Universidad de las Ciencias Informáticas
Facultad 5 “Entornos Virtuales”.

# Editor de Pistas de Carreras para el proyecto Juegos Consola.

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.


Ismael Viamontes Marrero y Sándor Labrada Garay.
**Autores**

Ing. Igr Alexánder Fernández Saúco.
**Tutor**


**Ciudad de La Habana**

**Junio 2008**

# DECLARACIÓN DE AUTORÍA

Declaramos ser los autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de junio del año 2008.

| _____ | _____ | _____ |
|---|---|---|
| Firma del Autor | Firma del Autor | Firma del Tutor |
| Ismael Viamontes Marrero | Sándor Labrada Garay | Ing. Igr A. Fernández Saúco |

# Datos de Contacto

**Nombre y Apellidos:** Igr Alexánder Fernández Saúco.

**Edad**: 28.

**Ciudadanía**: Cubana.

**Institución**: Universidad de las Ciencias Informáticas (UCI).

**Título**: Ingeniero Informático.

**Categoría Docente**: Profesor Instructor.

**E-mail**: alexanderfs@uci.cu

**Jefe del grupo de Arquitectura y Tecnologías, Dirección Técnica, IP – UCI.**

*"Es de importancia para quien desee alcanzar una certeza en su investigación, el saber dudar a tiempo."*

*Aristóteles*

# DEDICATORIA

*A mi mamá, que supo ser mi familia entera aún sabiendo que ya la tenía.*

***Ismael.***

*A mi familia y en especial a mi esposa e hijo.*

***Sándor.***

# Agradecimientos

*Buscaba en vano las palabras que se me escaparon justo cuando iba a decirlas ¿Adónde se habrán ido esas palabras que tenía en la punta de la lengua? ¿Habrá algún lugar donde se juntan las palabras que no quisieron quedarse? ¿Un reino de las palabras perdidas? Las palabras que se me fueron, ¿dónde me estarán esperando?*

*Me puse a recortar palabras de los diarios, palabras de todos los tamaños, y las guardé en cajas.*

*A esas que logré retener les propuse un trato, ellas se mezclarían para mí y yo les daría la libertad de hacerlo como quisieran, solo debían contar que les estoy agradecido:*

*A mis padres,*

*A mi familia,*

*A mis amigos,*

*A Sándor, mi compañero de tesis,*

*A mis compañeros del proyecto Juegos Consola,*

*A mi tutor, por sus correcciones a tiempo,*

*A mis profes,*

*A todos ustedes.*

*Muchas Gracias…*

***Ismael.***

*A la Revolución, por pensar que un mundo mejor es posible, a nuestra Universidad por abarrotarnos de conocimientos y forjarnos como la tropa del futuro.*

*A mi madre que ha sabido confiar en mí, a mi familia que de una forma u otra me ayudaron a lograrlo. A mis amigos que tanto me han apoyado y a mi mujer que ha sido mi inspiración.*

***Sándor.***

# Resumen

En los últimos años los videojuegos de carreras han sido favorecidos por la comunidad de jugadores, estas aplicaciones gráficas se caracterizan por el hecho de que los competidores tienen una pista bien definida que recorrer, así como leyes de tránsito que respetar, esto es logrado mediante los Editores de Pistas de Carreras, que constituyen la herramienta con la que cuentan los usuarios para editar nuevos circuitos de carreras a los mapas de su videojuego.

La inexistencia de un Editor de Pistas de Carreras en el proyecto Juegos Consola determinó la necesidad del desarrollo del mismo. Este trabajo propone y desarrolla una aplicación que permite cargar mapas en formato BSP para editar un grafo de caminos sobre el mismo, el cual será almacenado en un fichero binario de extensión TRK, facilitando así el trabajo de los desarrolladores del videojuego Rápido y Curioso, permitiéndoles de esta forma acelerar el trabajo de los diseñadores de las pistas de carreras y alcanzar un producto en menor tiempo. Además de que la implementación de esta herramienta brindará a los usuarios del videojuego la posibilidad de personalizar sus pistas de carreras.

**PALABRA CLAVE:** Editor de Pistas de Carreras, Grafo de Caminos.

**Índice de Contenido.**

**Introducción .....1**
Organización del documento .....3
**Capítulo 1. Fundamentación Teórica.....5**
1.1. Videojuegos de Carrera .....5
1.2. Editores de Niveles .....6
1.3. Editores de Pistas de Carreras .....7
1.4. Formatos de los Ficheros Gráficos 3D .....9
1.4.1.1. Características del formato BSP .....10
1.4.1.2. Características del formato 3DS .....12
1.4.1.3. Características del formato MD2 .....12
1.5. Motores Gráficos .....13
1.6. G3D Engine .....15
1.7. Metodologías y Herramientas de Desarrollo .....17
1.7.1. Metodología .....17
1.7.2. Herramientas de Desarrollo .....18
1.7.2.1. Visual Studio 2005 .....18
1.7.2.2. Rational Rose 2003 .....18
1.7.3. Lenguajes .....19
1.7.3.1. Lenguaje de Programación .....19
1.7.3.2. Lenguaje de Modelado .....20
**Capítulo 2. Solución Propuesta .....21**
2.1. El Sistema .....21
2.2. Grafos de Caminos .....22
2.3. Especificación del formato de fichero a exportar .....23
2.3.1. Estructura del fichero TRK .....23
2.3.1.1. Nombre del Mapa Asociado .....24

2.3.1.2. Header ..... 24
2.3.1.3. Bloque de Datos ..... 25
**Capítulo 3. Descripción de la Solución Propuesta ..... 27**
3.1. Reglas del Negocio ..... 27
3.2. Modelo de Domino ..... 27
3.3. Conceptos del Modelo de Dominio ..... 28
3.4. Captura de Requisitos ..... 29
3.4.1. Requisitos Funcionales ..... 29
3.4.2. Requisitos No Funcionales ..... 30
3.5. Modelo de Casos de Uso del Sistema ..... 31
3.5.1. Actores del Sistema ..... 32
3.5.2. Casos de Uso del Sistema ..... 32
3.5.3. Descripción de Casos de Uso del Sistema ..... 34
3.6. Interfaz de usuario ..... 46
3.6.1. Menú Developer Tools ..... 46
3.6.1.1. Menú Open File ..... 47
3.6.1.2. Menú Save Track ..... 48
3.6.1.3. Menú Control Pane ..... 48
**Capítulo 4. Análisis y Diseño del Sistema. ..... 51**
4.1. Diagrama de Clases del Análisis. ..... 51
4.1.1. Diagrama de Clases del Análisis de CU Cargar Mapa BSP ..... 51
4.1.2. Diagrama de Clases del Análisis de CU Cargar Modelos ..... 52
4.1.3. Diagrama de Clases del Análisis de CU Editar Pista de Carreras ..... 52
4.1.4. Diagrama de Clases del Análisis de CU Activar Modo Exploración ..... 52
4.1.5. Diagrama de Clases del Análisis de CU Gestión de Fichero TRK ..... 53
4.2. Diagramas de Colaboración del Análisis ..... 53
4.2.1. Diagramas de Colaboración del Análisis CU Cargar Mapa BSP ..... 54

4.2.2. Diagrama de Colaboración del Análisis para el CU Cargar Modelos.....54
4.2.3. Diagrama de Colaboración del Análisis para el CU Editar Pista de Carreras.....55
4.2.4. Diagrama de Colaboración del Análisis para el CU Activar Modo Exploración...55
4.2.5. Diagrama de Colaboración del Análisis para el CU Gestión de Fichero con Trayectoria de la Pista de Carrera.....56
4.3. Patrones.....56
4.3.1. Strategy.....57
4.3.2. Command.....57
4.3.3. Singleton.....58
4.4. Diagrama de Clases del Diseño del Sistema.....58
4.4.1. Diagrama de Clases del Paquete LOADER.....60
4.4.2. Diagrama de Clases del Paquete TrackEdit.....61
4.5. Diagramas de Secuencia del Diseño.....62
4.5.1. Diagrama de Secuencia del Diseño del CU Cargar Mapa BSP.....62
4.5.2. Diagrama de Secuencia del Diseño del CU Cargar Modelos.....62
4.5.3. Diagrama de Secuencia del Diseño del CU Editar Pista de Carreras.....64
4.6. Descripción de las Clases.....68
**Capítulo 5. Implementación y Resultados.....71**
5.1. Diagrama de Componentes.....71
5.1.1. Componentes del Paquete de Implementación Loader.....72
5.1.2. Componentes del Paquete de Implementación GUI.....72
5.1.3. Componentes del Paquete de Implementación TrackEditor.....73
5.2. Diagrama de Subsistemas.....74
5.3. Diagrama de Despliegue.....74
5.4. Estándares de codificación.....74
5.4.1. Reglas de codificación.....75
5.5. Resultados.....76

**Conclusiones.....80**

**Recomendaciones.....81**

**Bibliografía .....82**

**Glosario de Términos y Siglas.....84**

**Índice de Tablas .....91**

**Índice de Figuras.....92**

# Introducción

Desde el surgimiento de las computadoras el hombre aprovechando sus grandes potencialidades las ha usado en casi todas las esferas de la vida. Una de las industrias que más beneficios ha encontrado en las mismas es la de los videojuegos.

La importancia comercial de este sector con el paso de los años ha experimentado un amplio desarrollo, solo para citar un ejemplo en el año 2004 juegos como Doom3, Half Life2, Grand Theft Auto: San Andreas y Halo, alcanzaron ingresos para la industria del juego de $9.9 billones de dólares **[1].** Esta situación ha dado lugar a una amplia gama de productos derivados, por lo que actualmente existen varias clasificaciones de videojuegos, entre las que se destacan **[2]**:

**Aventura**: Juegos en los que el protagonista debe avanzar en la trama interactuando con diversos personajes y objetos **[2]**. Algunos ejemplos son:

THE LEGEND OF ZOLDA, METAL GEAR y SOUL REAVER.

**Deportivo**: Se basan en deportes, reales o ficticios, y pueden subdividir en simuladores y en "arcade" **[2]** (menos realistas que los primeros). Algunos ejemplos:

FIFA, MVP y NBA LIVE.

**Estrategia**: Se caracterizan por la necesidad de manipular a un numeroso grupo de personajes, objetos o datos para lograr los objetivos. Según su temática los hay de gestión (ya sea esta económica o social) y bélicos, mientras que por su mecánica pueden ser en tiempo real, también llamados "RTS" (Real Time Strategy, siglas en inglés), o por turnos **[2]**. Algunos ejemplos son:

STARTCRAFT, AGE OF EMPIRE, CIVILIZATION y COMMAND & CONQUER.

**Carreras o Velocidad**: Se imitan competencias entre vehículos. Usualmente el objetivo es recorrer cierta distancia o ir de un sitio hacía otro en el menor tiempo posible, como en el automovilismo y el motociclismo **[2]** (los que generalmente son imitados), algunos ejemplos son:

NEED FOR SPEED, MARIO KART, CRASH TEAM RACING y MOTO RACE.

Cuba, centrando su esfuerzo en el desarrollo de la industria del software, como una de las principales tareas de la Batalla de Ideas ha comenzado a dar sus primeros pasos en este campo.

Con el objetivo de insertar a Cuba en el mercado del software a nivel mundial se está potenciando el desarrollo de la industria del desarrollo de software, teniendo un espacio en la misma el desarrollo de videojuegos.

Esta industria se encuentra en estado incipiente, por lo que se necesita de herramientas que potencien la inserción de los productos en el mercado internacional, pero resulta que las herramientas con las que se cuentan a veces no cumplen con los requerimientos necesarios ya sea porque fueron creadas para un videojuego en particular o porque no tienen todas las características que se necesitan y las que existen son de difícil acceso, debido a las licencias y a lo alto de su costo.

Actualmente en el proyecto Juegos Consola de la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI) se está desarrollando el videojuego de carreras de carros Rápido y Curioso, para la edición de los mapas del mismo se utiliza el Editor de Niveles GTKRadiant, el cual es libre, pero no cubre todas las necesidades presentes en este proyecto, debido a que el objetivo de este editor es la creación de mapas para videojuegos de tipo “deathmatch”, como es el caso del Quake y no para los de carreras de carros.

Para complementar esta deficiencia es necesario la utilización de un Editor de Pistas de Carreras, herramienta con la que no se cuenta actualmente en el proyecto Juegos Consola, lo cual retrasa el proceso de edición de las pistas de carreras por parte de los diseñadores, además de que imposibilita que el usuario personalice sus propias pistas de carreras, quedando atado a las pistas que el videojuego propone originalmente.

Enmarcado en esta **Situación Problémica**, el **Problema Científico** reside en la inexistencia de un Editor de Pistas en el proyecto Juegos Consola para la edición de las pistas de carrera de los videojuegos que el mismo produce.

Por lo que el **objeto de Estudio** de esta investigación es el trazado de trayectorias en entornos virtuales, y el **campo de acción** son los grafos de caminos.

Donde se tiene como **objetivo general** obtener un Editor de Pistas de Carreras que permita la edición de las pistas de carreras para los videojuegos producidos por el proyecto Juegos Consola.

EL **objetivo específico** que propone este trabajo es elaborar de una herramienta con una interfaz gráfica de usuario amigable que permita:

- Editar gráficamente el circuito de carrera sobre una base de geometría tridimensional.

- Importar entornos geométricos que sirvan de base para el circuito (BSP, 3DS y MD2).
- Imponer y probar un conjunto de reglas de tráfico.
- Exportar los circuitos a un formato binario propietario.
- Importar los circuitos desde un formato binario propietario.

Para darle cumplimiento a los objetivos de la investigación se han trazado un grupo de **Tareas de la Investigación**, las cuales se pueden resumir en:

1. Definir las características de la interfaz gráfica del Editor de Pistas.
2. Analizar las descripciones de los ficheros a importar (BSP, 3DS y MD2).
3. Definir un formato binario para exportar los circuitos.
4. Implementar un Editor de Pistas de Carrera.

**Métodos científicos**

**Teóricos**

- **Analítico sintético**: Se analizan las teorías, documentos, entre otros, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

- **Inductivo deductivo:** Formas de razonamiento que permiten llegar a un grupo de conocimientos generalizadores, tanto desde el análisis de lo particular a lo general, como desde el análisis de los elementos generalizadores a uno de menor nivel de generalización.

**Empíricos**

- ✓ **Observación**: Registro visual de lo que ocurre en una situación real, en un fenómeno determinado, clasificando y consignando los hechos y acontecimientos pertinentes de acuerdo con algún esquema previsto.

# Organización del documento

El presente trabajo está estructurado de la siguiente manera: resumen, introducción, cinco capítulos de contenido, conclusiones, recomendaciones, bibliografía referenciada y consultada, así como un glosario de términos y siglas.

**Capítulo 1. Fundamentación teórica:** Se realiza un estudio de los videojuegos de carrera, se analizan las diferencias entre un editor de niveles y un editor de pistas de carrera así como la justificación de las herramientas y metodologías usadas.

**Capítulo 2. Características del Sistema:** Muestra los principales rasgos de la herramienta que se propone desarrollar, la cual representa la propuesta de solución al problema planteado.

**Capítulo 3. Descripción de la Solución Propuesta:** Se procede al levantamiento de requisitos del sistema haciendo un análisis más exhaustivo del objeto de estudio que se plantea, además de que se definen los prototipos de la interfaz de usuario.

**Capítulo 4. Análisis y Diseño:** Se realiza el Análisis y Diseño del Sistema, quedando reflejado en Diagramas de Clases y Diagramas de Secuencia de los Casos de Usos recogidos en el capítulo anterior, también se hace un análisis de los patrones de diseño usados.

**Capitulo 5. Implementación y Resultados:** Se procede a la implementación del sistema basado en los resultados del análisis y el diseño elaborado previamente, en este capítulo se exponen además los resultados obtenidos con la puesta en marcha del uso de la herramienta.

# Capítulo 1. Fundamentación Teórica

## 1.1.Videojuegos de Carrera

Desde el momento en que la idea de un videojuego fue concebida y patentada por Thomas T. Goldsmith Jr. y Estle Ray Mann en 1948, estos han experimentado un gran desarrollo, viéndose reflejado en las diversas clasificaciones de los mismos, destacándose por su diversidad los de carreras **[3]**, que se pueden agrupar en:

**Simuladores de carreras**: Estos videojuegos imitan a los vehículos reales, calculan por ejemplo el recorrido físico de la suspensión, el trabajo del motor y la fricción de los neumáticos con la pista.

Algunos juegos tienen licencias oficiales de competencias reales, como la serie Grand Prix, el World Rally Championship y el Fórmula 1. Otros simplemente presentan cientos de vehículos, como el Gran Turismo **[3]**. Varios videojuegos de carreras se especializan en cierto tipo de automóviles, como el Grand Prix Legends (Fórmula 1 de los años 1960), las series Nascar (de Electronic Arts, Papyrus y Hasbro), o el Richard Burns Rally (simulador de rally).

Algunos están especialmente diseñados para jugar online contra rivales de otros países, e incluso pueden ser modificados, como el Live for Speed, el GT Legends (automóviles de gran turismo de los años 1960 y 1970), el GTR (simulador de FIA GT) o el Racer.

**Semi-Simuladores de carreras:** Ciertos juegos sacrifican un poco el realismo para llegar a más jugadores **[3]**, que no siempre esperan perfeccionismo, por ejemplo:

TOCA RACE DRIVER y el COLIN McRae RALLY.

**Juegos de carreras arcade:** Por otra parte, muchos de estos videojuegos se enfocan en pasar un buen rato corriendo una carrera **[3]**. La simulación de los vehículos (aceleración, velocidad, adherencia, choques) y las pistas son irreales o incluso fantásticas. Ejemplos son:

NEED FOR SPEED, PROJECT GOTHAM RACING 3, BURNOT, MARIO KART y OUT RUM.

## 1.2.Editores de Niveles

Los Editores de Niveles, también conocidos como Editores de Mapas o Escenarios, son las herramientas que se usan para diseñar los mapas de los videojuegos, en algunos casos los creadores del videojuego lanzan una versión oficial del Editor de Niveles (Ver Figura 1) del videojuego y este aparece formando parte integral del mismo.

En otras ocasiones, y es la mayoría de las veces, el Editor de Niveles es una parte separada del videojuego debido a que sus creadores no lo liberaron y es entonces cuando la comunidad de fans construye uno para suplir las necesidades, lo cual es la causa de la amplia gama de Editores de Niveles que existen actualmente.

Entre los Editores de Niveles más conocidos se encuentran, aunque es válido decir que no están todos pues la lista se haría interminable:

- GtkRadiant por id Software y Loki Software.
- UnrealEd para las series de este juego.
- StarEdit Editor para el StarCraft.
- Re-Volt Track Editor hecho por PC, PS1, N64 y Dreamcast para el videojuego Re-Volt.
- Warcraft III World Editor (en conjunto con el editor de Warcraft y Warcraft II).

Muchas de estas aplicaciones han servido para editar mapas para otros videojuegos, como es el caso del GTKRadiant **[4]**, el cual exporta mapas en formato .BSP, pero en ocasiones el Editor de Niveles no cumple con las prestaciones que el videojuego necesita, debido a que no fue creado con ese objetivo.

GTKRadiant es un Editor de Niveles para juegos deathmatch (popular tipo de partida al estilo "todos contra todos"), como es el caso del Quake y el DOOM **[4]**, lo que hace evidente que no cuente con la funcionalidad de editar un circuito de carreras debido a que no es un herramienta para este tipo de juegos.

El código fuente del GTKRadiant está disponible en el repositorio Subversion de id Software y las colaboraciones al código están cubiertas bajo licencias de código abierto.

El proyecto Juegos Consola dedicado a la elaboración de un videojuego de carreras de carros utiliza este Editor de Niveles para la construcción de sus mapas, debido a que estos están en

formato BSP, pero para la edición del circuito de carreras se hacía necesario un Editor de Pistas de Carreras.

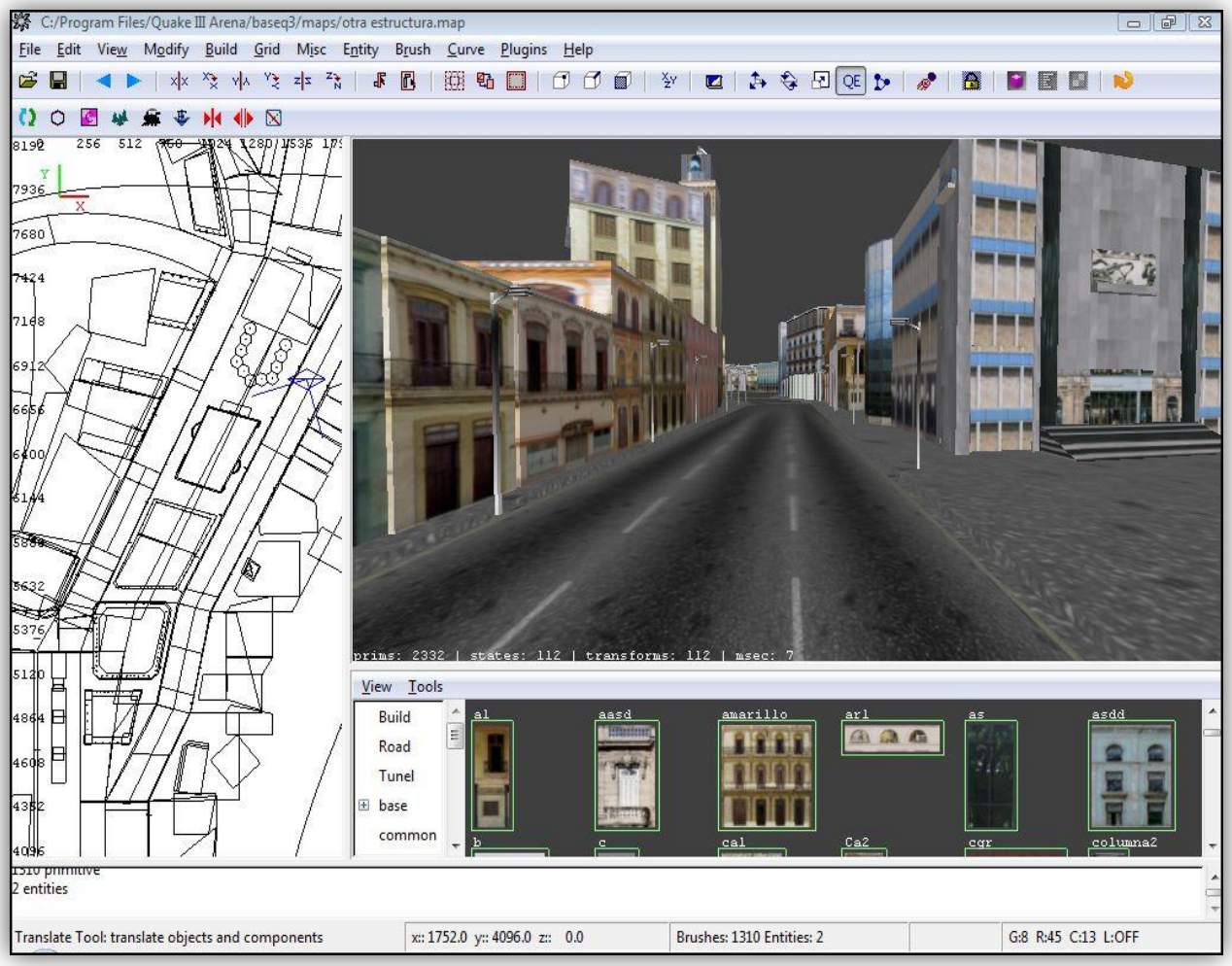


**Figura 1** Editor de Niveles – GtkRadiant.

## 1.3.Editores de Pistas de Carreras

Los videojuegos de carrera tienen en común que hay un camino que recorrer, un sentido que seguir y leyes que respetar, cuando algo de esto es violado por parte del competidor, el sistema lanza un aviso indicando un mensaje de incumplimiento.

