Zycars: juego de conducción en 2D

Alumno: José Jesús Marente Florín
Directores: Manuel Palomo Duarte y Juan Manuel Dodero Beardo

30 de agosto de 2011

Resumen

El siguiente documento se presenta a modo de resumen que complementa la Memoria del mismo Proyecto Fin de Carrera, entregada de forma simultánea a este resumen. El proyecto consiste en un videojuego de conducción en dos dimensiones.

Índice

1. Objetivos

El objetivo del proyecto es realizar un videojuego de conducción en dos dimensiones con vista cenital[1]. Se podría decir que el juego tendrá tintes de juegos como Micro Machines, disponible para diversas plataformas, y del Mario Kart de nintendo.

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.

2. Motivaciones

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.

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.

[1] Los elementos son vistos desde arriba

3.2. Fase de análisis

Esta etapa está dividida principalmente en las dos partes siguiente:

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

3.3. Fase Aprendizaje

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

- **Aprendizaje de *Python***: periodo empleado para el aprendizaje del lenguaje de programación *Python*, durante esta etapa se consultó varios libros sobre lenguaje, así como foros de internet y páginas web. Para un aprendizaje más ameno y llevadero, se 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 la siguientes:

- **Motor básico**: implementación de las necesidades básicas del proyecto, como control del teclado, carga de recursos, movimiento de los vehículos.
- **Carga de escenario**: carga de los circuitos que 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 se 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 un 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

Diagrama de Gantt de la planificación comentada (Figura 7 y 8, en la página 12 y 13 respectivamente)

4. Descripción general

El proyecto consiste en un juego de carreras en dos dimensiones con vista cenital, en el que se podrá competirá contra la inteligencia artificial.

Figura 1: Logo de Zycars

4.1. 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:

4.1.1. 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 controlados por la inteligencia artificial a lo largo de un campeonato completo, el cual habremos elegido previamente.

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

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

Contrarreloj 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.

4.1.2. 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 básico 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.

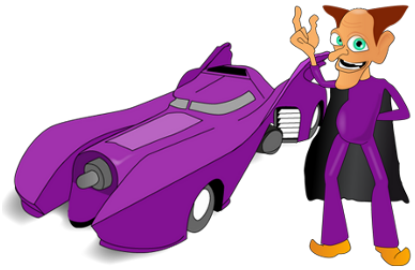

Figura 2: Personaje de Zycars.

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 3: 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 encuentres 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.

4.2. 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

5. 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.

5.1. Carga desde ficheros

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

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 lo menús, y las distintas imágenes que pueden aparecer en estos. Respecto a los personaje 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.

5.2. 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. En un principio se barajaron varias alternativas.

Finalmente se optó por usar el programa *Tiled*, 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.

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, lo tiles tendrían colores planos indicando de que 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 que tipo son. A continuación se muestran dos imágenes como ejemplo:

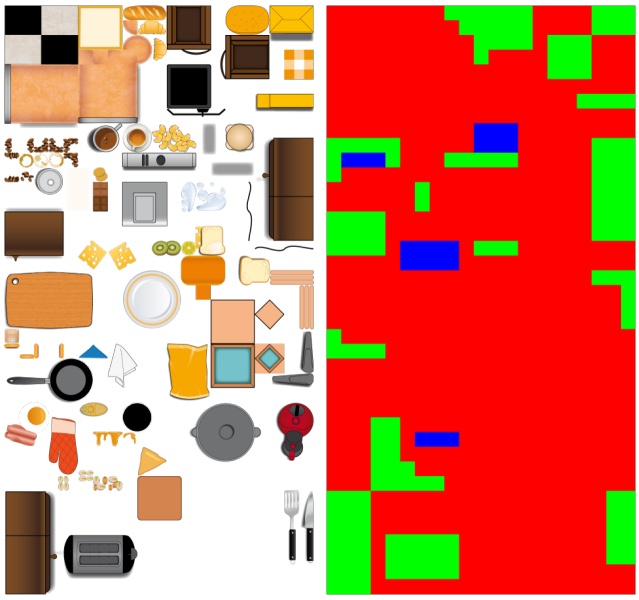

Figura 4: Conjunto de tiles y mapa de colisiones del mismo

5.3. 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.

