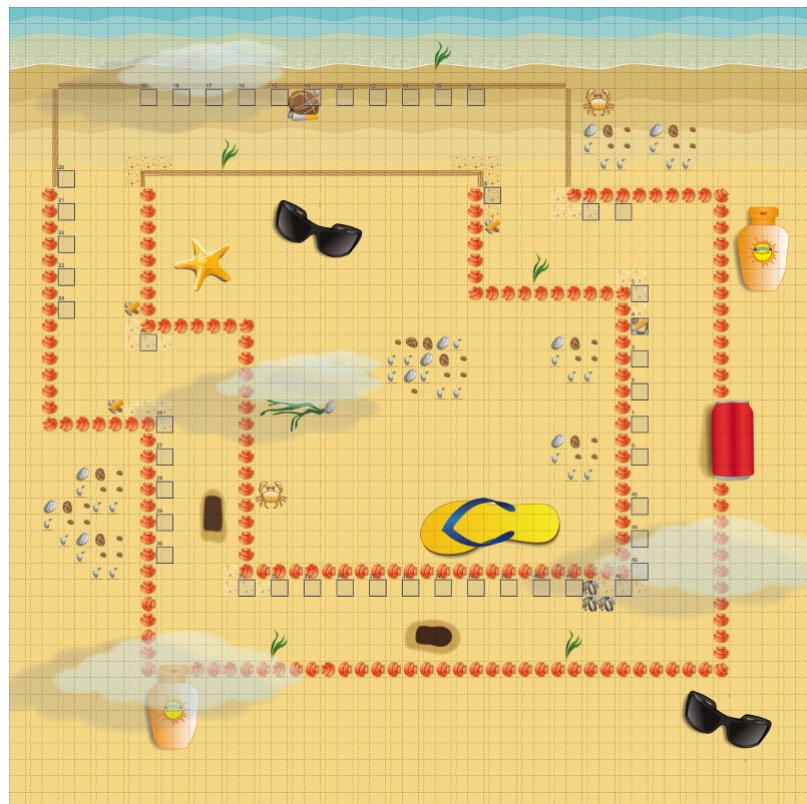


Figura E.15: Manual para añadir circuitos: Checkpoints paso 6.

Llegado a este punto ya hemos completado la introducción de todos los puntos de control necesarios en el circuito.

### E.2.8. Rellenando la capa objetos.

En esta sección nos centraremos en añadir los distintos objetos necesarios en el circuito, como pueden ser, las cajas de ítem y los puntos de control de la inteligencia artificial.

#### Añadiendo cajas de ítems.

Comenzaremos indicando mediante objetos donde se colocarán las cajas de ítems. Para ello nos situamos en la capa de objetos y seleccionamos en la barra de objetos añadir herramientas. Ahora hacemos click en alguna parte de escenario que esté dentro del trazado del circuito. Tras hacer esto aparecerá un elemento transparente, al que podremos dar el tamaño que deseemos (el tamaño no afectará). Personalmente siempre le doy el tamaño de un tile, para que quede más claro.

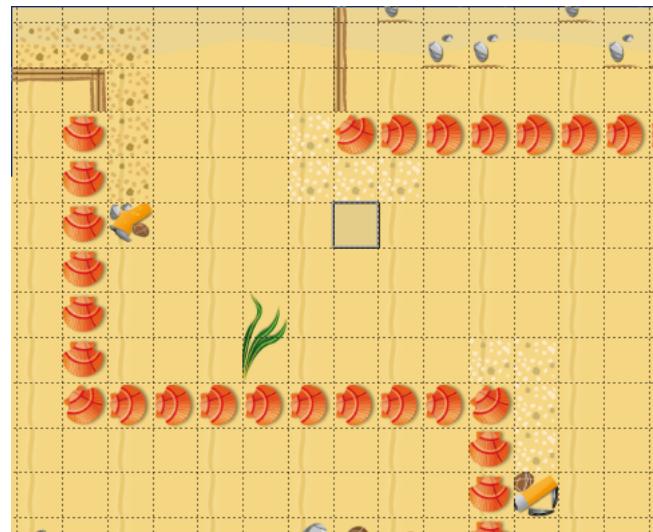


Figura E.16: Manual para añadir circuitos: Cajas de ítems paso 1.