Un ejemplo de esto es cuando un jugador recorre un pista en sentido contrario al del curso normal de la carrera, el sistema muestra un mensaje que puede ser desde una flecha que señale el

sentido correcto hasta un mensaje de Trayectoria Incorrecta (Wrong Way, en inglés), como muestra la Figura 2.


**Figura 2** Mensaje de Trayectoria Incorrecta.

Un Editor de Pistas de Carreras (Ver Figura 3 ) es la herramienta utilizada para realizar el trazado del circuito de la pistas de carreras del videojuego en cuestión, mediante la cual el diseñador de pistas de carreras define el recorrido de la pista construyendo de forma manual un grafo de caminos con la trayectoria a seguir por los competidores, también puede definir tipos de terrenos en partes de la pista, variándole parámetros tales como fuerza de rozamiento a los tramos de la pista; de esta forma se consigue simular hielo, arena, entre otros, todo es cuestión de la imaginación del usuario y de las posibilidades que brinde el Editor de Pistas usado.

Este grafo de caminos es guardado en un fichero binario que almacenará el recorrido de la pista, así como otros datos de interés y será encuestado por el sistema en todo momento de la carrera, para cada competidor, para comprobar que no se produzcan violaciones del sentido del circuito previamente definido y también para simular las condiciones físicas de la pista en el tramo en el que se encuentra el corredor.

La edición del circuito de carrera se hace antes de ponerla en uso en el videojuego, por lo que los Editores de Pistas tienen la opción de explorar el circuito de carrera antes de ponerla en uso. La herramienta propuesta presenta dos modos de exploración, un modo manual en el cual el usuario dirige el recorrido por la pista, y en caso de incumplir con el recorrido de la pista el sistema lanzará

un aviso. El otro modo de exploración es el modo automático y es el sistema el que se encarga del recorrido de la pista, lo cual es muy útil ya que permite al usuario visualizar si obtuvo el resultado deseado en el proceso de edición.


**Figura 3** Editor de Pistas de Carreras.

## 1.4.Formatos de los Ficheros Gráficos 3D

Un fichero gráfico 3d debe ser un contenedor de información de una escena, ya sea de un único objeto o de un mundo virtual complejo, que debe contar con todos los atributos que permitan conocer la estructura de la malla, como son los vértices y caras, en el caso más básico y además suelen tener un conjunto de características que brindan un mayor nivel de detalle a las escenas, entre los cuales se puede mencionar, los materiales, los colores de vértices, las normales de las caras, los estados de render, entre otros **[5]**. Otro aspecto importante es el formato de

almacenamiento y la organización de los atributos en el fichero, este último influye en la complejidad, tiempo de carga y facilidad de entendimiento del mismo **[6]**.

Algunos de los formatos binarios 3D más difundidos en la comunidad de Realidad Virtual (RV) para almacenar estas informaciones son los .BSP, .3DS y .MD2 **[6]**.

#### 1.4.1.1.Características del formato BSP

Particionado Binario del Espacio o Binary Space Partitioning (BSP, siglas en inglés) es un método para subdividir recursivamente un espacio en elementos convexos empleando hiperplanos. Esta subdivisión permite obtener una representación de la escena mediante un árbol conocido como Árbol BSP **[7]**.

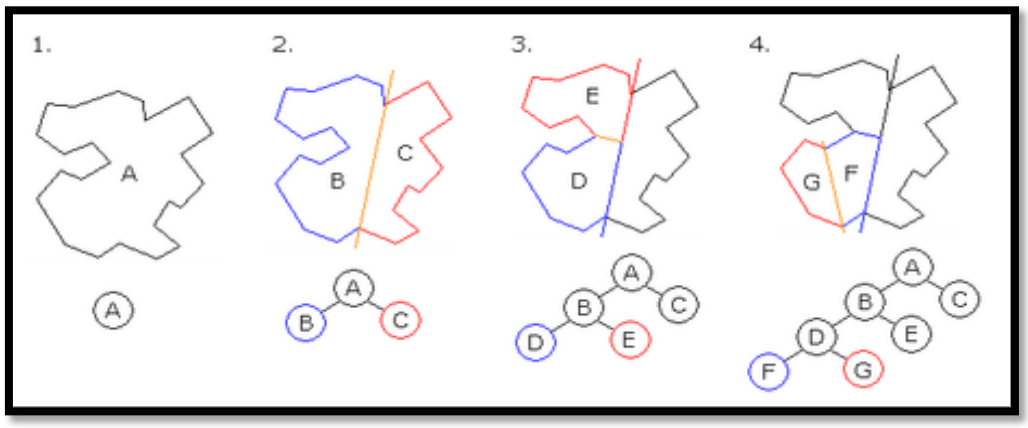


**Figura 4** Creación de un Árbol BSP.

1. A es la raíz del árbol y de todo el polígono.
2. Se divide A en B y C
3. Se divide B en D y E.
4. Se divide D en F y G, que son convexos y se convierten en hojas del árbol **[7]**.

Sus aplicaciones principales en el campo de la RV son incrementar la eficiencia del *render* de las escenas, así como la detección de colisiones.

Los árboles BSP se emplean normalmente en los videojuegos, especialmente en los de acción en primera persona y en los que tienen entornos de interior. Probablemente el primer juego que empleó esta técnica fue el DOOM.

Cada fichero BSP comienza con un header que, a su vez contiene un directorio global de los Lumps presentes en el fichero. Este directorio de Lump identifica la ubicación y el tamaño de los Lumps, los cuales almacenan un tipo particular de los datos del mapa **[14]**.

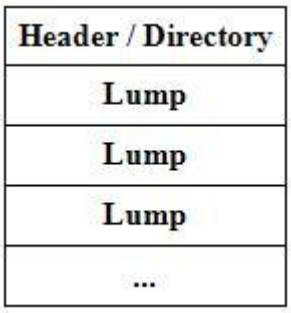


**Figura 5** Estructura del fichero BSP.

Un fichero BSP contiene 17 Lumps, los cuales aparecen en el siguiente orden **[4]**:

1. Entities
2. Textures
3. Planes
4. Nodes
5. Leafs
6. Leaffaces
7. Leafbrushes
8. Models
9. Brushes
10. Brushsides
11. Vertexes
12. Meshverts
13. Effects
14. Faces
15. Lightmaps
16. Lightvols
17. Visdata

### 1.4.1.2.Características del formato 3DS

Creado originalmente para 3D StudioMax por Autodesk. Se encuentra en forma de binario y presenta una estructura muy completa pero su especificación esta patentada bajo una licencia propietaria; por lo que se conoce poco sobre la información almacenada dentro del mismo.

Su estructura está caracterizada por bloques de información indexados conocidos como CHUNCK **[8]** (ver figura 6), la que viabiliza el acceso a los bloque de interés de forma rápida sin necesidad de leer e interpretar los restantes bloques; característica que lo ha hecho muy popular en la comunidad de desarrolladores dentro del campo de la RV **[6]**.

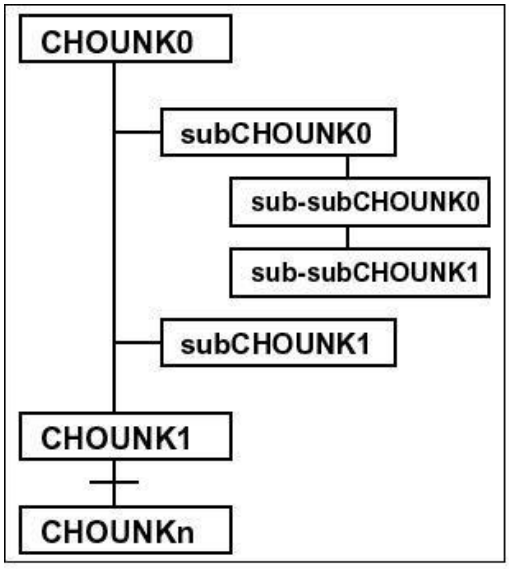


**Figura 6** Estructura del fichero 3DS**.**

### 1.4.1.3.Características del formato MD2

Es un fichero de composición binaria, creado por ID software inc para el juego Quake II **[9]**. Desde el momento que surgió se convirtió en un formato muy popular debido a su facilidad de uso. Los modelos MD2 pueden ser usados para cualquier objeto, tales como: armas, personajes y piezas del terreno **[6]**.

Al presentar una estructura en bloques (ver figura 7), permite que se pueda guardar de manera sencilla, todo el contenido en estructuras de datos. El uso de cabeceras ofrece la ventaja de saber en qué lugar y qué cantidad de un tipo determinado de información contiene el fichero **[9]**.

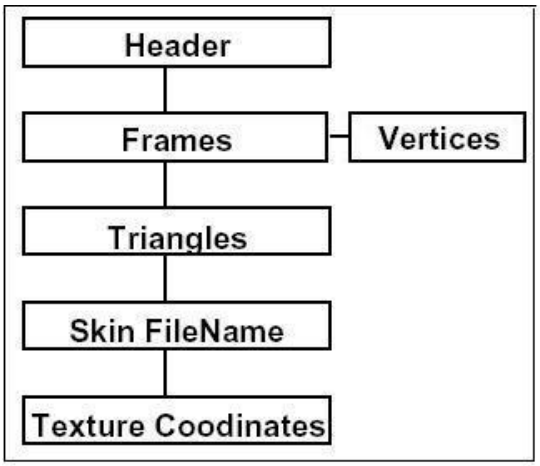


**Figura *7*** Estructura del fichero MD2.

## 1.5.Motores Gráficos

Un Motor Gráfico (Graphic Engine, en inglés), es el componente de software principal de un videojuego o de otra aplicación interactiva que se ejecute en tiempo real. Su uso simplifica el desarrollo de la aplicación y a menudo permite que el juego pueda correr en múltiples plataformas, tales como Consolas de videojuegos y Sistemas Operativos de Mac OS, GNU/Linux y Microsoft Windows. La abstracción del hardware es una de las principales ventajas que posee el Motor Gráfico como se ejemplifica en la Figura 8.

Un Motor Gráfico ofrece un conjunto de herramientas de desarrollo, además de componentes de software reutilizables, agrupados en subsistemas que presentan alta cohesión en relación a sus comportamientos **[10]**, los subsistemas más comunes son:

- Procesamiento de Entradas.
- Gráficos.
- Animación.
- Audio.
- Comportamiento e Inteligencia Artificial.
- Conectividad / Red.

Cada subsistema es un componente por derecho propio, un componente con una estructura particular e independiente, y formado por otros subsistemas más especializados.

Los Motores Gráficos son productos altamente complejos y especializados, capaces de ahorrar tiempo y dinero a través del fomento de la reutilización enfocada a diferentes aspectos del videojuego.

Tener un Motor Gráfico ayuda a ser productivo, a crear juegos sofisticados con menos esfuerzo, y a recuperar y aplicar las mejores prácticas que los expertos de la industria de producción de videojuegos han acumulado a lo largo de los años.

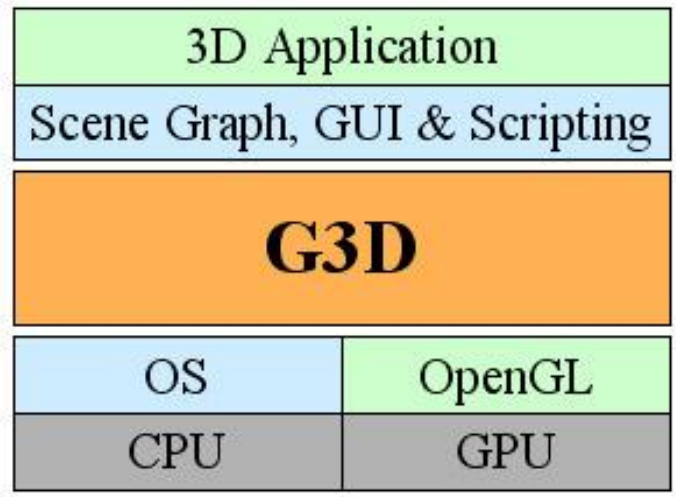


**Figura 8** Arquitectura de G3D Engine.

Algunos de los Motores Gráficos Open Source más conocidos son **[12]:**

1. Crystal Space.
2. Ogre3D.
3. Irrlicht.
4. jMonkeyEngine (jME).
5. G3D Engine.
6. The Nebula Device 2.
7. Realm Force.
8. Blender Game Engine.
9. OpenSceneGraph.
10. Panda3D
11. Reality Factory

## 1.6.G3D Engine

La elección del Engine a usar es de suma importancia, esta decisión repercutirá en la calidad del producto final y en la rapidez con que se termine el producto. Existe una amplia gama de Engines Open Source, destacándose entre los lugares cimeros G3D Engine.

Graphics Three Dimensional (G3D) es un Engine Open Source bajo licencia BSD, es muy usado en juegos comerciales (Ver Figura 9) y aplicaciones para simuladores militares. G3D soporta rendering en tiempo real, off-line rendering como el Raytracing, y propósitos generales en el cómputo del GPU (Graphics Processing Unit, en inglés). G3D proporciona un conjunto de rutinas y estructuras comunes que son necesitadas en todas las aplicaciones gráficas. Garantiza el uso de librerías de bajo nivel como OpenGL y los sockets para la red se pueden usar sin restricciones de las funcionalidades y de rendimiento **[13].** G3D presenta una arquitectura robusta y optimizada, permitiendo integrar otras librerías de forma sencilla y rápida; convirtiéndose en una excelente herramienta para el desarrollo de videojuegos y simuladores **[13].** A continuación se muestra una tabla de las características fundamentales de G3D Engine **[11].**

| **Author** | **Morgan McGuire** | | |
|---|---|---|---|
| **Graphics API** | OpenGL, DirectX, Software, Other | **Operating Systems.** | Windows, Linux. |
| **Programming Language** | C/C++ | **Status** | Alpha. |
| **Documentation** | Yes | | |
| ***General Features*** | Object-Oriented Design: | | |
| ***Physics*** | Collision Detection: | | |
| ***Lighting*** | Volumetric: | | |
| ***Shadows*** | Shadow Mapping, Projected planar, Shadow Volume: | | |
| ***Texturing*** | Basic, Bumpmapping. | | |

| | |
|---|---|
| ***Shaders*** | Vertex, Pixel. |
| ***Scene Management*** | BSP, LOD |
| ***Animation*** | Keyframe Animation. |
| ***Meshes*** | Mesh Loading: |
| ***Special Effects*** | Environment Mapping, Lens Flares, Billboarding, Particle System, Sky, Water, Fire, Explosion, Fog, Mirror: |
| ***Terrain*** | Rendering: |
| ***Networking System*** | Client-Server: |
| ***Artificial Intelligence*** | Scripted: |
| ***Rendering*** | Fixed-function, Render-to-Texture: |

**Tabla 1** Características de G3D Engine.


**Figura 9** Animal Race: Videojuego producido por la Universidad de Ulm, usando G3D, ODE, y Animadead.

## 1.7.Metodologías y Herramientas de Desarrollo

La existencia de una serie de aplicaciones posibles para llevar a cabo la realización de la herramienta propuesta hizo que fuera necesario hacer una selección de las más óptimas para ello. Para seleccionarlas se utilizaron una serie de parámetros como ventaja, tendencia actual, dominio y fortaleza.

### 1.7.1.Metodología

La metodología aplicada fue Rational Unified Process (RUP, siglas en inglés) que es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado, UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas Orientados a Objetos.

Sin embargo, hay tres características fundamentales que lo hacen una metodología robusta y poderosa: es dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

Cuando se caracteriza RUP como dirigido por casos de uso se refiere a que los casos de uso dirigen todo el proceso del desarrollo del software ya que estos reflejan lo que los clientes necesitan y desean, constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo del sistema.

Centrado en la arquitectura se refiere a que se incluye los aspectos estáticos y dinámicos más significativos del sistema. Además recoge una serie de factores como la plataforma en la cual funciona el software, los bloques de construcción reutilizables que se tienen, consideraciones de implementación, sistemas heredados y requisitos no funcionales.

Es iterativo e incremental porque el proyecto o el desarrollo de una aplicación se pueden dividir en partes y desarrollarlas de manera iterativa, incrementándose a medida que se integran unas con otras hasta llegar a formar la tarea final. Las iteraciones hacen referencia a pasos en el flujo de trabajo y los incrementos, el crecimiento del producto.

RUP deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

## 1.7.2.Herramientas de Desarrollo

Las herramientas de desarrollo recogen el software usado en el desarrollo de la aplicación y en el modelado de los casos de uso. En el desarrollo de la herramienta propuesta se usó el Visual Studio 2005 y para crear la documentación de esta, el Rational Rose 2003.

### 1.7.2.1.Visual Studio 2005

El Visual Studio 2005, es una herramienta poderosa, fuerte y voluminosa que tiene una gran integración de varios lenguajes entre ellos el C++, C#, y J#. Tiene la posibilidad de implementar aplicaciones para soluciones integrales que aprovechen de manera óptima la ventaja de cada lenguaje. Acelera de manera significativa la producción de software. La documentación del Visual Studio está entre las mejores. La interfaz es altamente amigable con el usuario permitiendo que el tiempo en implementar una solución o aplicación determinada sea mucho menor. Los ejecutables desarrollados por esta herramienta son generalmente de menor tamaño lo que hace que ocupe un lugar cimero en la producción de software al lograr aplicaciones óptimas y de poco volumen. Otro de los motivos por los cuales se usa esta herramienta para desarrollar el software propuesto es que esta es la plataforma de desarrollo usada en el proyecto Juegos Consola para el desarrollo de los productos que el mismo produce.

### 1.7.2.2.Rational Rose 2003

Es una de las más poderosas herramientas de modelado visual para el análisis y diseño de sistemas basados en objetos. Rational Rose Enterprise es la mejor elección para el ambiente de modelado que soporte la generación de código a partir de modelos en Ada, ANSI C++, C++, CORBA, Java™/J2EE™, Visual C++® y Visual Basic®. Como todos los demás productos Rational Rose, proporciona un lenguaje común de modelado (Unified Modeling Language, UML siglas en inglés) para el equipo que facilita la creación de software de calidad más rápidamente.

**Características adicionales incluidas**:

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "Design Patterns: Elements of Reusable Object-Oriented Software".
- Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos.
- Soporte de ingeniería Forward y/o reversa para algunos de los conceptos más comunes de Java 1.5.

- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo-código configurables.
- Soporte Enterprise Java Beans™ 2.0.
- Capacidad de análisis de calidad de código.
- El Add-In para modelado Web provee visualización, modelado y las herramientas para desarrollar aplicaciones de Web.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Capacidad de crear definiciones de tipo de documento XML (DTD) para el uso en la aplicación.
- Integración con otras herramientas de desarrollo de Rational.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo.

## 1.7.3.Lenguajes

Para el desarrollo de la herramienta se escogieron dos lenguajes fundamentales, el C++ y el UML. El primero para realizar la implementación de la aplicación y el otro para modelar los artefactos de software en el proceso de desarrollo del mismo.

### 1.7.3.1.Lenguaje de Programación

El C++ se escogió por ser un lenguaje de programación estandarizado por la norma ISO/IEC 14882:1998. Entre sus principales características está el soporte para la programación orientada a objetos y el soporte de plantillas o programación genérica, además de ser un lenguaje de alto nivel que está considerado como un lenguaje potente al poder trabajar tanto en bajo como en alto nivel. Por lo que es el lenguaje idóneo para las aplicaciones de RV y videojuegos.

Posee dos propiedades fundamentales que son difíciles de encontrar en otros lenguajes que son la posibilidad de redefinir los operadores, comúnmente conocido como la sobrecarga de operadores, y la identificación de tipos en tiempo de ejecución. Además presenta una biblioteca estándar muy poderosa.

### 1.7.3.2.Lenguaje de Modelado

UML se escogió para realizar el modelado del análisis y el diseño de la aplicación debido a que:

- El UML ofrece un modo estándar de visualizar, especificar, construir, documentar y comunicar los artefactos de un sistema basado en el software que debe usarse en el proceso completo del desarrollo del mismo.
- Permite modelar sistemas utilizando técnicas orientadas a objetos (OO).
- Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, entre otros).

UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

# Capítulo 2. Solución Propuesta

El recorrido correcto de la pista por parte del corredor se hace siguiendo un Grafo de Caminos que pudo haber sido guardado previamente en un fichero que también almacena otros datos de interés, por ejemplo características del terreno en un tramo determinado. Los Grafos de Caminos están formados por los nodos y las aristas que los unen entre sí. Los nodos se ubican de forma tridimensional.

Los Grafos de Caminos son usados en aplicaciones como Videojuegos, Entornos Virtuales y Simuladores, las cuales realizan un análisis de los diferentes nodos para obtener datos o valores como el camino más largo o el más corto a un nodo, así como lugares favorables para situar emboscadas entre otras aplicaciones.