5.3.1. Colisión con el escenario

Como se comento 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 muestra un ejemplo visual:

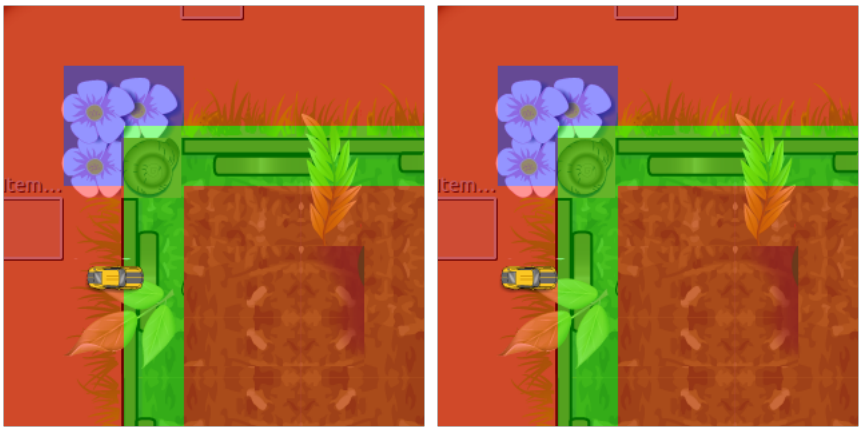

Figura 5: Colisión con el escenario

5.3.2. Colisiones entre objetos

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

A continuación se exponen las distintas situaciones que pueden suceder:

- **Colisión vehículo-vehículo**: dos vehículos en carrera colisionan entre sí, en este caso debemos comprobar cual de los dos vehículos ha colisionado con el otro, es decir, cual se ha interpuesto. En ese caso corregiremos la posición de ese vehículo y produciremos algún tipo de rebote en función de la velocidad que llevara en ese momento.
- **Colisión vehículo-ítem**: en este caso se responderá a la colisión dependiendo del tipo de ítem con el que hemos colisionado.

5.4. 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 la inteligencia artificial.

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

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

5.4.1. Realización del recorrido. Algoritmo de búsqueda A*

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

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

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.

5.4.2. Lanzamiento de ítems.

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: Segmentos de la inteligencia artificial

6. Tecnologías implicadas

6.1. Lenguaje de programación

Se decidió usar *Python* como lenguaje principal para el desarrollo de *Zycars*. *Python* es un lenguaje de programación interpretado, de alto nivel, usa tipado dinámico, es fuertemente tipado y multiplataforma. Se trata de un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Cabe destacar que se han obtenido unos resultado muy satisfactorios y ha cumplido todas las expectativas esperadas.

6.2. Biblioteca gráfica

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 necesario 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.

6.3. 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.

6.4. 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*.

6.5. Documentación del código

Para la documentación del código me decanté por usar *Doxygen*, 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 *Doxygen*, por lo que nos ahorra trabajo y sigue la normativa de código *Python*.

6.6. Realización de diagramas: Dia

Dia es un programa de creación de diagramas en GNU/Linux, MacOS 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.

6.7. Programa de edición de escenarios: Tiled

El programa de edición de mapas *Tiled* 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.

7. Conclusiones

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

7.1. 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.

7.2. 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, LaTeXes una de esas en 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. 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 las vueltas más rápida en cada uno de los circuitos, almacenandolas en un fichero, con el objetivo de poder visualizarlas posteriormente.

Figura 7: Diagrama de Gantt 1/2

WBS	Nombre	Trabajo
4.5	**Menus**	**26d**
4.5.1	Principal	5d
4.5.2	Opciones	6d
4.5.3	Selección personaje	6d
4.5.4	Selección circuito	7d
4.5.5	Pausa	2d
4.6	Control de carrera	15d
4.7	**Items**	**21d**
4.7.1	Bola de items	3d
4.7.2	Misil	3d
4.7.3	Bola	3d
4.7.4	3 Misiles	3d
4.7.5	Aceite	3d
4.7.6	Chicle	3d
4.7.7	Turbo	3d
4.8	HUD Carrera	15d
4.9	Inteligencia Artificial	40d
4.10	**Modos de juego**	**35d**
4.10.1	Carrera rápida	10d
4.10.2	Contrarreloj	10d
4.10.3	Campeonato	15d
4.11	Creación de circuitos	30d
5	Pruebas y soluciones de bugs	250d
6	Redacción de la memoria	257d

Figura 8: Diagrama de Gantt 2/2

Referencias

[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*, 3ª 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

[9] Artículo de wikipedia española sobre el algoritmo de búsqueda *A**.
http://es.wikipedia.org/wiki/Algoritmo_de_búsqueda_A*

[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 *Doxygen*.
http://www.stack.nl/~dimitri/doxygen/

[12] Lambert M. Surhone; Mariam T. Tenroe Y Susan F. Henssonow (Ed). *Doxygen*. 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.

[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*, 1ª edición. O'Reilly Media, 2005. 304p. ISBN:978-0596007591.

[18] Página oficial de la herramienta de edición de imágenes *GIMP*.
http://www.gimp.org/