Una vez lo tengamos como la imagen anterior, pulsamos con el botón derecho del ratón sobre este y seleccionamos "Las propiedades del objeto". Una vez se nos habrá la ventana, añadiremos tanto en nombre como en tipo la palabra `Item_box`". Debe quedar como en la siguiente imagen:



Figura E.17: Manual para añadir circuitos: Cajas de ítems paso 2.

Una vez hecho esto, solo nos quedará hacer lo mismo con tantas cajas de ítems que deseemos añadir. También podremos copiar el primer elemento e ir pegándolo por todo el circuito.

### Añadiendo control de la inteligencia artificial

El último paso para completar la creación de nuestro circuito, será añadir los distintos puntos de control necesarios para que la inteligencia artificial realice el recorrido correctamente. Para ellos deberemos añadir unos objetos en la capa de objetos, con el tipo `checkz` en el nombre el número correspondiente al

punto de control comenzando desde 1. Añadiremos uno en cada curva del circuito, siguiente el trazado de este en el orden en el que lo realizarán los jugadores. Quedando como resultado final la siguiente imagen:

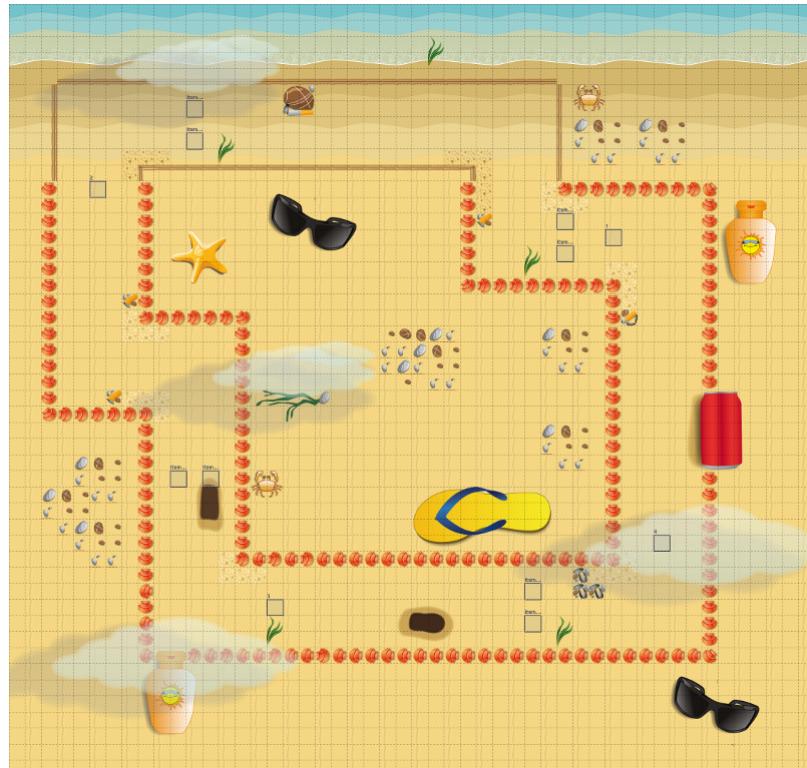


Figura E.18: Manual para añadir circuitos: Puntos de control de la inteligencia artificial.

### E.2.9. Modificando el xml generado.

Una vez finalizado la creación del circuito, deberemos guardarla en la carpeta "zycars/xml/circuits", que es donde se encuentran todos los circuitos del juego.

Tras hacer esto nos situamos en dicha carpeta y abrimos el archivo tmx generado al almacenar el circuito. Al inicio del xml, aproximadamente en la línea 13, deberemos ver la siguiente línea:

```
| <image source="../../multimedia/image/tilesetbeach2.png" width="945" height="1530"/>
```

En esta línea deberemos borrar el siguiente trozo: "../../multimedia/image/", sin las comillas. Quedando de la siguiente forma:

```
| <image source="tilesetbeach2.png" width="945" height="1530"/>
```

### E.2.10. Añadiendo el mapa al juego.

Por último, para añadir el circuito al juego, deberemos acceder al xml "fastracemenu.xml" que se encuentra en "zycars/xml/menu", una vez en el archivo deberemos buscar la etiqueta <layers>, dentro

de esta estarán las distintas capas con los circuitos para cada uno de los campeonatos, si vamos aadir un circuito de playa nos centraremos en la capa cuyo atributo name sea "Copa Chancla", si es un circuito de jardín será "Copa arbusto" y si es de cocina será "Copa rodillo".