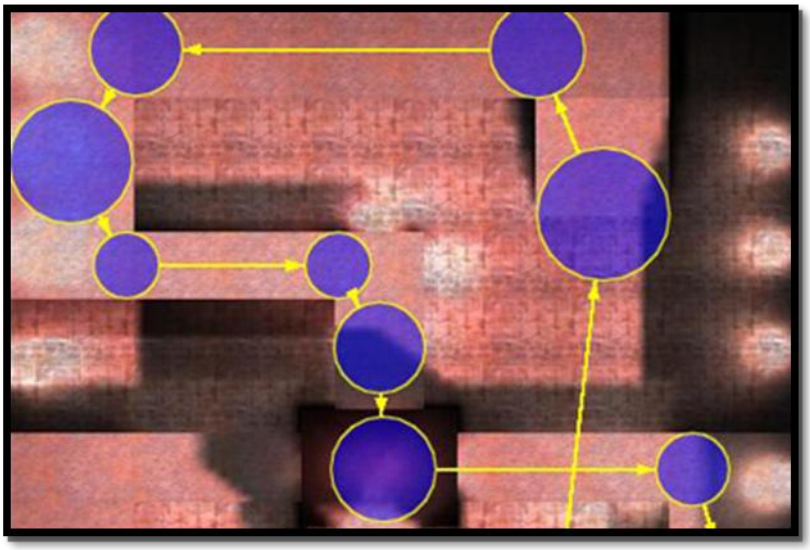

**Figura 10** Proceso de edición de una pista de carreras.

## 2.1.El Sistema

Para la edición de los Grafos de Caminos los usuarios se apoyan en aplicaciones especializadas, conocidas como Editores de Pistas de Carrera.

Estas herramientas tienen como entrada un fichero con la geometría del mundo a cargar para definirle el circuito de la carrera y luego exportar un fichero con el grafo de caminos y otras

informaciones de la pista, para luego ser usado por el videojuego u otra aplicación, ver este proceso en la Figura 11.

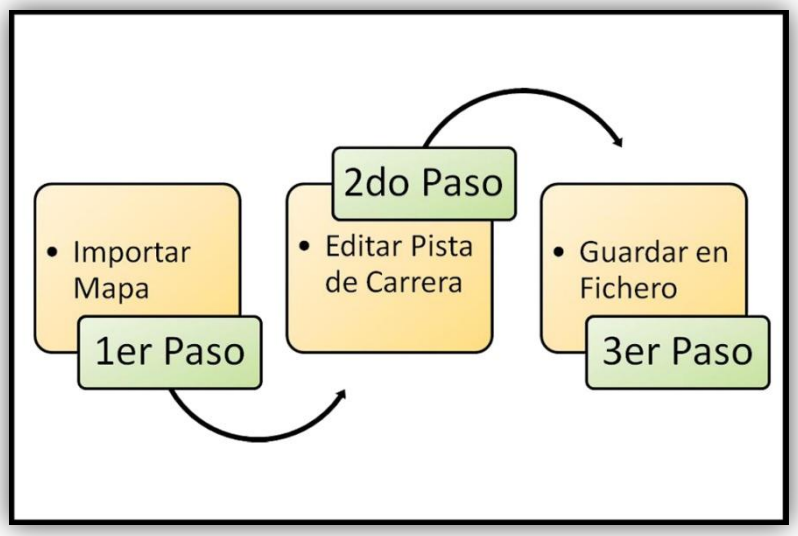


**Figura 11** Dinámica de un Editor de Pistas de Carreras.

## 2.2.Grafos de Caminos

Un grafo es un objeto matemático que se utiliza para representar circuitos, redes, caminos, entre otras. Los grafos son muy utilizados en computación, ya que permiten resolver problemas muy complejos.

Imaginen que se dispone de una serie de ciudades y de carreteras que las unen. De cada ciudad saldrán varias carreteras, por lo que para ir de una ciudad a otra se podrán tomar diversos caminos. Cada carretera tendrá un coste asociado que puede ser representado mediante la longitud de la misma. Mediante la representación por grafos (Ver Figura 12) se puede elegir el camino más corto que conecta dos ciudades, determinar si es posible llegar de una ciudad a otra, si desde cualquier ciudad existe un camino que llegue a cualquier otra, entre otras aplicaciones.

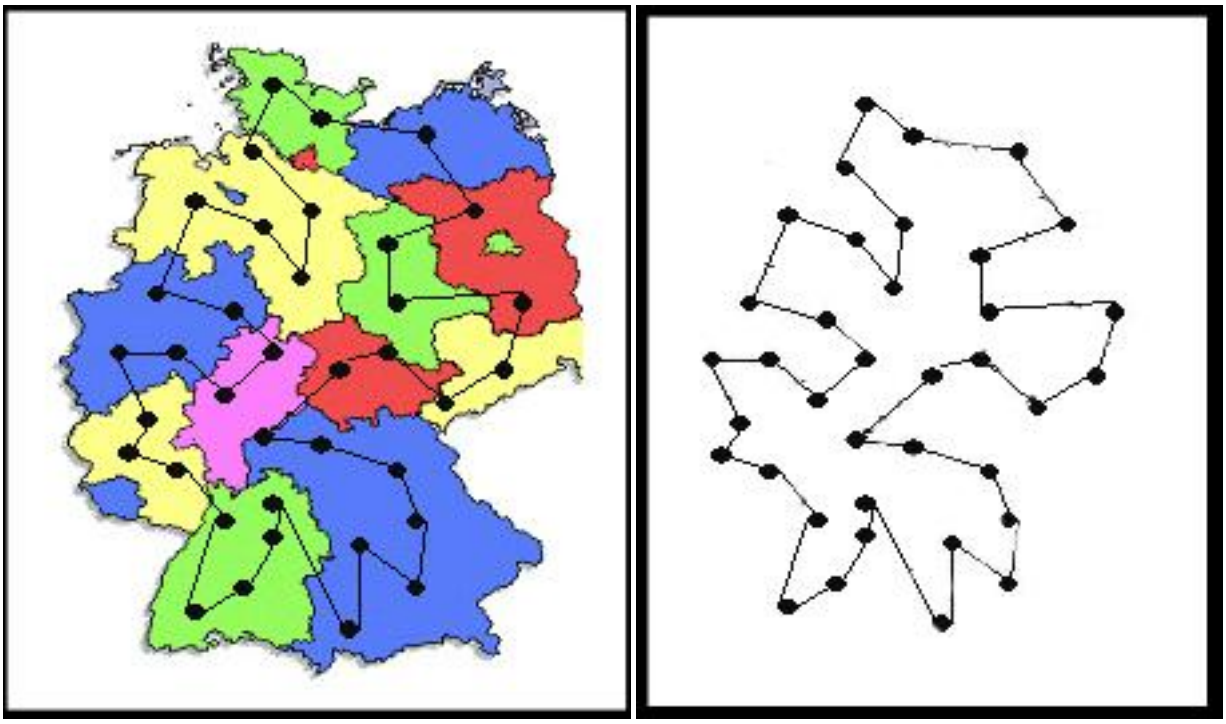

**Figura 12** Grafo de Caminos de una región.

## 2.3.Especificación del formato de fichero a exportar

Una vez terminado el proceso de edición y prueba de la pista de carrera es necesario contar con un mecanismo de persistencia que permita utilizar posteriormente el grafo de caminos y demás características de la pista editada. Para almacenar estos datos se creó un fichero binario, ya que es la mejor opción en estos tipos de aplicaciones por las potencialidades de accesibilidad a la información, por tener mayor confidencialidad en los datos y contar con mayor velocidad de carga que su contraparte en texto plano. Este fichero tendrá extensión ***.TRK.**

### 2.3.1.Estructura del fichero TRK

Siguiendo estándares de formatos de ficheros 3D usados en el campo de la Realidad Virtual como son los ficheros 3DS o MD2, el fichero .trk tendrá una estructura compuesta por bloques de datos precedidos por un *header* el cual especifica el tipo de bloque que se leerá como se muestra en la Figura 13, quedando la estructura del fichero trk como se muestra a continuación:

**[Nombre del mapa asociado] [Header] [Bloque de Datos] [Header] [Bloque de Datos].**

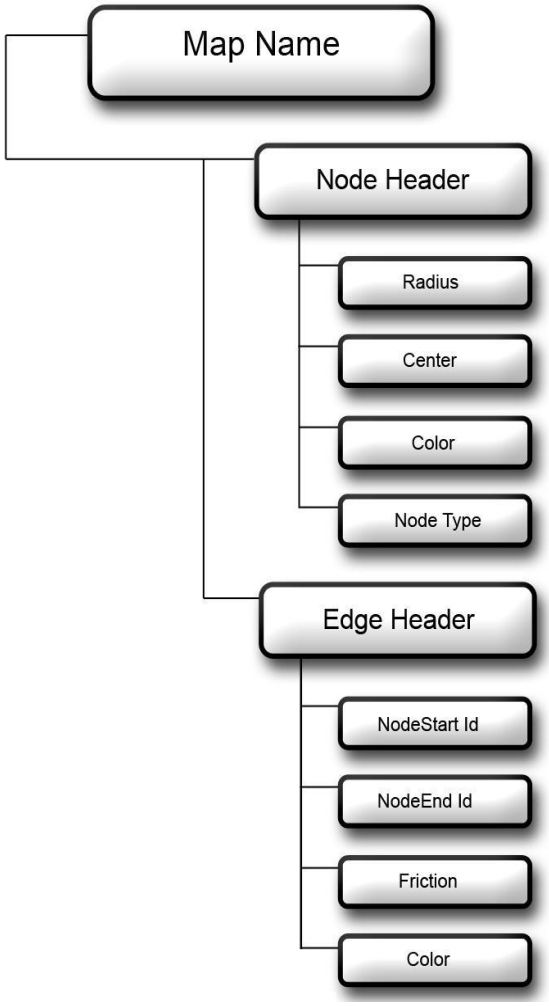


**Figura 13** Estructura del fichero TRK.

### 2.3.1.1.Nombre del Mapa Asociado

Es una etiqueta que contiene el nombre del mapa sobre el cual se editó el grafo de caminos, lo cual asegura que este grafo solo sea aplicable sobre el mapa para el cual fue editado.

### 2.3.1.2.Header

El Header es un contenedor de informaciones generales que describen aspectos básicos del bloque al cual precede; en él se almacena la información referente al tipo de dato del bloque que vendrá a continuación y la cantidad de datos de ese tipo que contiene ese bloque.

### 2.3.1.3.Bloque de Datos

Constituye el bloque principal de información del fichero ya que contiene la información real de los datos almacenados, los cuales pueden ser de los tipos que se reflejan en la siguiente tabla:

| Tipo de Dato | Descripción | Representación en Fichero |
|---|---|---|
| **INT** | **Entero de 32 bits** | **4** |
| **STRING** | **Cadena de Caracteres** | **"Esto es una cadena"** |
| **VECTOR3** | **Vector 3D de valores enteros** | **1 2 3** |
| **FLOAT** | **Decimal de 32 bits** | **10.0** |

**Tabla 2** Tipos de dato en un archivo *TRK.

El fichero estará compuesto por dos bloques de datos, Nodes y Edges, los cuales se especifican a continuación:

| Atributo | Tipo de dato | Descripción |
|---|---|---|
| **Radius** | **float** | **Representa el radio del nodo.** |
| **Center** | **Vector3** | **Posición del centro del nodo.** |
| **Color** | **Color3** | **Color del nodo.** |
| **Type** | **int** | **Tipo de nodo.** |

**Tabla 3** Estructura del Bloque Nodes del fichero TRK.

| Atributo | Tipo de dato | Descripción |
|---|---|---|
| **NodeStartId** | **int** | **Id del nodo inicial.** |
| **EndNodeIdx** | **Vector3** | **Id del nodo final.** |
| **Friction** | **Float** | **Coeficiente de fricción de la arista.** |
| **Color** | **Color3** | **Color de la arista.** |

**Tabla 4** Estructura del Bloque Edges del fichero TRK.

# Capítulo 3. Descripción de la Solución Propuesta

En este capítulo se hará una descripción a nivel conceptual de la solución propuesta en el capítulo anterior. Se definen las reglas a cumplir para asegurar el correcto funcionamiento del sistema, relacionadas como reglas del negocio. Se tratarán los conceptos más importantes del área de interés mediante un modelo de dominio. Además serán descritas las capacidades que deberá cumplir el sistema en forma de requisitos funcionales y no funcionales y por último se hace un análisis del prototipo de interfaz de usuario.

## 3.1.Reglas del Negocio

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio, para asegurar el buen funcionamiento de la herramienta se han definido las siguientes reglas:

- ✓ Los ficheros a cargar con la geometría de los mapas deben estar en formato .BSP y cumplir con todas las especificaciones de este formato.
- ✓ Los formatos de los modelos que se cargan deben estar en formato .3DS o .MD2 y cumplir con las especificaciones de los mismos.
- ✓ El archivo trk que se carga debe cumplir con las especificaciones que proponen los autores.

## 3.2.Modelo de Domino

El modelo de dominio representa un acercamiento a la solución propuesta, donde se modelan los principales conceptos implicados en el desarrollo de la solución, así como las relaciones existentes entre ellos.

A continuación se representa el modelo del dominio referente a la herramienta propuesta.

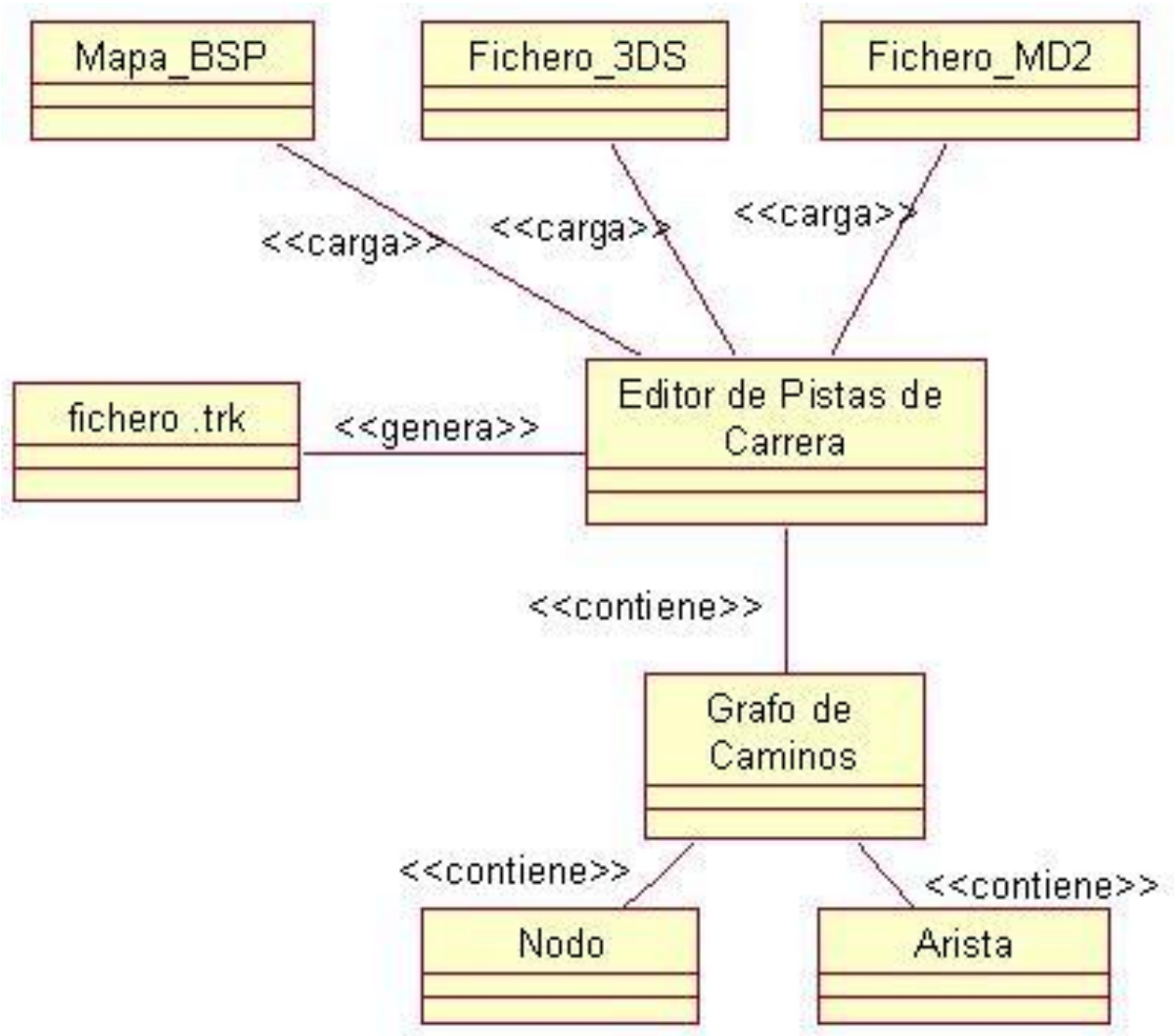


**Figura 14** Modelo de Dominio.

## 3.3.Conceptos del Modelo de Dominio

Con el objetivo de facilitar la comprensión del modelo anterior, a continuación se relacionan los conceptos con sus respectivas descripciones.

**Nodo:** Contiene la información de los nodos del Grafo de Caminos de la pista de carrera, actuando como los puntos de control del mismo.

**Arista:** Es la unión entre dos **Nodos**, contiene información del camino en ese tramo, por ejemplo el coeficiente de rozamiento, con lo que se puede simular tipos de terreno (hielo, arena, entre otros).

**Grafo de Caminos:** Contiene la información correspondiente al grafo de caminos de la pista de carreras, está conformado por una lista de **Nodos** y una lista de **Aristas.**

**Editor de Pistas de Carreras:** Es la unidad controladora del sistema, contiene el grafo de caminos con sus características, así como el control de los ficheros a cargar, los cuales pueden ser bsp, 3ds, md2 y trk, este último es generado por el editor de pistas, siendo el producto final de la aplicación. El mismo consta con 2 modos, un modo de edición para la creación de la pista de carrera y un modo de prueba de lo editado.

**Mapa_BSP:** Fichero que contiene el mapa de la pista a editar, estos mapas estarán en formato .BSP.

**Fichero_MD2:** Fichero que contiene la información referente a un modelo .MD2.

**Fichero_3DS:** Fichero que contiene la información referente a un modelo .3DS.

**Fichero .TRK:** Fichero donde se almacena la pista de carrera editada, es el fichero utilizado por terceros y solo puede ser obtenido mediante el proceso salva de la pista de carrera editada previamente.

## 3.4.Captura de Requisitos

La IEEE Standard Glossary of Software Engineering Terminology define un requerimiento como condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo, la cual tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

Los Requerimientos son un acuerdo que se firman entre el cliente y los desarrolladores del sistema, donde quedan reflejados las cualidades y capacidades que tendrá el mismo.

### 3.4.1.Requisitos Funcionales

Son capacidades o condiciones que el sistema debe cumplir. Los requerimientos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen, los Requerimientos Funcionales de la herramienta a desarrollar se muestran a continuación:

**R1. Cargar Mapa BSP.**

R1.1. Definir mapa a cargar.

R1.1.1. Cargar mapa en formato .BSP.

**R2. Cargar Modelos.**

R2.1 Seleccionar Modelo a Cargar.

R2.1.1 Cargar Archivo en formato MD2.

R2.1.2 Cargar Archivo en formato .3DS.

**R3. Editar Pista de Carreras.**

R3.1 El usuario podrá realizar el proceso de edición hasta que obtenga los resultados deseados antes de pasar al proceso de Exploración de lo realizado.

R3.1.1 Edición de Nodos.

R3.1.2 Edición de Aristas.

**R4. Activar Modo Exploración.**

R4.1. El proceso de exploración contará con un modo automático el cual realiza un recorrido por el camino definido por el usuario.

R4.2. El proceso de exploración contará con un modo manual.

R4.2.1 Si el usuario incumple con el recorrido definido, el sistema lanza un aviso de incumplimiento del mismo.

**R5. Gestionar Fichero con Trayectoria de la Pista de Carrera.**

*R5.1.* Se exportará toda la información generada durante el proceso de edición de la pista de carrera en un fichero binario, de extensión *.TRK.

*R5.2.* Se cargará toda la información guardada en un fichero binario, de extensión *.TRK.

## 3.4.2.Requisitos No Funcionales

Representan las características, cualidades y limitaciones del producto, los Requisitos no Funcionales de la aplicación son los que se relacionan a continuación.

**Usabilidad**

- ✓ Cualquier usuario que quiera editar una pista de carreras sobre un mapa BSP.

**Portabilidad**

- ✓ Debe ser portable a cualquier Sistema Operativo Windows, familia de Unix o Mac OS.

**Requerimientos de Software**

- ✓ Sistema Operativo Windows XP o superior.

**Requerimientos de Hardware**

- ✓ Microprocesador Intel Pentium 3 o superior.
- ✓ Memoria RAM de 256 MB o superior.
- ✓ Tarjeta de Video 64 MB o superior.

**Restricciones en el Diseño e Implementación**

- ✓ Lenguaje de programación C++ bajo el paradigma de programación Orientado a Objeto.

**Diseño e Implementación**

- ✓ Se debe desarrollar utilizando el Entorno de Desarrollo Integrado (del inglés Integrated Development Environment (IDE)) VisualStudio.Net 2005.
- ✓ Se debe desarrollar utilizando la biblioteca gráfica G3D v7.0.

## 3.5.Modelo de Casos de Uso del Sistema

Los Casos de Uso se han convertido en la técnica más utilizada a nivel mundial para el levantamiento y la comunicación clara y eficiente de los requisitos, también conocidos como requerimientos, para el desarrollo de sistemas.

Asimismo, los casos de uso son un artefacto clave en el Proceso Unificado de Desarrollo de Software, ya que son el depósito principal de los requisitos funcionales que gobiernan el diseño, la construcción, las pruebas, y muchos otros aspectos de este proceso.

### 3.5.1.Actores del Sistema

Los actores son los roles que un usuario o usuarios del sistema llevan a cabo en algún momento del tiempo. También pueden ser otros sistemas con los que el sistema interactúa. Estos estimulan al sistema con eventos de entrada o la recepción de algún resultado que este produzca.

En este caso particular quien hará uso del sistema será un diseñador de pistas de carreras, el cual constituye el actor del sistema, específicamente será llamado diseñador de pistas de carreras, como se muestra en la tabla 3.

| Actores | Justificación |
|---|---|
| Diseñador de pistas de carreras. | Persona que trabaja e interactúa con la Herramienta. |

**Tabla 5** Actor del Sistema.

### 3.5.2.Casos de Uso del Sistema

| | |
|---|---|
| **CU1** | Cargar Mapa BSP. |
| **Actor** | Diseñador de pistas de carreras. |
| **Descripción:** | Su propósito es cargar el mapa en formato .BSP sobre el cual editar la pista de carreras. |
| **Referencia:** | R1, R1.1, R1.1.1 |

**Tabla 6** CU1 Cargar Mapa BSP.

| | |
|---|---|
| **CU2** | Cargar Modelos. |
| **Actor** | Diseñador de pistas de carreras. |
| **Descripción:** | Su propósito es cargar un modelo MD2 o 3DS para ganar en claridad en cuanto al realismo del circuito que se editará. |
| **Referencia:** | R2, R2.1, R2.1.1, R2.1.2 |

**Tabla 7** CU2 Cargar Modelos.

| | |
|---|---|
| **CU3** | Editar Pista de Carreras. |
| **Actor** | Diseñador de pistas de carreras. |
| **Descripción:** | Su propósito es la edición de la pista de carrera. |
| **Referencia:** | R3, R3.1, R3.1.1, R3.1.1.2 |

**Tabla 8** CU3 Editar Pista de Carreras.

| | |
|---|---|
| **CU4** | Activar Modo Exploración. |
| **Actor** | Diseñador de pistas de carreras. |
| **Descripción:** | Se activa el modo Exploración de la herramienta para verificar que el camino editado es el deseado. |
| **Referencia:** | R4, R4.1, R4.2, R4.2.1 |

**Tabla 9** CU4 Activar Modo Exploración.

| | |
|---|---|
| **CU5** | Gestionar Fichero con Trayectoria de la Pista de Carrera. |
| **Actor** | Diseñador de pistas de carreras. |
| **Descripción:** | Se Guarda o se Carga fichero .trk con la información de la pista de carrera editada previamente. |
| **Referencia:** | R5, R5.1, R5.2 |

**Tabla 10** CU5 Gestionar Fichero con Trayectoria de la Pista de Carrera.

**Diagrama de Casos de Uso del Sistema.**

El diagrama siguiente representa la relación entre el Actor y los Casos de Uso del Sistema.

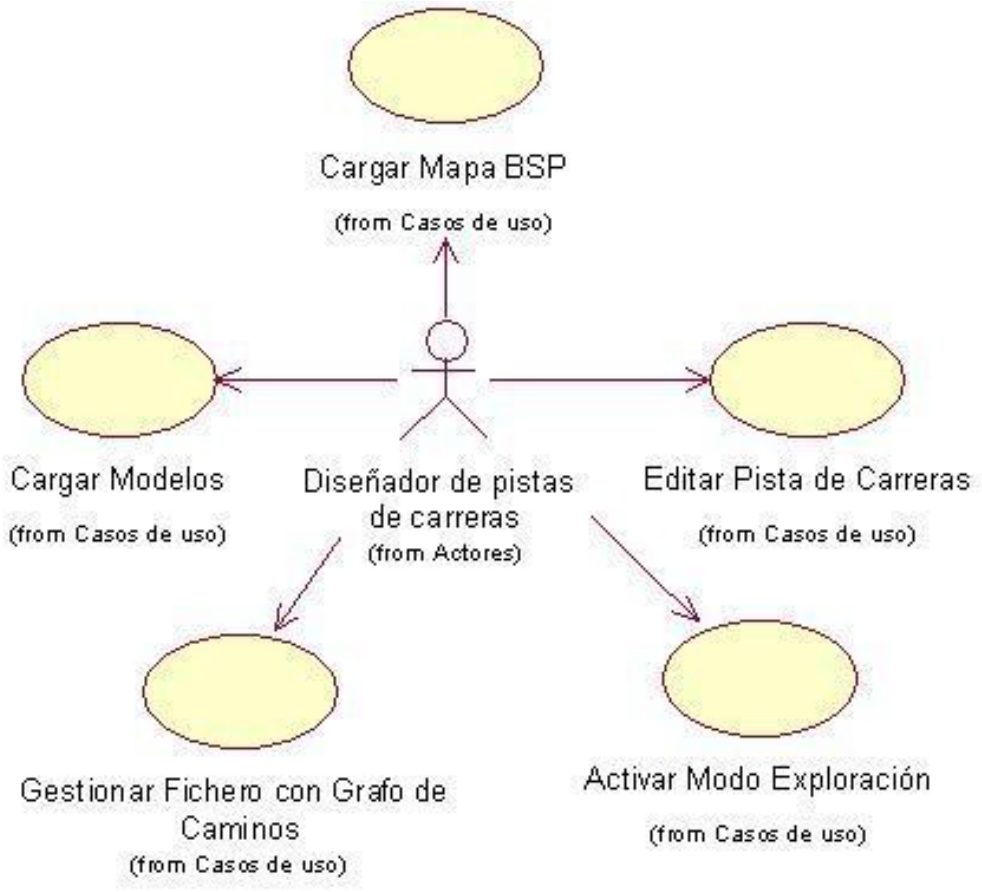


**Figura 15** Diagrama de Casos de Uso del Sistema.

## 3.5.3.Descripción de Casos de Uso del Sistema

Cada caso de uso tiene una descripción que detalla la funcionalidad que se implementará en el sistema propuesto, las tablas presentadas a continuación forman parte de este formato expandido, donde se argumentan con mayor profundidad los flujos operacionales de cada caso de uso.

| **Caso de Uso:** | Cargar Mapa BSP. |
|---|---|
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Cargar el Entorno Virtual para Editar la Pista de Carrera. |
| **Resumen:** | El caso de uso se inicia cuando el Diseñador de pistas de carreras elige en el menú Developer Tools la opción Open File y carga un mapa BSP para editar una pista de carrera en el mismo. El sistema procede a cargar el fichero seleccionado. |
| **Referencia:** | R1, R1.1, R1.1.1. |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El Diseñador de pistas de carreras escoge la opción Open File en el menú Developer Tools. | 2. Se muestra el cuadro de diálogo Open File que le permitirá al Diseñador de pistas de carreras especificar la dirección y el nombre del mapa BSP a cargar. |
| 3. Escribe la dirección y el nombre del fichero a cargar y acciona el botón OK. | 4. Se cierra el cuadro de diálogo y se procede a cargar el fichero BSP especificado por el Diseñador de pistas de carreras. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1.1 El Diseñador de pistas de carreras hace uso de la funcionalidad Drag & Drop de la aplicación y arrastra para la ventana de la aplicación el mapa BSP a cargar. | 5. Procede a la carga del fichero BSP “arrastrado” por el Diseñador de pistas de carreras. |
| 3.1 Selecciona la opción de cancelar en el cuadro de diálogo Open File. | 8. Se cierra el cuadro de diálogo y se termina el caso de uso. |

| **Prioridad:** | Crítica. |
|---|---|

**Tabla 11** Descripción CU Cargar Mapa BSP.

| **Caso de Uso:** | Cargar Modelos | |
|---|---|---|
| **Actores:** | Diseñador de pistas de carreras. | |
| **Propósito:** | Cargar un modelo MD2 o un modelo 3DS para facilitar el proceso de edición de la pista de carreras. | |
| **Resumen:** | El caso de uso se inicia cuando el usuario elige en el menú Developer Tools la opción Open File y especifica un modelo MD2 o 3DS a cargar. El sistema procede a cargar del modelo seleccionado. | |
| **Referencia:** | R2, R2.1, R2.1.1, R2.1.2 | |
| **Precondición:** | Debe haberse realizado el CU Cargar Mapa BSP. | |
| **Flujo Normal de Eventos** | | |
| **Acción del Actor** | | **Respuesta del Sistema** |
| 1. Selecciona la opción Open File. | | 2. Se muestra el cuadro de diálogo Open File que le permitirá al Diseñador de pistas de carreras especificar la dirección y el nombre del modelo a cargar. |
| **Flujos Alternos** | | |
| 1.1 Selecciona un modelo MD2. | | 1.2. Ir a la Sección Cargar Modelo MD2. |
| 1.3 Selecciona un modelo 3DS. | | 1.4. Ir a la Sección Cargar Modelo 3DS. |
| **Postcondiciones:** | Aparecerá un modelo en el mapa. | |
| **Prioridad:** | Crítica. | |

**Tabla 12** Descripción CU Cargar Modelos.

| | |
|---|---|
| **Caso de Uso:** | Editar Pista de Carreras. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Editar la Pista de carreras. |
| **Resumen:** | El caso de uso se inicia cuando el Diseñador de pistas de carreras escoge la opción de editar la pista de carreras. |
| **Referencia:** | R3, R3.1, R3.1.1, R3.1.2 |
| **Precondición:** | Realización del CU Cargar Mapa BSP. |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona la opción de editar la pista de carreras. | 2. Muestra el menú Control Pane que contiene los elementos necesarios para editar la pista de carrera. |
| **Flujos Alternos** | |
| 1.1 No modifica la pista de carrera. | 3. no se realiza ningún proceso y se termina el caso de uso. |
| 4. Crea un nodo. | 5. Ir a la sección crear nodo. |
| 6. Crea una arista. | 7. Ir a la sección crear arista. |
| 8. Modificar información de un nodo. | 9. Ir a la sección modificar información de un nodo. |
| 10. Modificar información de una arista. | 11. Ir a la sección modificar información de una arista. |
| 12. Eliminar nodo. | 13. Ir a la sección eliminar nodo. |
| 14. Eliminar arista. | 15. Ir a la sección eliminar arista. |

| | |
|---|---|
| **Postcondiciones:** | Se visualizará en la pantalla el resultado obtenido del proceso de edición de la pista de carrera. |
| **Prioridad:** | Crítica. |

| | |
|---|---|
| **Sección:** | Crear Nodo. |
| **C U al que pertenece:** | Editar Pista de Carreras. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Su propósito es crear un nodo en la pista de carreras. |
| **Resumen:** | La sección se inicia cuando el Diseñador de pistas de carreras oprime la “barra espaciadora” y crea un nodo. |
| **Referencia:** | R3, R3.1, R3.1.1, R3.1.2 |
| **Precondición** | Debe haberse realizado el CU Cargar Mapa BSP. |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Oprime la barra espaciadora. | 2. Crea un nodo con la información especificada previamente por el usuario y en la posición en la que se encuentra el modelo cargado. |
| | 3. Se visualizará el nodo creado y se da fin al caso de uso. |

| | |
|---|---|
| **Postcondiciones:** | Se visualizará en la pantalla el nodo creado sobre el mapa BSP. |
| **Prioridad:** | Crítica. |
| **Sección:** | Crear Arista. |
| **C U al que pertenece:** | Editar Pista de Carreras. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Su propósito es crear una arista en la pista de carreras. |
| **Resumen:** | La sección se inicia cuando el Diseñador de pistas de carreras define el nodo inicial y el nodo final de la arista a crear. |
| **Referencia:** | R3, R3.1, R3.1.1, R3.1.2 |
| **Precondición** | Deben haberse seleccionado dos nodos válidos para crear una arista. |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona los dos nodos desde los cuales crear la(s) arista. | 2. Crea una arista con la información especificada previamente por el usuario. |
| | 3. Se visualizará la arista creada y se da final caso de uso. |

| | |
|---|---|
| **Postcondiciones:** | Se visualizará en la pantalla la arista creada entre los nodos seleccionados. |
| **Prioridad:** | Crítica. |
| **Sección:** | Modificar Información de un Nodo. |
| **C U al que pertenece:** | Editar Pista de Carreras. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Su propósito es modificar la información de un nodo. |
| **Resumen:** | La sección se inicia cuando el Diseñador de pistas de carreras selecciona un nodo(s) y modifica su información. |
| **Referencia:** | R3, R3.1, R3.1.1, R3.1.2 |
| **Precondición** | Deben haberse seleccionado un nodo. |
| **Flujo Normal de Eventos** | |

| **Acción del Actor** | **Respuesta del Sistema** |
|---|---|
| 1. Selecciona el nodo(s) al cual modificar su información. | 2. Muestra la información del nodo seleccionado. |
| 3. Cambia la información referente al nodo seleccionado. | 4. Actualiza la información del nodo entrada por el Diseñador de Pistas de carreras. |
| | 5. Se visualizará el nodo con su información actual y termina el caso de uso. |

| | |
|---|---|
| **Postcondiciones:** | Se visualizará en la pantalla el nodo con su información actualizada. |
| **Prioridad:** | Crítica. |
| **Sección:** | Modificar Información de una Arista. |
| **C U al que pertenece:** | Editar Pista de Carreras. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Su propósito es modificar la información de una Arista. |
| **Resumen:** | La sección se inicia cuando el Diseñador de pistas de carreras selecciona una arista(s) y modifica su información. |
| **Referencia:** | R3, R3.1, R3.1.1, R3.1.2 |
| **Precondición** | Deben haberse seleccionado una arista. |
| **Flujo Normal de Eventos** | |

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. Selecciona la arista(s) a la cual modificar su información. | 2. Muestra la información de la arista seleccionada. |
| 3. Cambia la información referente a la arista seleccionada. | 4. Actualiza la información de la arista entrada por el Diseñador de Pistas de carreras. |
| | 5. Se visualizará el nodo con su información actual y termina el caso de uso. |

| | |
|---|---|
| **Postcondiciones:** | Se visualizará en la pantalla la arista con su información actualizada. |
| **Prioridad:** | Crítica. |
| **Sección:** | Eliminar Nodo. |
| **C U al que pertenece:** | Editar Pista de Carreras. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Su propósito es eliminar un nodo. |
| **Resumen:** | La sección se inicia cuando el Diseñador de pistas de carreras selecciona un nodo(s) y lo elimina. |
| **Referencia:** | R3, R3.1, R3.1.1, R3.1.2 |
| **Precondición** | Deben haberse seleccionado un nodo. |

**Flujo Normal de Eventos**

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. Selecciona el nodo(s) a eliminar. | 2. Muestra la información del nodo seleccionado. |
| 3. Elimina el nodo seleccionado. | 4. Elimina el nodo y actualiza el Grafo de Caminos y termina el caso de uso. |

| | |
|---|---|
| **Postcondiciones:** | Se visualizará en la pantalla Grafo de Caminos actualizado. |
| **Prioridad:** | Crítica. |
| **Sección:** | Eliminar Arista. |
| **C U al que pertenece:** | Editar Pista de Carreras. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Su propósito es eliminar una arista. |
| **Resumen:** | La sección se inicia cuando el Diseñador de pistas de carreras |

| | |
|---|---|
| | selecciona una arista(s) y la elimina. |
| **Referencia:** | R3, R3.1, R3.1.1, R3.1.2 |
| **Precondición** | Deben haberse seleccionado una arista. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona la arista(s) a eliminar. | 2. Muestra la información de la arista seleccionada. |
| 3. Elimina la arista seleccionada. | 4. Elimina la arista y actualiza el Grafo de Caminos. Se culmina el caso de uso |
| **Postcondiciones:** | Se visualizará en la pantalla Grafo de Caminos actualizado. |
| **Prioridad:** | Crítica. |

**Tabla 13** Descripción CU Editar Pista de Carreras.

| | |
|---|---|
| **Caso de Uso:** | Activar Modo Exploración. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Proporcionarle al Diseñador de pistas de carreras un mecanismo de comprobación de que el proceso de edición fue el deseado. |
| **Resumen:** | El caso de uso se inicia cuando el Diseñador de pistas de carreras escoge la opción de Modo Exploración. |
| **Referencia:** | R4, R4.1, R4.2, R4.2.1 |
| **Precondición:** | Realización del CU Editar Pista de Carreras. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona la opción Test Mode. | Muestra los 2 posibles modos de prueba. |
| 2. Selecciona un Modo de Exploración. | Realiza el modo de prueba seleccionado. |
| **Flujos Alternos** | |

| | |
|---|---|
| 2.1 Selecciona Modo Exploración Automático mediante la opción Play del menú Edit Mode. | 5. Realiza un recorrido automático por la pista editada previamente, en caso de que exista caminos que se bifurquen el sistema irá pasando por todos de forma secuencial. |
| 2.2 Selecciona Modo Exploración Manual mediante la opción Stop del menú Edit Mode. | 6. Realiza un recorrido dirigido por el Diseñador de pistas de carreras a la pista editada previamente, en caso de que el usuario cometiera alguna violación el sistema le avisará de la misma. |
| **Postcondiciones:** | En el modo de Exploración Automático el sistema representará los caminos mediante curvas Spline. |
| **Prioridad:** | Crítica. |

**Tabla 14** Descripción CU Activar Modo Exploración.

| | |
|---|---|
| **Caso de Uso:** | Gestionar Fichero con Grafo de Caminos. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Guardar o Cargar fichero .trk con la información de la pista de carrera editada previamente. |
| **Resumen:** | El caso de uso se inicia cuando el Diseñador de pistas de carreras realiza una acción de salva o de carga sobre un fichero .trk. |
| **Referencia:** | R5, R5.1, R5.2 |
| **Precondición:** | Debe haberse realizado el CU Editar Pista de Carreras para el caso de salva y para el caso de carga debe existir un fichero .trk previamente guardado. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona la opción de Guardar o Cargar fichero .trk | |

| **Flujos Alternos** | |
|---|---|
| 1.2 Selecciona la opción de Guardar fichero .trk. | 1.2. Ir a la Sección Guardar Fichero trk. |
| 1.3 Selecciona la opción de Cargar fichero .trk. | 1.4. Ir a la Sección Cargar Fichero trk. |
| **Poscondición:** | |
| **Prioridad:** | Crítica. |

**Tabla 15** Descripción CU Gestionar Fichero con Grafo de Caminos.

| **Sección:** | Cargar Modelo MD2. |
|---|---|
| **C U al que pertenece:** | Cargar Modelo. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Su propósito es cargar un modelo en formato MD2. |
| **Resumen:** | La sección se inicia cuando el Diseñador de pistas de carreras selecciona la opción Open File y carga un modelo MD2. |
| **Referencia:** | R2, R2.1, R2.1.1. |
| **Precondición** | Debe haberse realizado el CU Cargar Mapa BSP. |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona la opción de Open File. | 2. Se muestra el cuadro de diálogo Open File que le permitirá al Diseñador de pistas de carreras especificar el directorio y el nombre del fichero que desea cargar. |
| 3. Especifica el directorio y el nombre del fichero a cargar y acciona el botón OK. | 4. Se cierra el cuadro de diálogo y se procede a cargar el modelo seleccionado por el Diseñador de pistas de carreras. |
| | 5. Se visualizará el mapa con el modelo cargado y se termina el caso de uso. |
| **Postcondiciones:** | Se visualizará en la pantalla el mapa con el modelo cargado |
| **Prioridad:** | Crítica. |

**Tabla 16** Descripción de la sección Cargar Modelo MD2.

| **Sección:** | Cargar Modelo 3DS. |
|---|---|
| **C U al que pertenece:** | Cargar Modelo. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Su propósito es cargar un modelo en formato 3DS. |
| **Resumen:** | La sección se inicia cuando el Diseñador de pistas de carreras selecciona la opción Open File y carga un modelo 3DS. |
| **Referencia:** | R2, R2.1, R2.1.2. |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona la opción de Open File Cargar Modelo 3DS. | 2. Se muestra el cuadro de diálogo Open File que le permitirá al Diseñador de pistas de carreras especificar el directorio y el nombre del fichero que desea cargar. |
| 3. Especifica el directorio y el nombre del fichero que se desea cargar y acciona el botón OK de menú del cuadro de diálogo Open File. | 4. Se cierra el cuadro de diálogo y se procede a cargar el modelo seleccionado por el Diseñador de pistas de carreras. |
| | 5. Se visualizará el mapa con el modelo cargado y se termina el caso de uso. |

| **Postcondiciones:** | Se visualizará en la pantalla el mapa con el modelo cargado |
|---|---|
| **Prioridad:** | Crítica. |

**Tabla 17** Descripción de la sección Cargar Modelo 3DS.

| **Sección:** | Guardar Fichero .trk. |
|---|---|
| **C U al que pertenece:** | Gestionar Fichero con Grafo de Caminos. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Su propósito es Guardar la pista de carrera editada en un fichero .trk. |
| **Resumen:** | La sección se inicia cuando el Diseñador de pistas de carreras selecciona la opción Save Track y guarda el fichero .trk. |
| **Referencia:** | R5, R5.1 |

| **Precondición** | Debe haberse realizado el CU Editar Pista de Carreras. |
|---|---|
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona la opción Save Track. | 2. Se muestra el cuadro de diálogo Save Track que le permitirá al Diseñador de pistas de carreras especificar el directorio y el nombre del fichero .trk a guardar. |
| 3. Especifica el directorio y nombra al fichero que se desea guardar. | 4. Se cierra el cuadro de diálogo y se procede a guardar el fichero .trk con la los datos de la edición de la pista de carreras. |
| | 5. Se visualizará el mapa con el modelo cargado y se termina el caso de uso. |
| **Postcondiciones:** | Se muestra el mapa con la pista guardada. |
| **Prioridad:** | Crítica. |

**Tabla 18** Descripción de la sección Guardar Fichero .trk.

| **Sección:** | Cargar Fichero .trk. |
|---|---|
| **C U al que pertenece:** | Gestionar Fichero con Grafo de Caminos. |
| **Actores:** | Diseñador de pistas de carreras. |
| **Propósito:** | Su propósito es Cargar la pista de carrera desde un fichero .trk. |
| **Resumen:** | La sección se inicia cuando el usuario selecciona la opción Open File. |
| **Referencia:** | R5, R5.2 |
| **Precondición** | Debe existir algún fichero .trk que cumpla con las especificaciones del mimo. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Selecciona la opción de Open File. | 2. Se muestra el cuadro de dialogo Open File que le permitirá al Diseñador de pistas de carreras especificar el directorio y el nombre del fichero que desea cargar. |
| 3. Especifica el directorio y el nombre del fichero que se desea cargar. | 4. Se cierra el cuadro de dialogo y se procede a cargar el fichero seleccionado por el Diseñador |

| | |
|---|---|
| | de pistas de carreras. |
| | 5. Se visualizará el mapa con la edición guardada en el fichero .trk cargado y se culmina el caso de uso. |
| **Flujo Alterno** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1.1 El Diseñador de pistas de carreras hace uso de la bondad Drag and Drop de la aplicación y arrastra para la ventana el mapa BSP a cargar. | 6. El sistema carga el fichero trk y se termina el caso de uso. |
| **Postcondiciones:** | Se visualizará en la pantalla el mapa con la edición previamente salvada. |
| **Prioridad:** | Crítica. |

**Tabla 19** Descripción de la sección Cargar Fichero .trk.

## 3.6.Interfaz de usuario

La interfaz de usuario también conocida por sus siglas en inglés como GUI (Graphical User Interface) es la encargada de mediar la interacción del usuario con el software de una forma fácil, obteniéndose resultados mejores y llegando a una solución más viable. La misma debe ser sencilla para que el usuario logre con un mínimo de conocimiento llegar a la solución deseada.