Para cada una de ellas tenemos 4 circuito, por lo que deberemos sustituir el que queramos, modificando el atributo circuit\_file, donde añadiremos el nombre del circuito que hemos creado.

Por ejemplo si deseamos sustituir el primer circuito de la copa chancla:

```
1 <layers>
2   <layer name="Copa Chancla" font_code="cheesebu" size="30">
3     <buttons>
4       <button type="image_button" xml_file="menu/circuitbutton.
      xml"
5         circuit_file='circuits/circuit1-beach.tmx' font="cheesebu"
6         text="circuit1-beach" image_circuit="circuit1" image="
      logo_beach1"
7         image_x="6" image_y="7" center="True" x="80" y="345"
8         show_text='False'></button>
```

Y el circuito que hemos creado se llama "mi\_circuito.tmx", debería quedar de la siguiente forma:

```
1 <layers>
2   <layer name="Copa Chancla" font_code="cheesebu" size="30">
3     <buttons>
4       <button type="image_button" xml_file="menu/circuitbutton.
      xml"
5         circuit_file='circuits/mi_circuito.tmx' font="cheesebu"
6         text="circuit1-beach" image_circuit="circuit1" image="
      logo_beach1"
7         image_x="6" image_y="7" center="True" x="80" y="345"
8         show_text='False'></button>
```

Una vez hecho esto ya podremos disfrutar del circuito que hemos creado.



# Bibliografía

- [1] Página oficial sobre el lenguaje de programación *Python*.  
<http://www.python.org/>.
- [2] González Duque, Raúl. *Python para todos*.
- [3] Pilgrim, Mark. *Dive into Python*. Apress, 2004. 413p. ISBN:978-1590593561.
- [4] Página oficial sobre la biblioteca *Pygame*.  
<http://pygame.org/news.html>.
- [5] Traducción de la documentación de *Pygame*.  
<http://www.losersjuegos.com.ar/traducciones/pygame>
- [6] Larman, Craig. *Applying UML and Patterns*, 3<sup>a</sup> Edición. Prentice Hall, 2004. 736p. ISBN:978-0131489066.
- [7] Russell, Stuart y Norvig, Peter. *Artificial Intelligence Modern Approach*, 2003. 905. ISBN:978-0136042594.
- [8] Artículo de wikipedia inglesa sobre el algoritmo de búsqueda *A\**.  
[http://en.wikipedia.org/wiki/A\\*\\_search\\_algorithm](http://en.wikipedia.org/wiki/A*_search_algorithm)
- [9] Artículo de wikipedia española sobre el algoritmo de búsqueda *A\**.  
[http://es.wikipedia.org/wiki/Algoritmo\\_de\\_búsqueda\\_A%2A](http://es.wikipedia.org/wiki/Algoritmo_de_búsqueda_A%2A)
- [10] Artículo sobre el algoritmo de búsqueda *A\**.  
<http://razonartificial.com/2010/03/a-pathfinding-camino-optimo/>
- [11] Página oficial de la herramienta para documentar código *Doxxygen*.  
<http://www.stack.nl/~dimitri/doxygen/>
- [12] Lambert M. Surhone; Mariam T. Tenroe Y Susan F. Henssonow (Ed). *Doxxygen*. Betascript Publishing, 2010. 168p. ISBN:978-3639910025.
- [13] 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](http://osl2.uca.es/wikiformacion/index.php/LaTeX_para_Proyecto_Fin_de_Carrera).
- [14] Página sobre la herramienta para la edición de mapas *Tiled Map Editor*.  
<http://www.mapeditor.org/>
- [15] Página sobre la herramienta de generación de diagramas *Dia*.  
<http://projects.gnome.org/dia/>

[16] Página sobre la herramienta para realizar bocetos *Pencil Project*.

<http://pencil.evolus.vn/en-US/Home.aspx>

[17] Foguel, Karl. *Producing Open Source Software*, 1<sup>a</sup> edición. O'Reilly Media, 2005. 304p.  
ISBN:978-0596007591.

[18] Página oficial de la herramienta de edición de imágenes *GIMP*.

<http://www.gimp.org/>

# 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.