A continuación se presentan cada uno de los menús de la herramienta propuesta y se da una descripción de las funciones que de los mismos.

### 3.6.1.Menú Developer Tools

La Figura 16 muestra la apariencia general de la aplicación sin un mapa BSP cargado. El menú Developer Tools es el controlador de todas las ventanas de la aplicación, en él están contenidos todos los demás menús de la aplicación.


**Figura 16** Menú Developer Tools.

### 3.6.1.1.Menú Open File

La Figura 17, representa la opción Open File, al ejecutarse la aplicación se escoge esta opción en el menú Developer Tools para proceder a la carga del mapa a editar. En el cuadro de texto File name el usuario especifica la dirección y el nombre del fichero a cargar, el cual puede ser de tipo *.BSP, *.3DS, *.MD2 o *.trk. Al presionar el botón Ok se procede a la carga del fichero especificado.

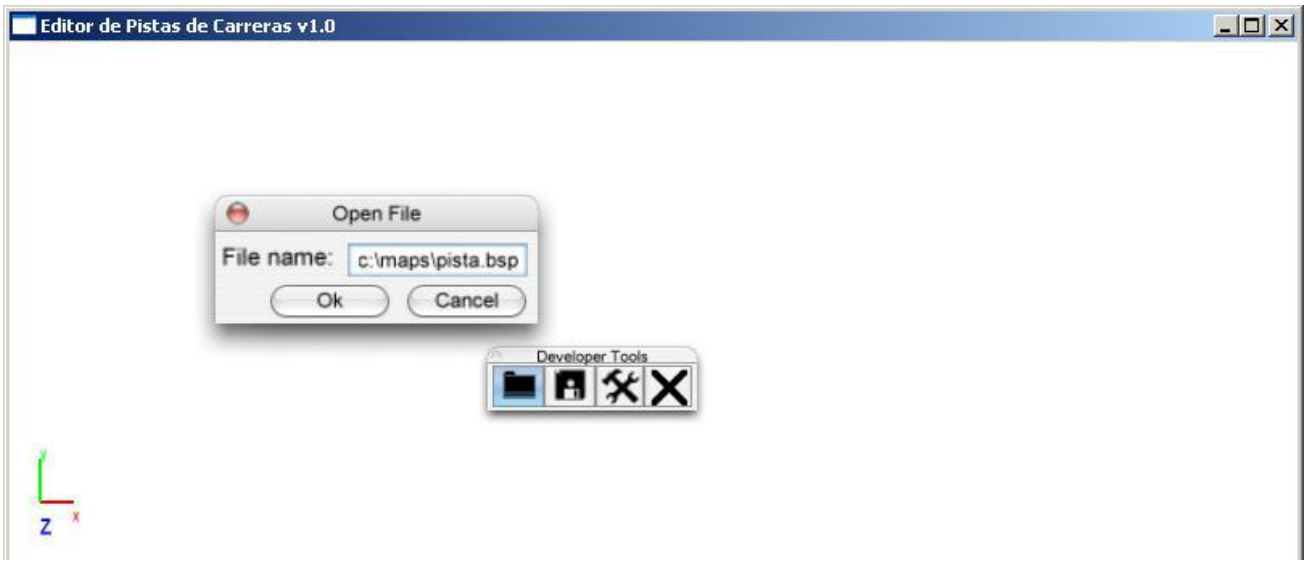


**Figura 17** Menú Open File.

Es válido destacar que no es obligatorio usar este menú para cargar algún fichero, ya que el sistema cuenta con un sistema DRAG & DROP (ver glosario de términos), lo cual le permite al usuario “arrastrar” el fichero deseado hasta la aplicación y el sistema cargará dicho fichero.

### 3.6.1.2.Menú Save Track

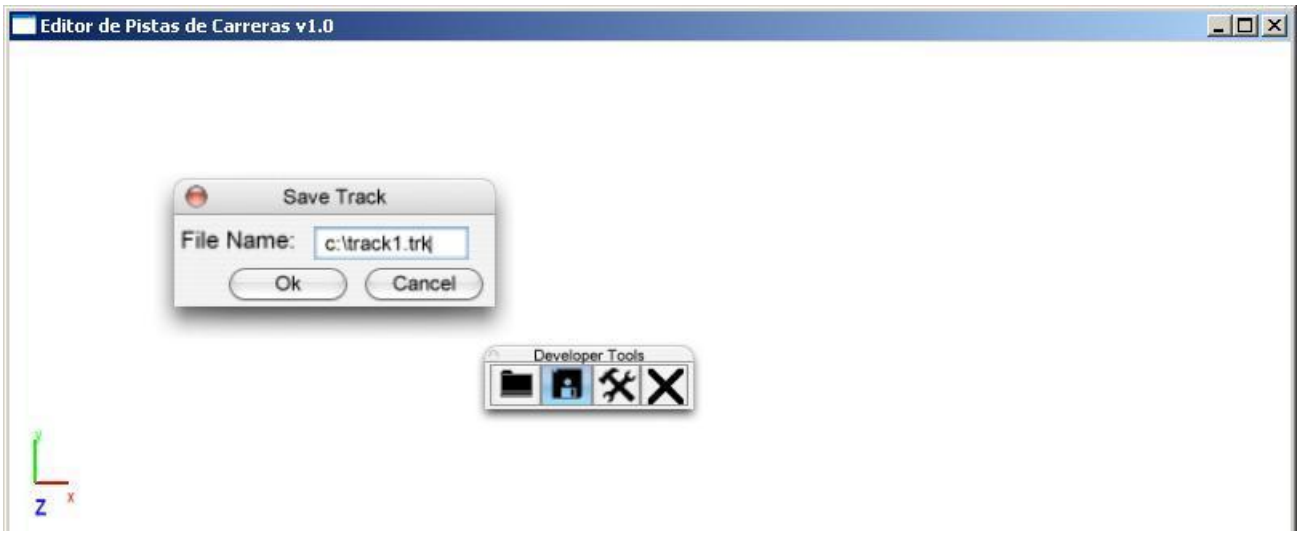


**Figura 18** Menú Save Track.

El menú Save Track presenta un cuadro de texto en el cual el usuario de la aplicación especificará la ubicación física y el nombre con el cual guardará el fichero .trk, el cual guarda la información de la pista editada previamente como se ha explicado antes.

### 3.6.1.3.Menú Control Pane

El menú Control Pane constituye la parte principal de la interfaz de usuario para el proceso de edición del circuito de carrera. Está compuesto por 3 partes principales NODE, EDGE, TEST MODE, las cuales se explican a continuación:

**NODE:** Panel encargado de la creación y edición de los nodos, contiene elementos como el tipo de nodo, radio de los mismos, así como la posición en la que se encuentra el nodo. Todos estos parámetros son ajustables lo que permite personalizar el proceso de edición de los nodos que contendrá el grafo de caminos.

**EDGE:** Contiene la información referente a las aristas del grafo de caminos, mediante este panel se puede variar el coeficiente de rozamiento de los caminos, lo cual permite simular diferentes tipos de terrenos.

**TEST MODE:** Contiene los dos modos de exploración con que cuenta la aplicación, Modo de Exploración Automática mediante el cual es el sistema el que realiza un recorrido por la pista editada, este modo será activado mediante el botón Play. El otro modo de exploración es el Manual el cual es activado mediante el botón Stop, y es el usuario el que dirige este proceso, en caso de cometer alguna violación del sentido de la pista de carrera, será alertado del mismo mediante un mensaje de Wrong Way.

El menú Control Pane también cuenta con la opción Show Path, la que controla que se muestre o no, la pista editada. Además de contar con un checkbox nombrado Edit Mode, el cual solo tiene un fin informativo para que el usuario sepa si se encuentra en modo edición o en modo de exploración.


**Figura 19** Menú Control Pane.

En este capítulo se establecieron las reglas del negocio, los requerimientos, se identificaron los actores del sistema, se detectaron los casos de uso del sistema y se realizaron las descripciones detalladas de cada uno de los casos de uso; lo que permitirá en un futuro realizar el análisis, diseño e implementación del sistema. Además de que se establecieron los prototipos de la interfaz de usuario de la aplicación.

# Capítulo 4. Análisis y Diseño del Sistema.

En este capítulo se profundizará más en el desarrollo del sistema, dando continuidad a lo tratado anteriormente. En la primera parte se abordan los temas del análisis, identificado con los diagramas de clases del análisis y los diagramas colaboración de la realización de los casos de uso correspondientes.

A partir del Análisis se realizará el Diseño de la aplicación, soportada sobre los diagramas de clases del diseño agrupados por paquetes y sus respectivos diagramas de secuencia. El caso específico de la GUI, por no ser objetivo de este trabajo no se describe sus clases, ni se le realizan diagramas de secuencia.

En este capítulo también se hace una justificación de los patrones utilizados en el flujo de trabajo Diseño.

## 4.1.Diagrama de Clases del Análisis.

### 4.1.1.Diagrama de Clases del Análisis de CU Cargar Mapa BSP

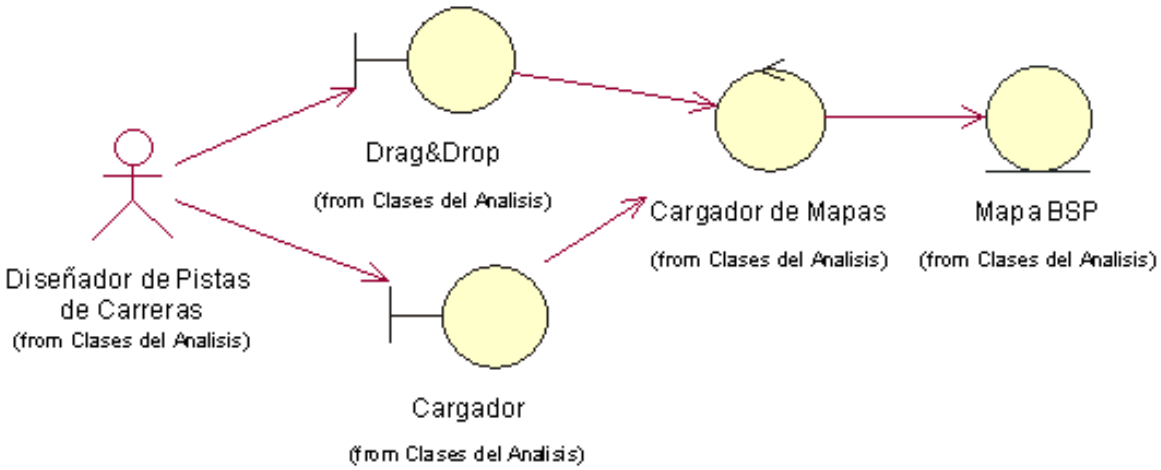


**Figura 20** Diagrama de Clases del Análisis de CU Cargar Mapa BSP.

### 4.1.2.Diagrama de Clases del Análisis de CU Cargar Modelos

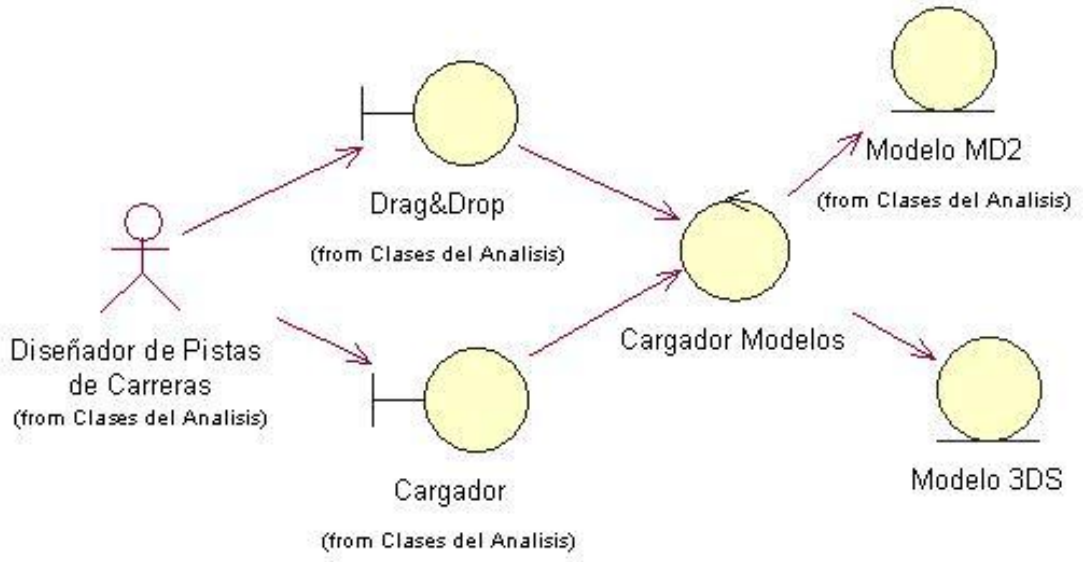


**Figura 21** Diagrama de Clases del Análisis de CU Cargar Modelo.

### 4.1.3.Diagrama de Clases del Análisis de CU Editar Pista de Carreras

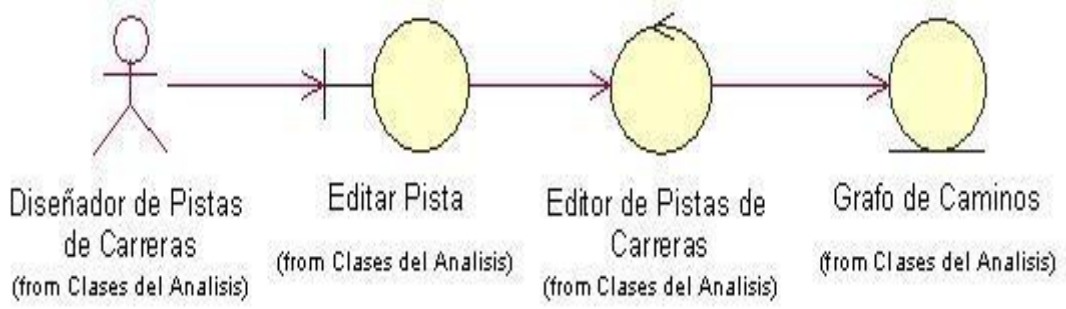


**Figura 22** Diagrama de Clases del Análisis de CU Editar Pista de Carreras.

### 4.1.4.Diagrama de Clases del Análisis de CU Activar Modo Exploración

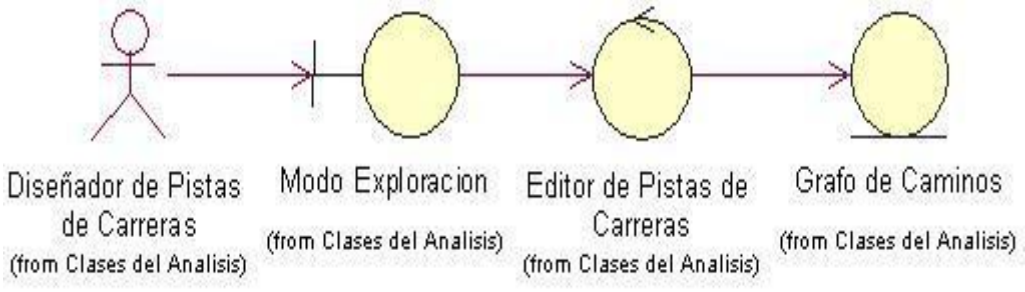


**Figura 23** Diagrama de Clases del Análisis de CU Activar Modo Exploración.

### 4.1.5.Diagrama de Clases del Análisis de CU Gestión de Fichero TRK

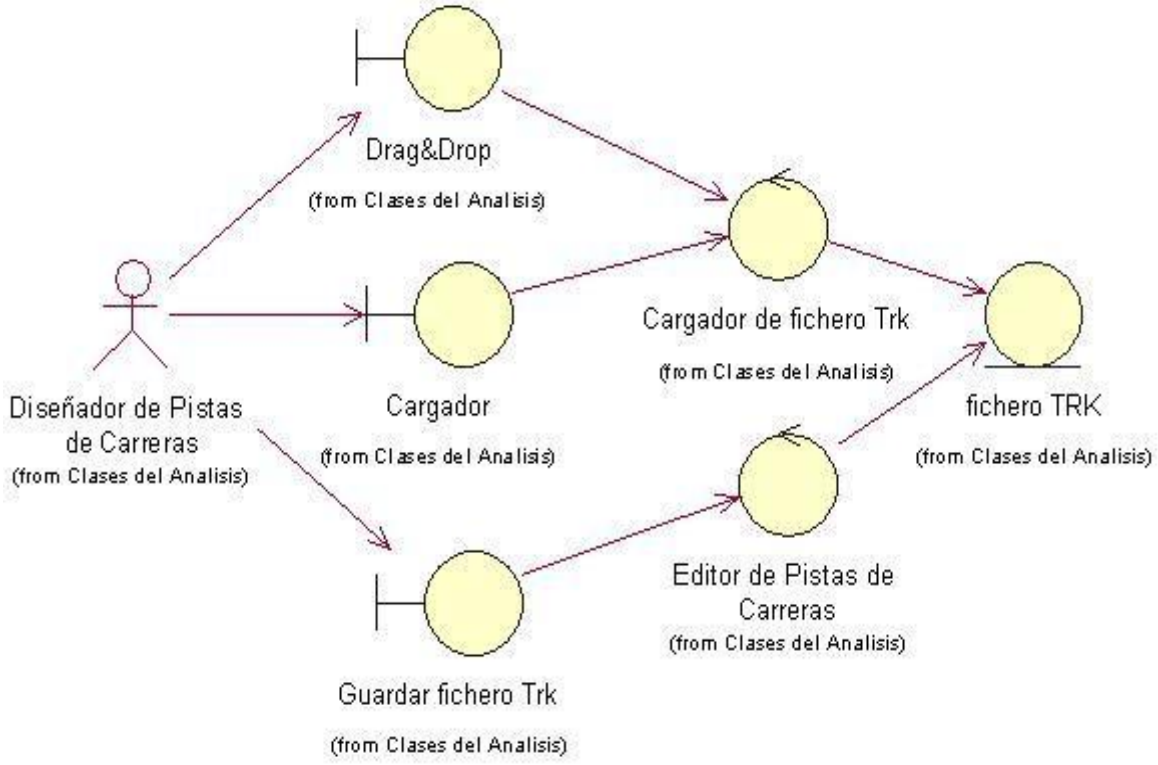


**Figura 24** Diagrama de Clases del Análisis de CU Gestión de Fichero TRK.

## 4.2.Diagramas de Colaboración del Análisis

Para ganar en claridad con respecto a la interacción entre los objetos de las clases del análisis se realizan los siguientes diagramas de colaboración de cada uno de los CU, capturados en el capítulo anterior.

### 4.2.1.Diagramas de Colaboración del Análisis CU Cargar Mapa BSP

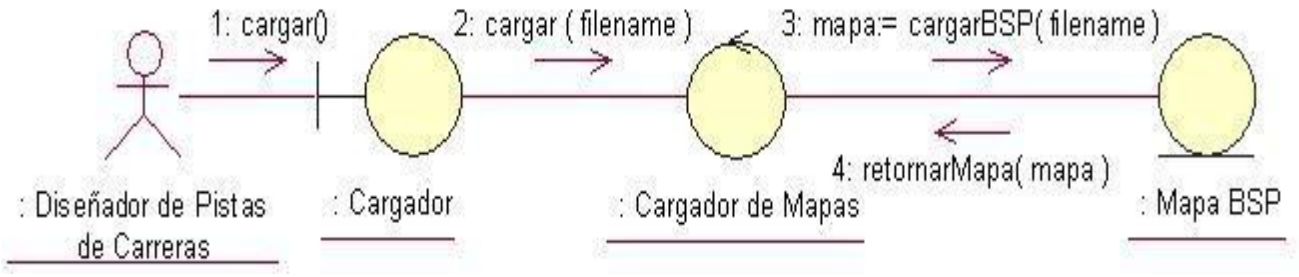


**Figura 25** Diagrama de colaboración del análisis CU Cargar Mapa BSP

### 4.2.2.Diagrama de Colaboración del Análisis para el CU Cargar Modelos

El CU Cargar Modelo por razones de claridad fue separado en las secciones: Cargar Modelo MD2 y Cargar Modelo 3DS, por este motivo a continuación se presentan los diagramas de colaboración de cada una de estas secciones que componen el CU.

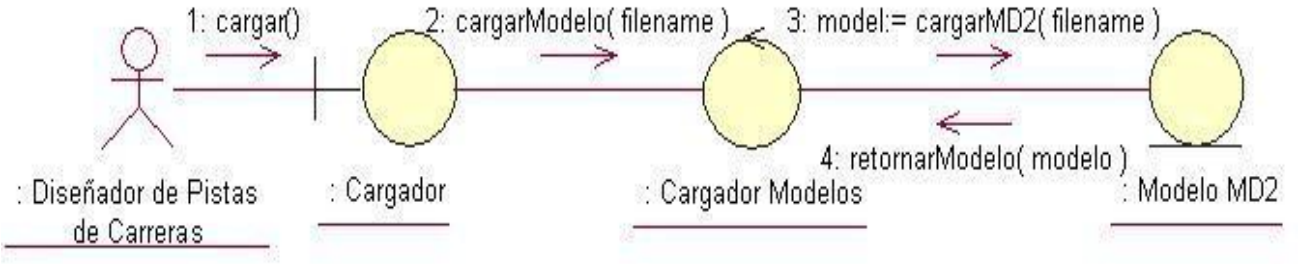


**Figura 26** Diagrama de colaboración del análisis del CU Cargar Modelos. Sección Cargar Modelo MD2.

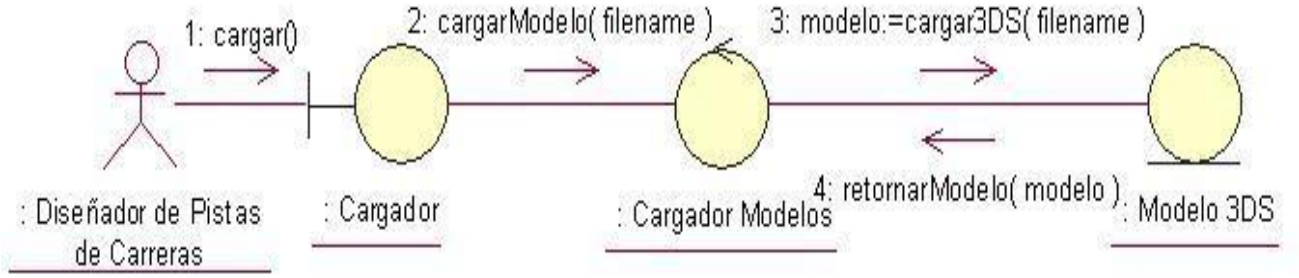


**Figura 27** Diagrama de colaboración del análisis del CU Cargar Modelos. Sección Cargar Modelo 3DS.

Para los diagramas de colaboración antes presentados es válido destacar que solo representan la carga de estos ficheros a través de la interfaz cargador contenida en el cuadro de diálogo Open File, no se realizaron los diagramas de colaboración para la interfaz Drag & Drop porque solo sería cambiar el nombre de la interfaz cargador por el de la interfaz Drag & Drop, todo lo demás se mantendría inalterable. Lo mismo sucede con la carga del fichero trk.

### 4.2.3.Diagrama de Colaboración del Análisis para el CU Editar Pista de Carreras

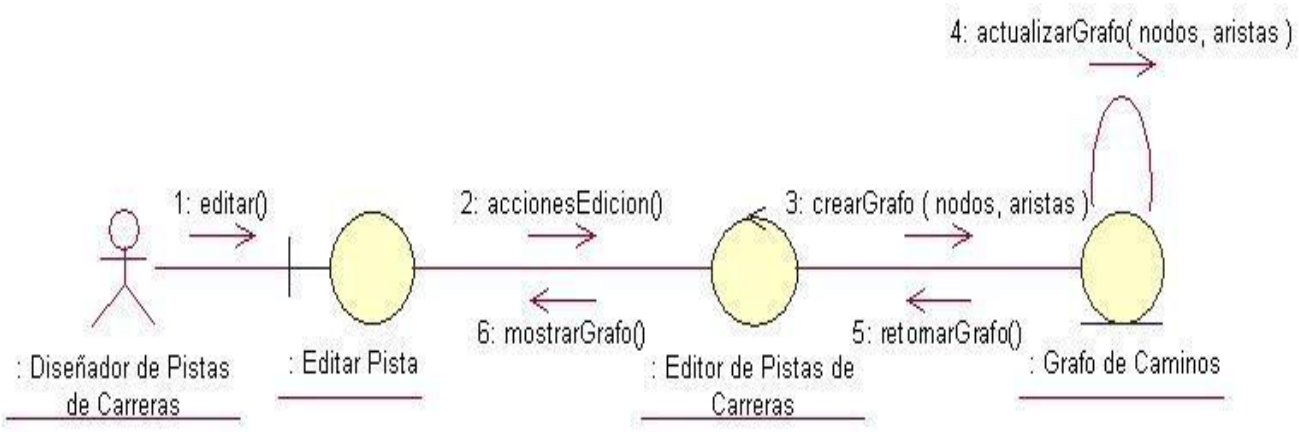


**Figura 28** Diagrama de colaboración del análisis para el CU Editar Pista de Carreras.

### 4.2.4.Diagrama de Colaboración del Análisis para el CU Activar Modo Exploración

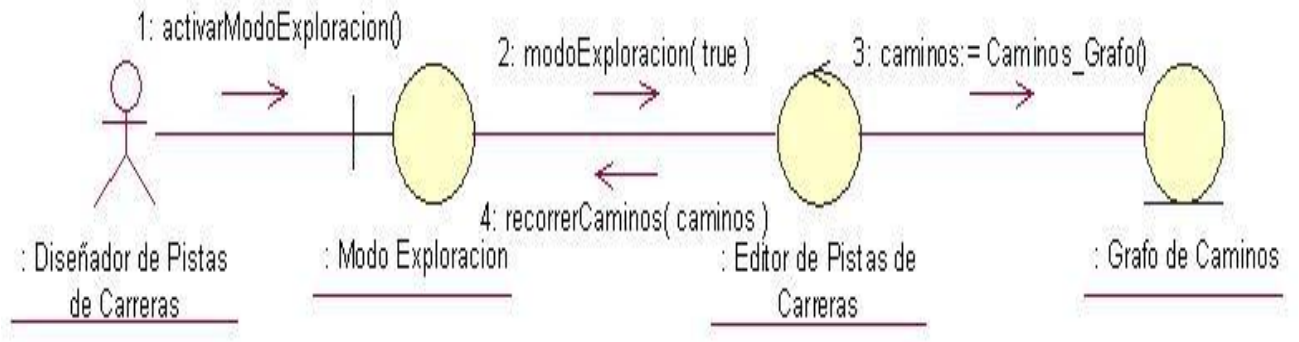


**Figura 29** Diagrama de colaboración del análisis para el CU Activar Modo Exploración.

### 4.2.5.Diagrama de Colaboración del Análisis para el CU Gestión de Fichero con Trayectoria de la Pista de Carrera

Igual que en los CU anteriores, al CU Gestión de Fichero con Trayectoria de la Pista de Carreras también se le realizó un diagrama de colaboración, pero por razones de claridad el caso de uso fue separado en las secciones: Guardar Fichero .TRK, Cargar Fichero TRK, por este motivo a continuación se presentan los diagramas de colaboración de cada una de estas secciones que componen el CU.

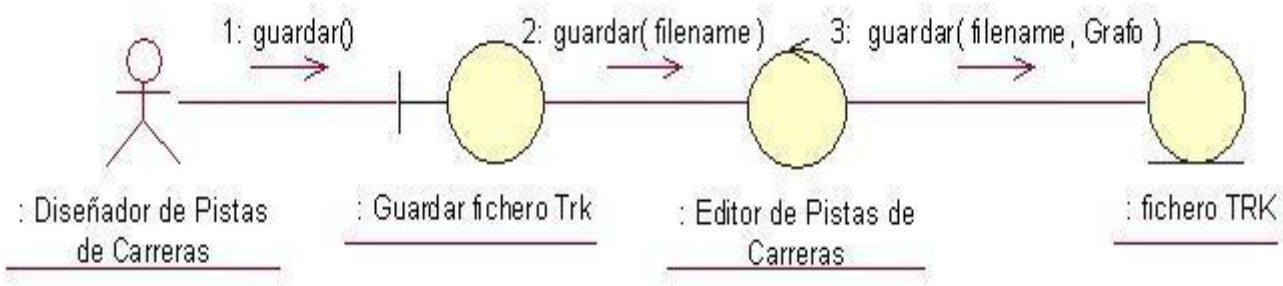


**Figura 30** Diagrama de colaboración del análisis del CU Gestión de Fichero con Trayectoria de la Pista de Carrera. Sección Guardar Fichero .TRK.

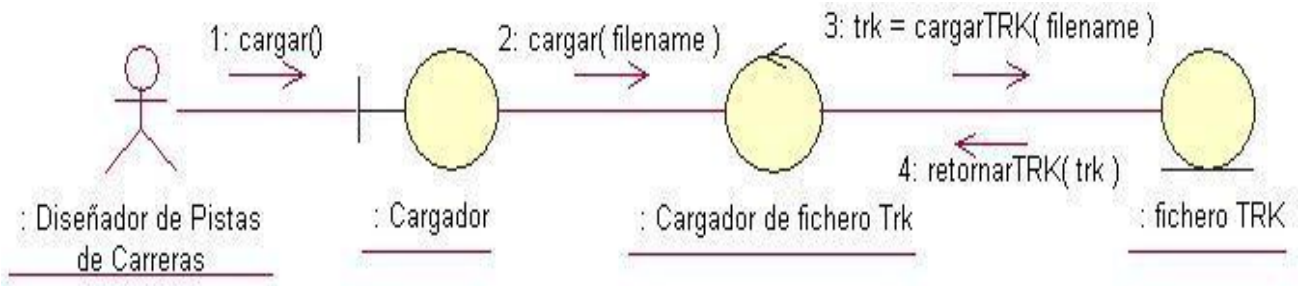


**Figura 31** Diagrama de colaboración del análisis del CU Gestión de Fichero con Trayectoria de la Pista de Carrera. Sección Cargar Fichero .TRK.

## 4.3.Patrones

Un patrón es la descripción etiquetada de un problema, de la solución, de cuándo aplicar la solución y la manera de hacerlo dentro de otros contextos **[15].** Los patrones de diseño describen un problema que ocurre repetidas veces en algún contexto determinado de desarrollo de software y entregan una buena solución ya probada. Esto ayuda a diseñar correctamente y en menos tiempo, ayuda a construir problemas reutilizables y extensibles y facilita la documentación.

En el presente trabajo se estudiaron y aplicaron los siguientes patrones de diseño que pertenecen al grupo de los famosos 23 patrones de Gang of Four (GOF)

### 4.3.1.Strategy

El patrón strategy (estrategia) está orientado a resolver situaciones en las que se origina el problema base de existir diversas estrategias para abordar un mismo problema. En este sentido el problema define una interface que será implementada de diversas formas. **[16]**.

Cualquier programa que ofrezca un servicio o función determinada, que pueda ser realizada de varias maneras, es candidato a utilizar este patrón. Puede haber cualquier número de estrategias y cualquiera de ellas podrá ser intercambiada por otra en cualquier momento, incluso en tiempo de ejecución.

Este patrón fue implementado en el proceso de carga de los ficheros, en el cuadro de diálogo Open File, el usuario pasa la dirección y el nombre del fichero a cargar, y en dependencia de la extensión del mismo, la aplicación decide que estrategia de carga poner en práctica.

El funcionamiento de este patrón es muy simple y el añadir nuevas estrategias al programa es muy sencillo y apenas implica modificación de código alguna.

### 4.3.2.Command

Este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma **[17]**. Para ello se encapsula la petición como un objeto, con lo que además se facilita la parametrización de los métodos.

Encapsula un mensaje como un objeto, con lo que permite gestionar colas o registro de mensaje y deshacer operaciones. Este patrón tiene como principales funcionalidades:

- Restaurar el estado a partir de un momento dado.
- Ofrecer una interfaz común que permita invocar las acciones de forma uniforme y extender el sistema con nuevas acciones de forma más sencilla.
- Facilitar la parametrización de las acciones a realizar.
- Independizar el momento de petición del de ejecución.

- Implementar CallBacks, especificando que órdenes queremos que se ejecuten en ciertas situaciones de otras órdenes. Es decir, un parámetro de una orden puede ser otra orden a ejecutar.
- Soportar el "deshacer".

Este patrón fue utilizado para la construcción de la GUI de la aplicación, debido a que el mismo permite independizar la parte de la aplicación que invoca las órdenes de la implementación de los mismos. De esta forma se garantiza que la parte de presentación sea independiente de la parte lógica de la aplicación. Además de brindar el mecanismo de “undo” para futuras versiones de la herramienta propuesta.

### 4.3.3.Singleton

El patrón Singleton es uno de los más sencillos patrones de diseño, su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella **[17],** por lo que si más de un objeto necesita utilizar una instancia de la clase Singleton, esos objetos comparten la misma instancia de la clase Singleton.

Una clase que implementa el patrón Singleton se conoce como una clase Singleton, este es el caso de la clase App, la cual es la clase controladora de la aplicación. Este patrón, como se explicó anteriormente, proporcionó que la clase App fuera una instancia única en toda la aplicación.

## 4.4.Diagrama de Clases del Diseño del Sistema

El lenguaje UML ofrece el mecanismo paquete que permite describir los grupos de elementos o subsistemas. Un paquete es un conjunto de cualquier tipo de elementos de un modelo: clases, casos de uso, diagramas de colaboración u otros paquetes (los anidados). El paquete define un espacio de un nombre anidado, de modo que los elementos del mismo nombre pueden duplicarse dentro de varios paquetes **[16]**.

El diagrama de clases del diseño fue desglosado por paquetes para mantener la claridad y la fácil comprensión del mismo. Fueron desarrollados 3 paquetes: **Gui**, **Loader** y **TrackEditor**.

**Gui**, encierra todo lo referente a la interfaz gráfica de usuario, entiéndase ventanas, botones, menús, además de la visualización del grafo de camino. Este paquete cuenta con un modelo de

delegados que permite independizar completamente la interfaz de usuario de la lógica de la aplicación, fomentando de esta forma la reutilización de cualquiera de sus paquetes.

El paquete **Loader**, como indica su nombre, es el encargado de la manipulación de todos los ficheros que carga el sistema, entiéndase ficheros BSP, MD2, 3DS y TRK el cual es el fichero que produce el sistema para almacenar el grafo de camino editado.

El paquete **TrackEditor** es el encargado de llevar la lógica del sistema, además de ser el encargado de crear el fichero *TRK, que es el producto final de la aplicación.

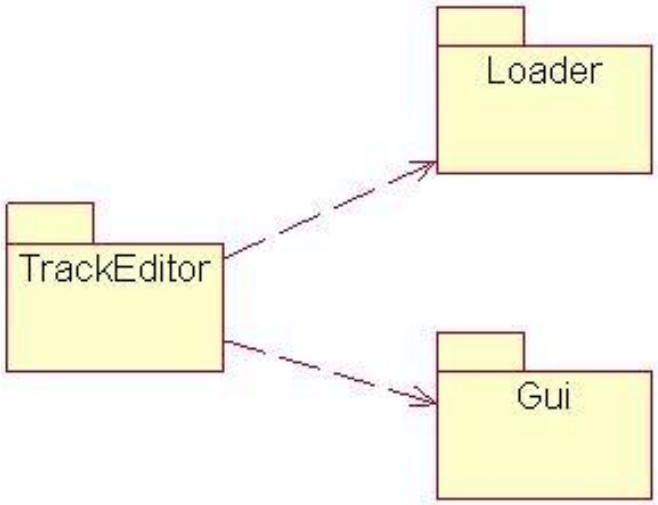


**Figura 32** Diagrama General de Paquetes de Clases del Diseño.

### 4.4.1.Diagrama de Clases del Paquete LOADER

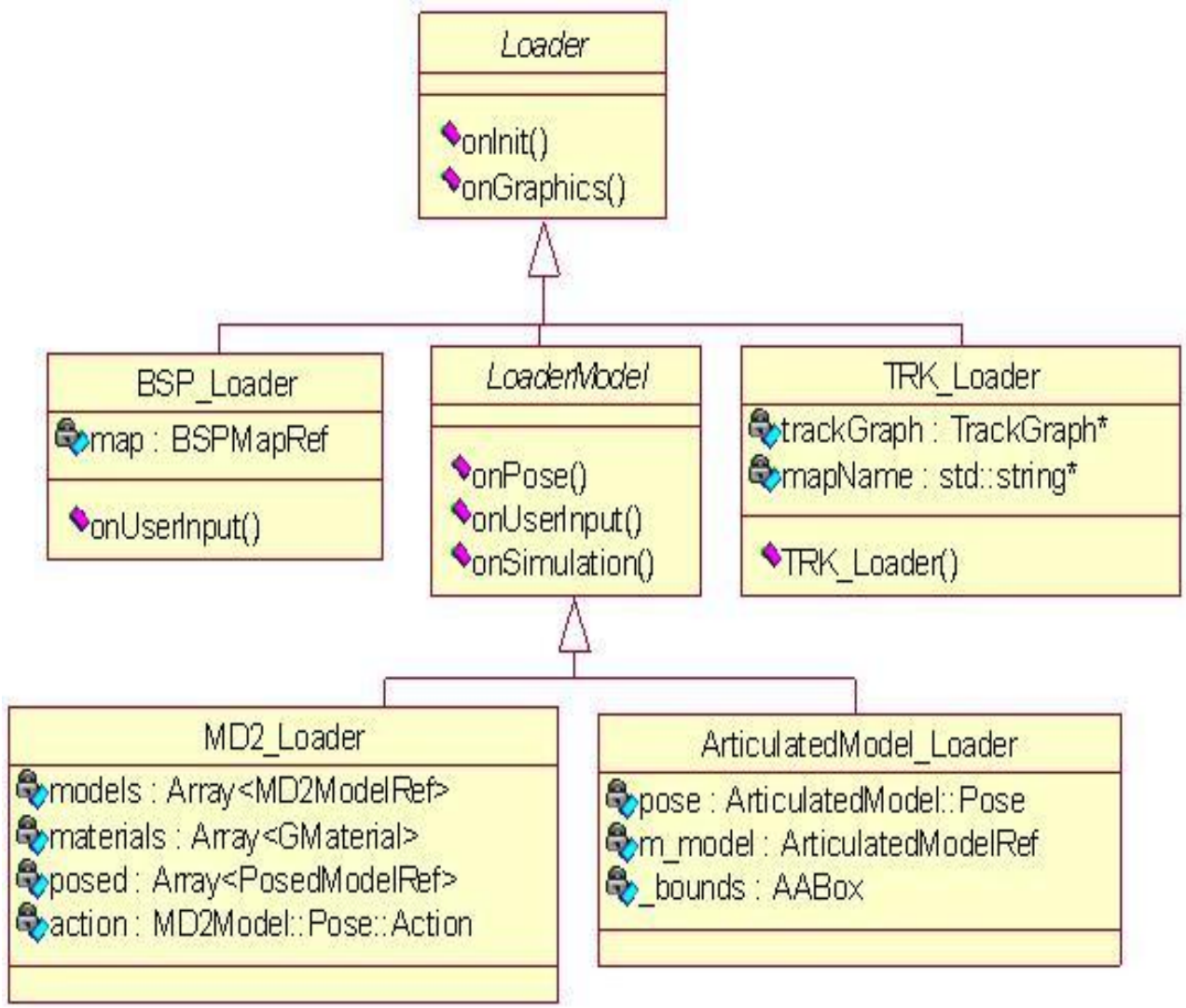


**Figura 33** Diagrama de Clases del Diseño del Paquete Loader**.**

### 4.4.2.Diagrama de Clases del Paquete TrackEdit

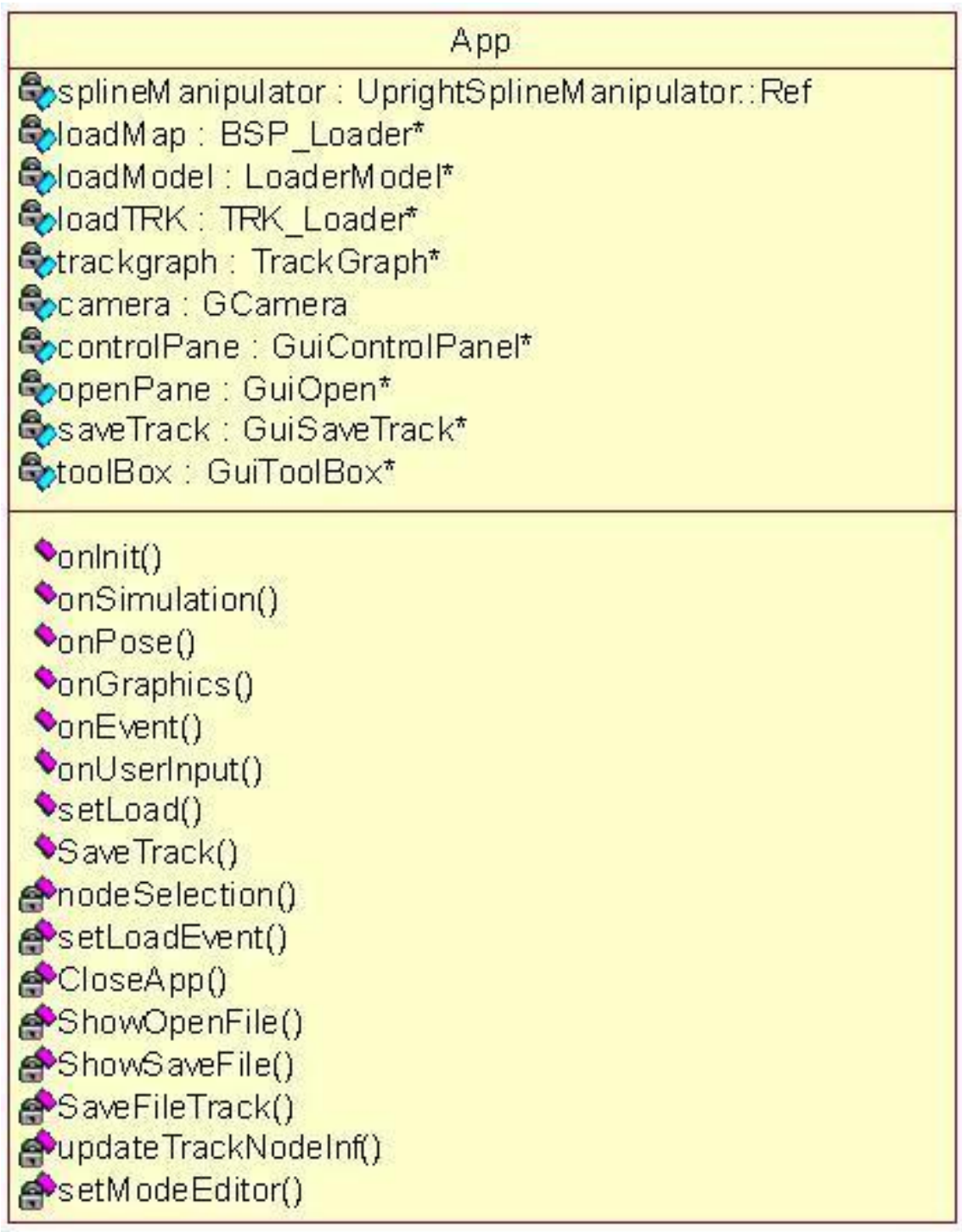


**Figura 34** Diagrama de Clases del Diseño del Paquete TrackEditor.

# 4.5.Diagramas de Secuencia del Diseño

## 4.5.1.Diagrama de Secuencia del Diseño del CU Cargar Mapa BSP

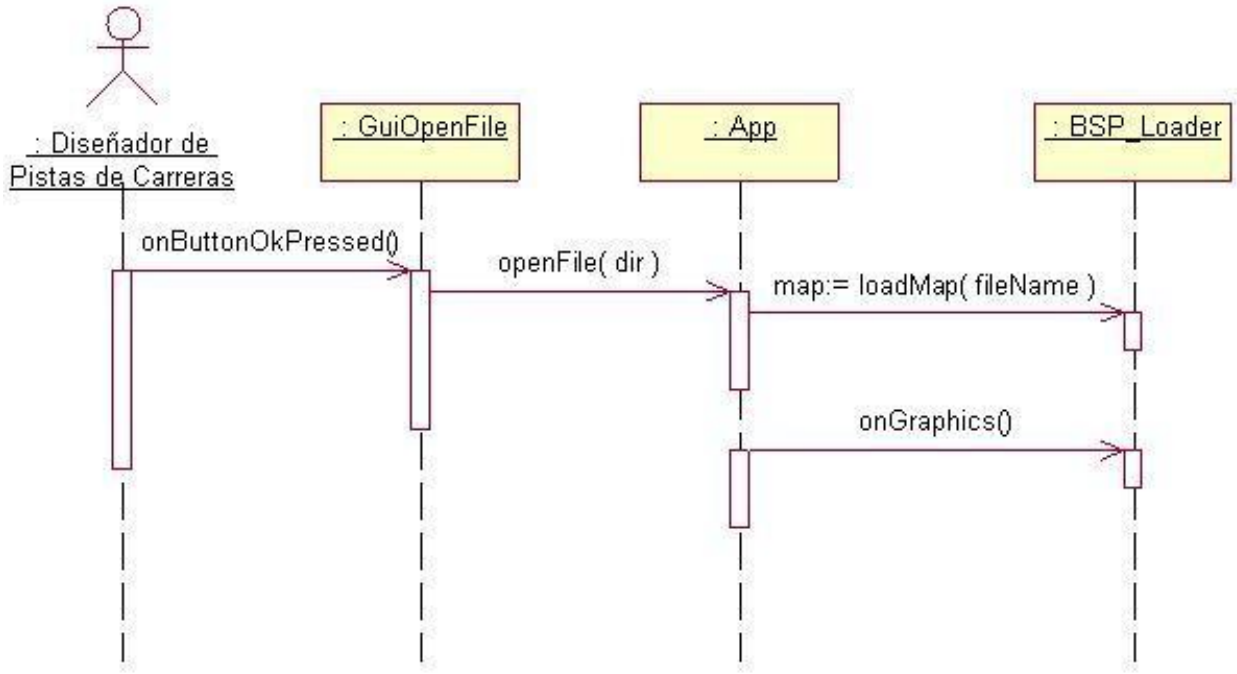


**Figura 35** Diagrama de Secuencia del Diseño del CU Cargar Mapa BSP.

## 4.5.2.Diagrama de Secuencia del Diseño del CU Cargar Modelos

Para ganar en claridad se ha dividido esta sesión en dos casos de uso, caso de uso Cargar Modelo MD2 y caso de uso Cargar Modelo 3DS al igual que se hizo en el Análisis.

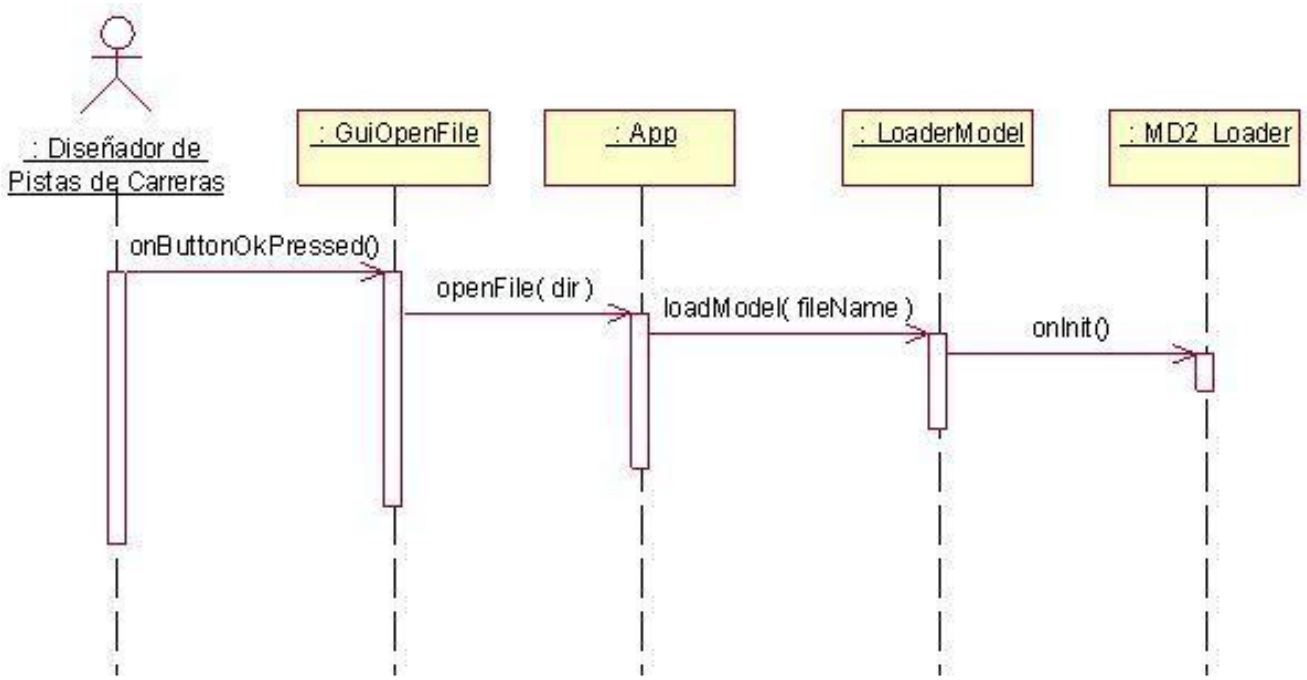


**Figura 36** Diagrama de Secuencia del Diseño del CU Cargar Modelo. Sección Cargar Modelo MD2.

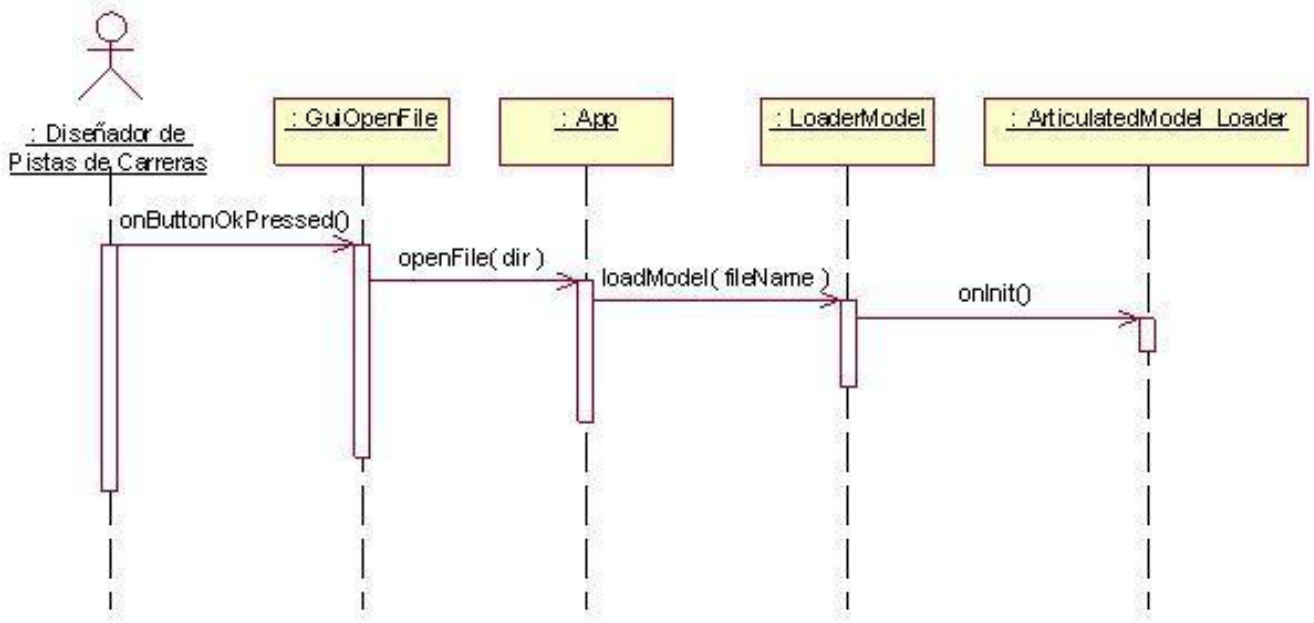


**Figura 37** Diagrama de Secuencia del Diseño del CU Cargar Modelo. Sección Cargar Modelo 3DS.

### 4.5.3.Diagrama de Secuencia del Diseño del CU Editar Pista de Carreras

Para ganar en claridad se realizó un diagrama de secuencia para cada escenario que presenta este complejo caso de uso, por ser similares en ocasiones algunos diagramas de secuencia para algunos escenarios, no se incluyeron en esta sección, en cada caso se especificará su similitud con el otro diagrama de secuencia.

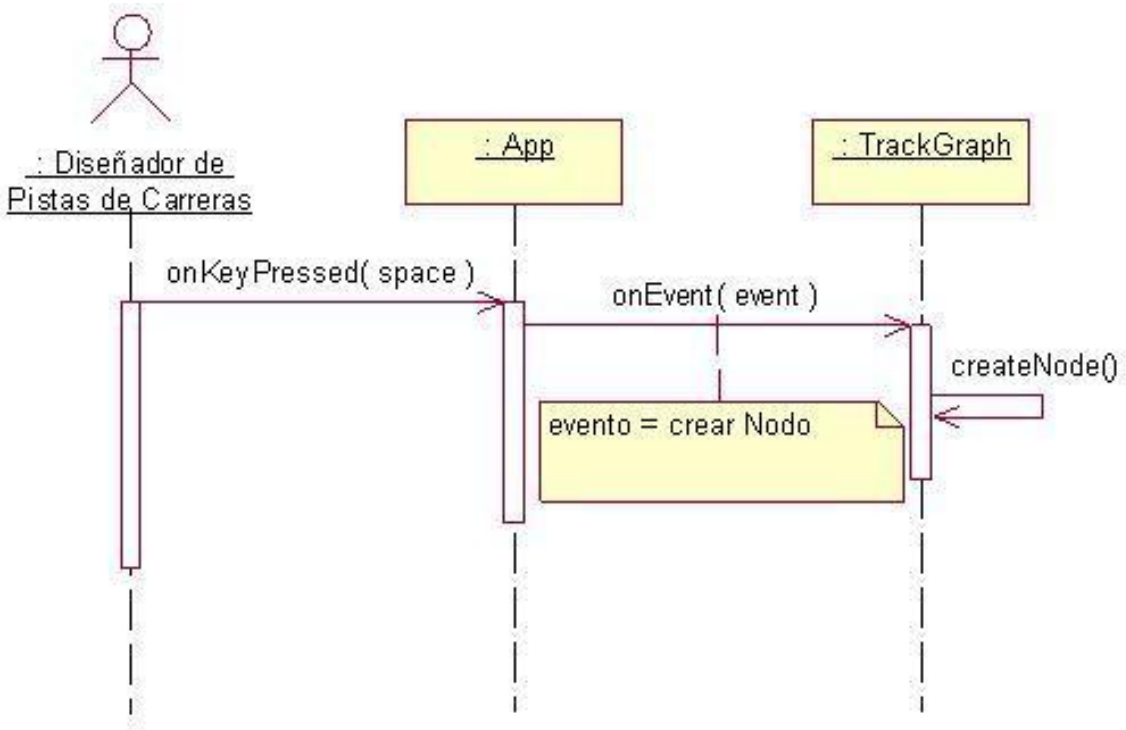


**Figura 38** Diagrama de Secuencia del Diseño del CU Editar Pista de Carreras. Sección Crear Nodo.

El diagrama de Secuencia de la Sección Crear Arista del CU Editar Pista de Carreras es similar al antes descrito, solo sería cambiar Node por Edge.

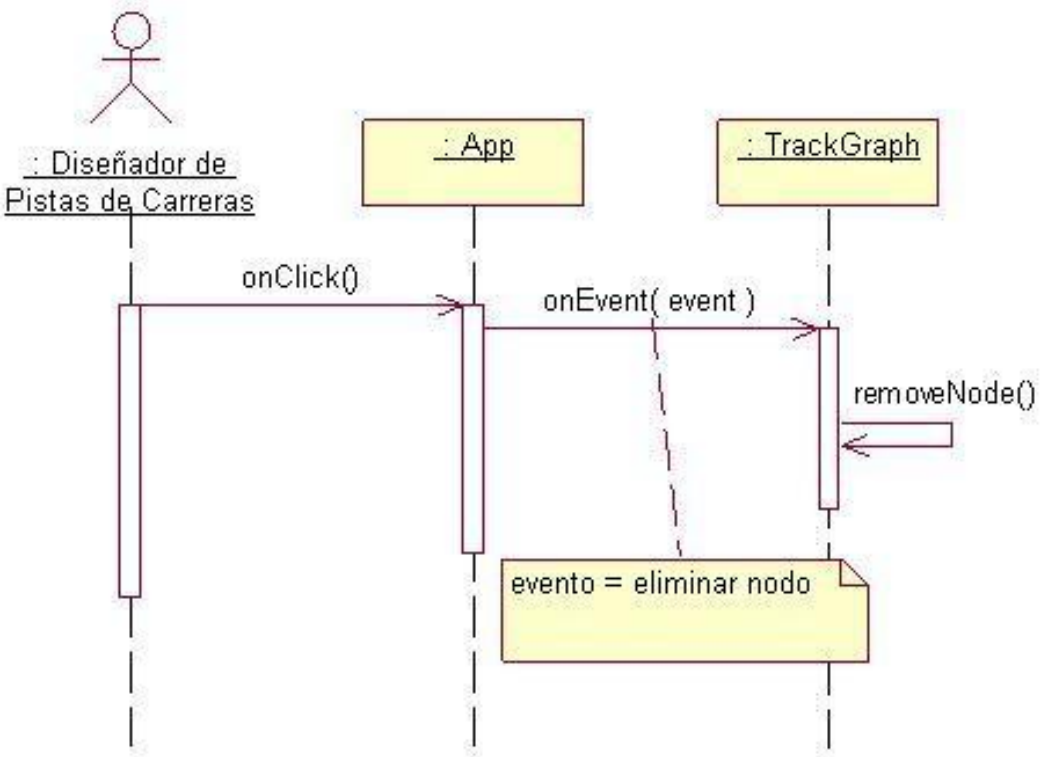


**Figura 39** Diagrama de Secuencia del Diseño del CU Editar Pista de Carreras. Sección Eliminar Nodo.

El diagrama de Secuencia de la Sección Eliminar Arista del CU Editar Pista de Carreras es similar al antes descrito, solo sería cambiar Node por Edge.

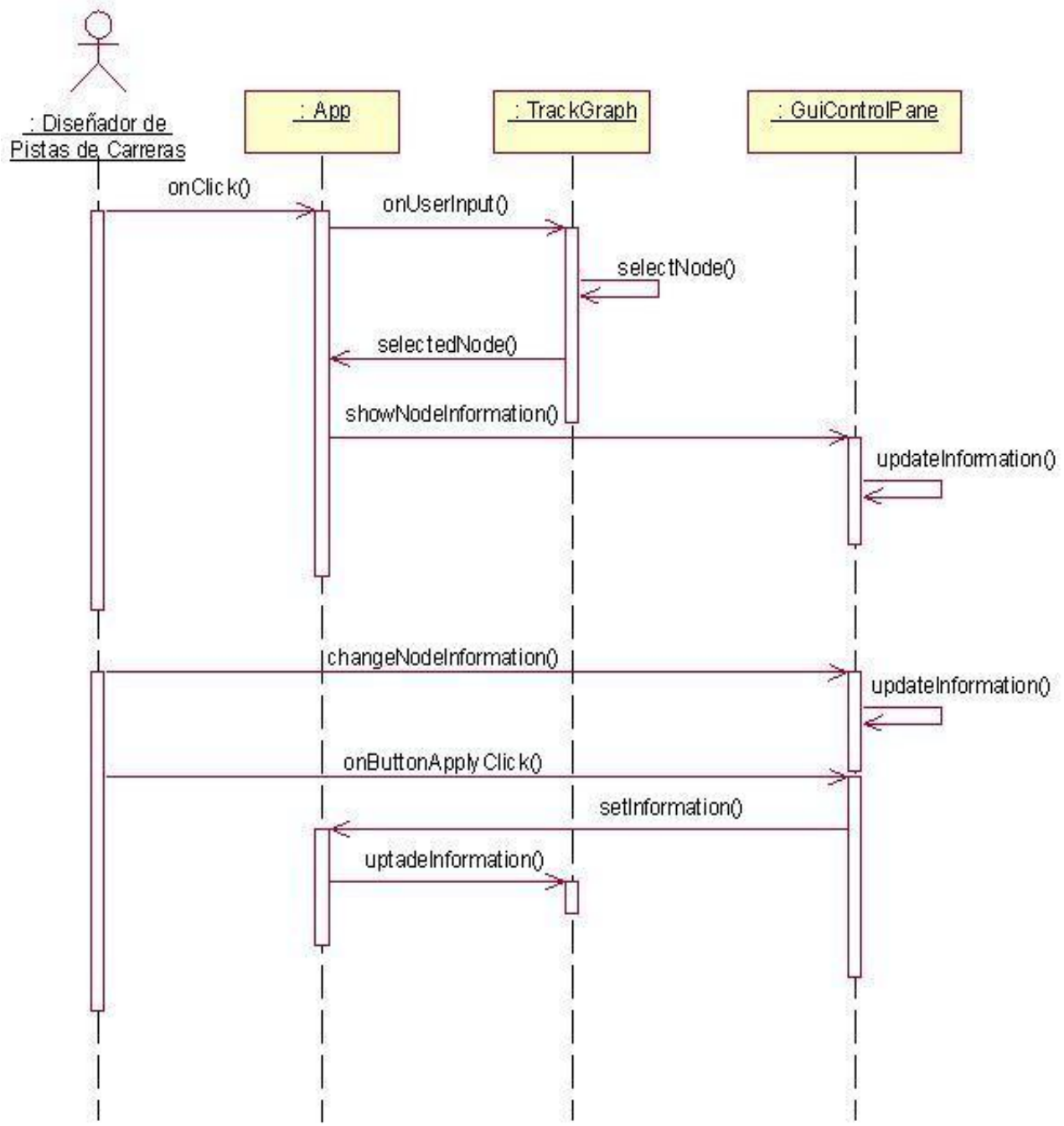


**Figura 40** Diagrama de Secuencia del Diseño del CU Editar Pista de Carreras. Sección Modificar Información de un Nodo.

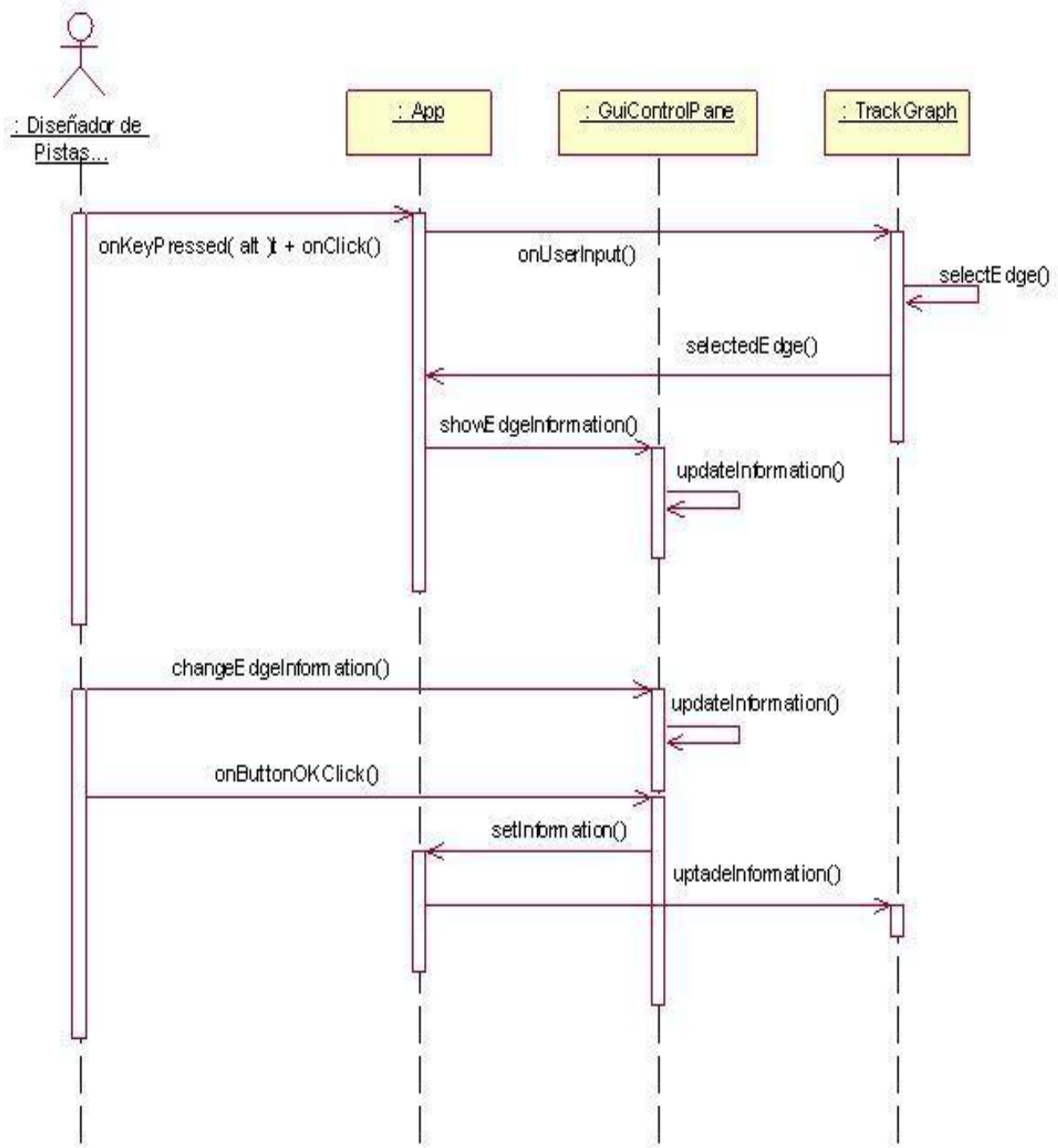


**Figura 41** Diagrama de Secuencia del Diseño del CU Editar Pista de Carreras. Sección Modificar Información de una Arista.

## 4.6.Descripción de las Clases

La descripción de las clases describe los atributos y métodos que tiene una clase, brindando la información necesaria para lograr un completo entendimiento de la misma. En esta sección se proporciona la descripción de la clase App, que constituye la principal clase del modelo pues es el núcleo de la arquitectura. La descripción del resto de las clases, cuyo conocimiento es indispensable para hacer uso del modelo creado, constituye parte de la documentación y se encuentra en formato web, siendo generada con la herramienta Doxygen.

| **Nombre**: | App |
|---|---|
| **Tipo de Clase**: | Controladora |
| **Atributo** | **Tipo** |
| splineManipulator | UprightSplineManipulator::Ref |
| splines | Array<UprightSpline> |
| _cameraManipulator | Manipulator::Ref |
| third_person | bool |
| loadMap | BSP_Loader* |
| mapName | std::string |
| loadModel | LoaderModel* |
| loadTRK | TRK_Loader* |
| trackgraph | TrackGraph* |
| camera | GCamera* |
| controlPane | GuiControlPanel* |
| openPane | GuiOpen* |
| saveTrack | GuiSaveTrack* |

| | |
|---|---|
| toolBox | GuiToolBox* |
| lighting | LightingRef |
| skyParameters | SkyParameters |
| sky | SkyRef |
| **Responsabilidades:** | |
| **Nombre**: | **Descripción**: |
| App(const GApp::Settings& settings = GApp::Settings()) | Constructor de la clase. |
| onInit () | Inicializa la aplicación y a todos los componentes de la misma. |
| onLogic () | Encargado de la lógica de la aplicación. |
| onSimulation (RealTime rdt, SimTime sdt, SimTime idt) | Simulación de las entidades de la aplicación. |
| onPose(Array<PosedModelRef>& posed3D, Array<PosedModel2DRef>& posed2D) | Encargado del render de todos los elementos que implementen este método, esta es una de las bondades del uso del Engine G3D. |
| onGraphics(RenderDevice*rd, Array<PosedModelRef>& posed3D, Array<PosedModel2DRef>& posed2D) | Controla el render de la aplicación además de invocar al método onPose para las entidades que lo implementen. |
| onEvent(const GEvent& e) | Controla los eventos de la aplicación. |
| onUserInput(UserInput* ui) | Controla las entradas del usuario de la aplicación, las entradas pueden ser por teclado o por el mouse. |
| setLoad(const std::string& filename) | Decide cuál estrategia usar para cargar los diferentes archivos válidos para la aplicación. |

| | |
|---|---|
| SaveTrack(std::string filename) | Guarda el fichero trk, el cual que contiene el grafo de caminos editado previamente. |

**Tabla 20** Descripción de la clase controladora App.

# Capítulo 5. Implementación y Resultados

Este capítulo aborda el modelo de la implementación de la aplicación basado en los análisis realizados en los capítulos anteriores, el flujo de trabajo de implementación se hace con el objetivo de:

- Definir la organización del código. Teniendo en cuenta los subsistemas de implementación organizadas por capas.
- Implementar los elementos de diseño en términos de ficheros fuentes, binarios, ejecutables y otros.
- Integrar los componentes desarrollados y generar un ejecutable o producto final.

Al finalizar la implementación deben quedar plasmados todos los requisitos recogidos en la fase de Requerimientos y además está fuertemente regido por el flujo de Análisis y Diseño**.**

En este capítulo también se hará un análisis de los resultados del sistema, en cuanto a rendimiento, cumplimiento de los objetivos propuestos.

## 5.1.Diagrama de Componentes

Las clases resultantes del análisis y el diseño se hacen físicas mediante componentes. A continuación se muestra como se agruparon en paquetes todos los componentes según las clases que estos contienen.

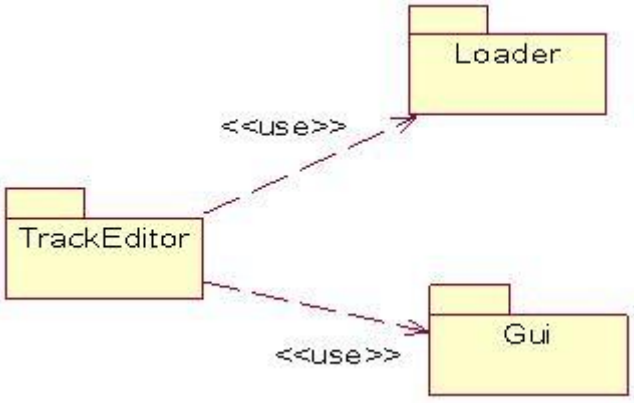


**Figura 42** Paquetes del Diagrama de Componentes.

### 5.1.1.Componentes del Paquete de Implementación Loader

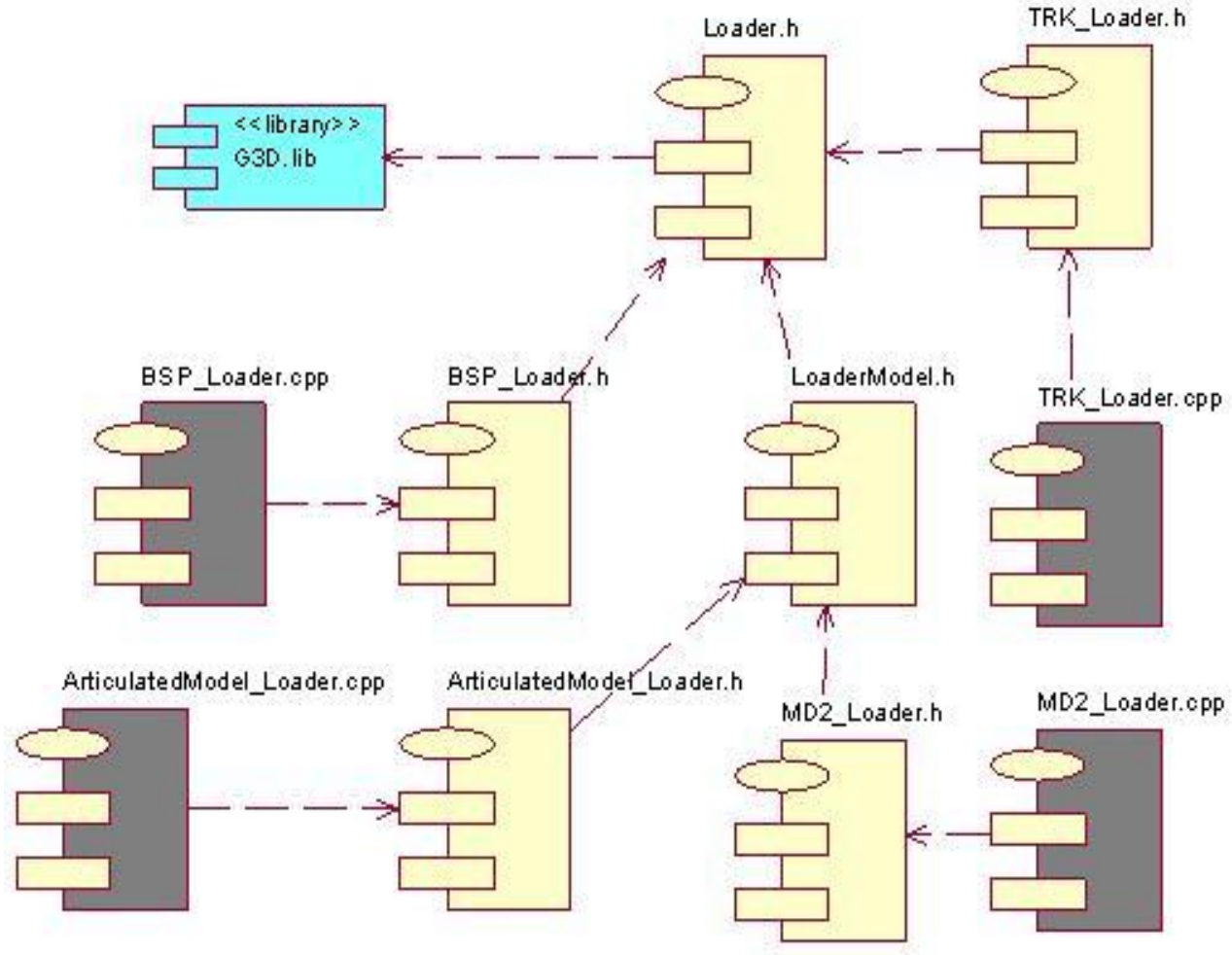


**Figura 43** Diagrama de Componentes del Paquete Loader.

### 5.1.2.Componentes del Paquete de Implementación GUI

Este módulo se encarga de controlar todos los componentes gráficos usados en la aplicación. Su descripción no es propósito de la tesis. No obstante está justificado ya que G3D 7.0 presenta unos controles GUI, los cuales se encuentran en su versión BETA. Este módulo es simplemente una ayuda a los controles de G3D 7.0 para incorporarle los eventos y además hacer el control de Diseño. Este Subsistema está basado en un modelo de Delegados, los cuales están contemplados en este módulo, el subsistema de Delegados es bastante pequeño y su uso como se ha venido explicando es para soportar el modelo de delegados.

### 5.1.3.Componentes del Paquete de Implementación TrackEditor

Este subsistema es el corazón de la aplicación y es el encargado de inicializar la aplicación e integrar todos lo sistemas en él.

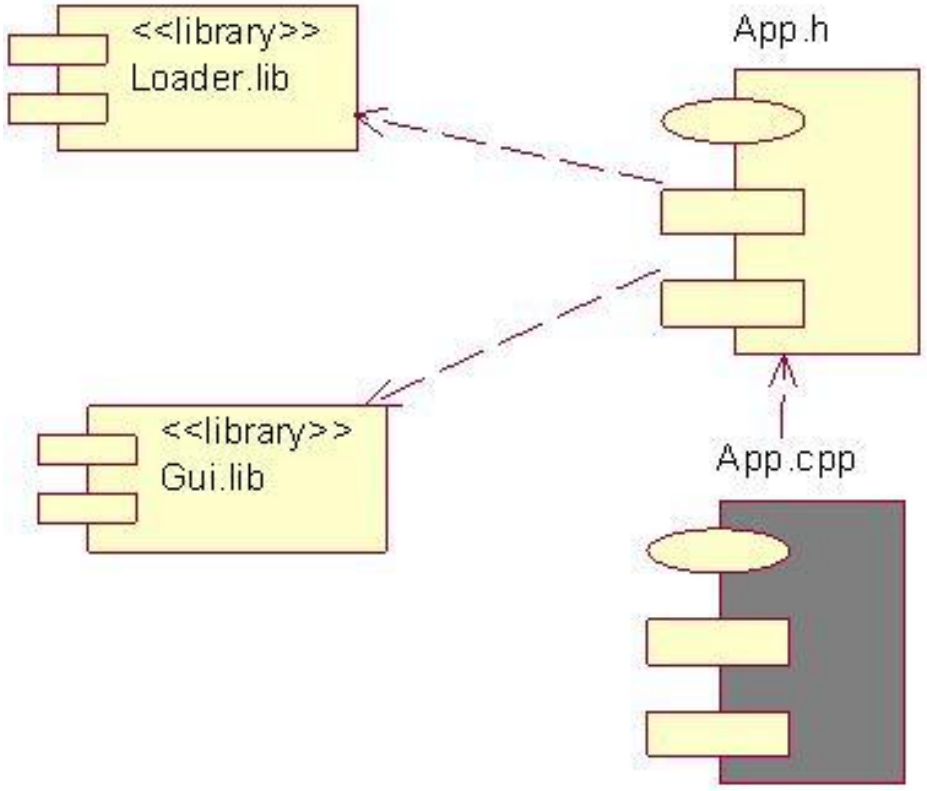


**Figura 44** Diagrama de Componentes del Paquete TrackEditor.

## 5.2.Diagrama de Subsistemas

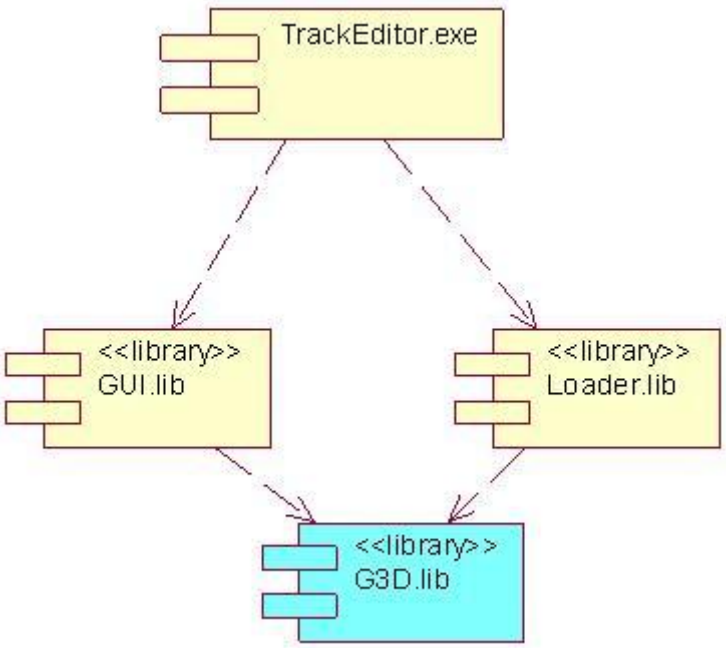


**Figura 45** Diagrama de Subsistemas.

## 5.3.Diagrama de Despliegue

El producto final es una aplicación desktop que solo es usada por el Diseñador de Pistas de Carreras y estaría instalado en una sola máquina, por esta razón quedó decidido que no es necesario representar un diagrama de despliegue.

## 5.4.Estándares de codificación

Los estándares de codificación son reglas específicas a una lengua que reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores. Los estándares de codificación no destapan problemas existentes, evitan más bien que los errores ocurran. Los bugs frecuentes en programas pueden ser detectados mucho antes o pueden ser incluso evitados totalmente. Durante el desarrollo, los estándares de codificación ayudan a los ingenieros a producir un código de alta calidad y a entender y utilizar el código de sus colegas. Pero también realzan considerablemente la capacidad de mantenimiento y rehúso a largo plazo del producto final. Tal práctica del control de bugs en el proceso del desarrollo mejora la calidad mientras que reduce el tiempo de desarrollo, el costo y el esfuerzo.

### 5.4.1.Reglas de codificación

**Constantes**

***Regla*:** Los nombres de constantes se escriben con letras en mayúscula. En caso de estar compuesta por más de una palabra, éstas se separan por un carácter „_".

```
#define SCALE        0.45f

#define SCALE_MODEL 0.6f
```

**Variables**

***Regla*:** Las variables se escriben con minúsculas, las variables con nombres compuestos, comienzan con la primera palabra enteramente en minúscula y el resto comenzando con mayúscula.

```
TrackGraph*      trackGraph

CoordinateFrame cframe
```

**Parámetros (argumentos) de métodos**

***Regla*:** Cumplen con la misma regla de las variables.

```
void setPosition(CoordinateFrame newcframe)
```

**Clases**

***Regla*:** Los nombres de clases comienzan con mayúsculas, con las palabras que la forman en minúsculas y separadas por mayúsculas.

```
class LoaderModel
```

**Métodos miembros de clases**

***Regla*:** Los métodos miembros de clases siguen la notación idéntica que para las variables, es decir, comenzando con minúscula y separando las palabras con mayúsculas.

```
virtual void onSimulation(RealTime dt)
```

**Nombres de enumerados**

***Regla*:** El nombre del tipo de enumerador cumple con la regla de las clases, y los valores con la misma notación que las constantes.

```
enum TrackNodeType
{
      NT_START,
      NT_NORMAL,
      NT_PARTIAL,
      NT_END
};
```

**Comentarios**

***Regla:*** Se usan los comentarios en línea para facilitar la comprensión del código, sobre todo en procedimientos complejos.

```
// esto es un comentario.
```

## 5.5.Resultados

Con el desarrollo del Editor de Pistas de Carreras propuesto, se obtuvo como resultado una herramienta que permite que el proceso de edición de los circuitos de carreras del proyecto Juegos Consola se realicen de forma fácil, con un mínimo de conocimientos, permitiendo a su vez que los usuarios del juego personalicen sus propias pistas de carreras.

A continuación se muestran 3 imágenes que representan las principales funcionalidades del Editor de Pistas implementado.

En la primera figura se muestra el proceso de edición de la pista de carreras, el cual se hace utilizando la interfaz de usuario lo que hace mucho más fácil este proceso para los diseñadores de las mismas.

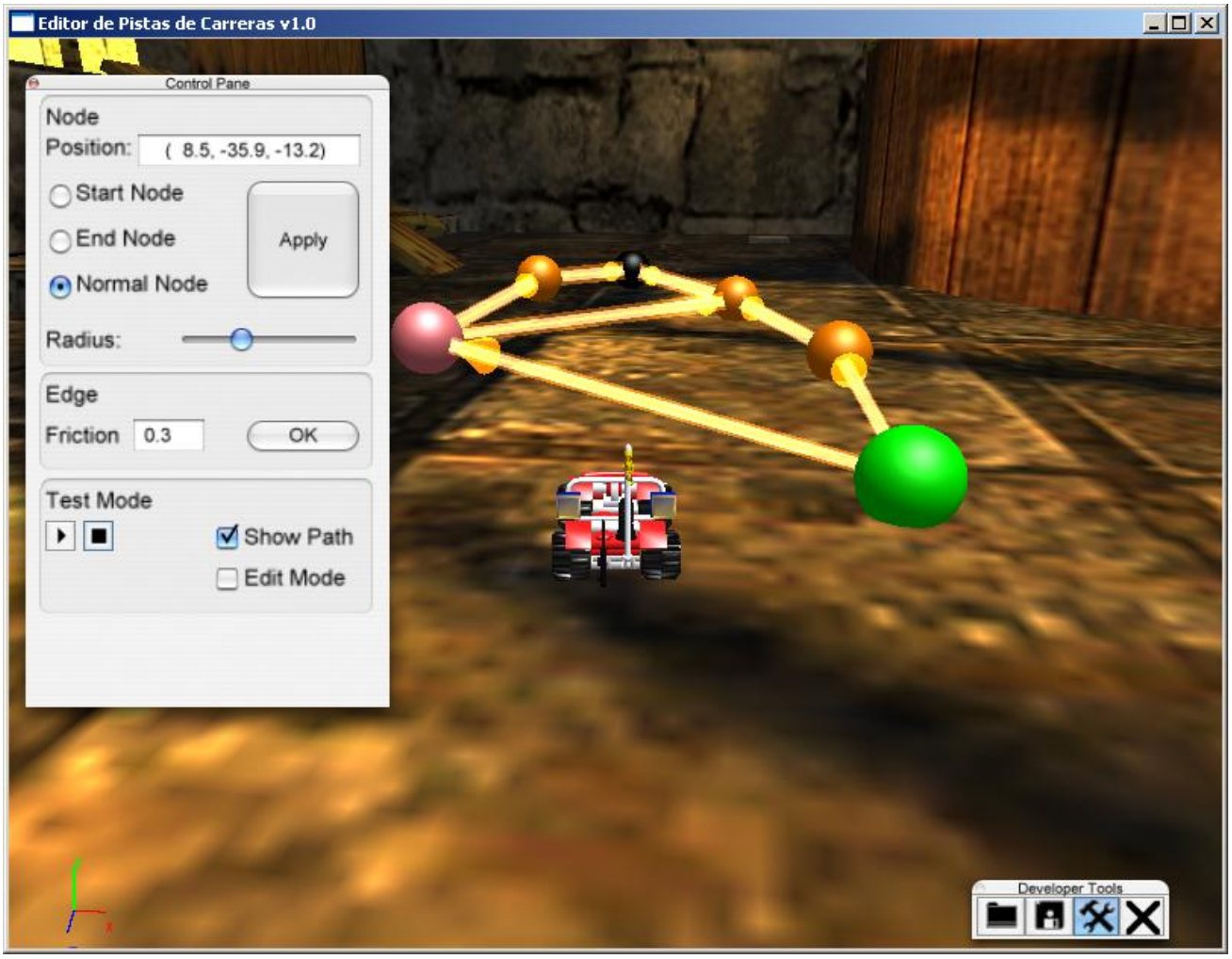


**Figura 46** Proceso de Edición de una pista de carreras.

En la figura 47 se muestra una foto del usuario probando de forma manual el circuito editado previamente, en la imagen tomada el usuario había violado el sentido de la pista, y el sistema indicó el incumplimiento con un mensaje de Wrong Way.

La figura 48 muestra al usuario haciendo uso del modo de prueba automático de la pista editada, el sistema como su nombre lo dice, es el que asume el control del modelo y realiza un recorrido por todos los caminos del grafo de caminos, para que el usuario vea cuantos caminos posibles existen en el circuito, el sistema le asigna un color a cada camino y representa a los mismos mediante un spline.


**Figura 47** Proceso de Prueba de la Pista Editada, modo manual.


**Figura 48** Proceso de Prueba de la Pista Editada, modo automático.

# Conclusiones

La realización de este Trabajo de Diploma cumplió con los objetivos propuestos al desarrollar un Editor de Pistas que permite editar un circuito de carreras sobre un mapa BSP, donde se puede definir reglas de tránsito, además de contar con un medio de comprobación de las mismas, así como guardar el proceso de edición en un fichero binario propietario.

Luego de terminado el estudio de forma general y definir el dominio se comenzó con la captura de requisitos, a través de los cuales se realizaron los cinco casos de uso directrices del desarrollo de la aplicación. Con estos artefactos se comenzaron a elaborar los primeros diseños del modelo, que gradualmente fue mejorado con el estudio y aplicación de patrones de diseño.

# Recomendaciones

- Continuar el desarrollo del software con la incorporación de otros formatos de mapas para cargar y editar la pista de carreras.

- Incorporar nuevas funcionalidades al editor de pistas de carreras, por ejemplo agregarle al mapa objetos como semáforos, señales de tránsito, vallas, entre otros.

- Ampliar la información modificable de las aristas del Grafo de Caminos.

# Bibliografía

## Referenciada

**[1]**. **Morris, Chris.** Video game sales jump 8 percent in 2004 . *CNN, Money.com.* [En línea] diciembre de 2007. http://money.cnn.com/2005/01/18/technology/gamesales/.

**[2]**. **Guerenabarrena, Ines y Llano, Iurgi.** TIPOS DE VIDEOJUEGOS Y HABILIDADES. *Mundo Hogar.* [En línea] diciembre de 2007 http://www.mundogar.com/ideas/ficha.asp?ID=7841.

**[3]**. **Marmad.** Los tipos de videojuegos que existen. *Gamers.* [En línea] enero de 2008. http://www.gamers.com.mx/noticias/12556_Los_tipos_de_videojuegos_que_existen.html.

**[4]**. **Edgewall.**qeradiant.com. *trac, Integrated SCM & Project Management.* [En línea] febrero de 2008. http://www.qeradiant.com/cgi-bin/trac.cgi.

**[5]**. **F. Engel, Wolfgang, Geva, Amir y LaMothe, André.** Chapter 9: Working with Files. *Beginning Direct3D Game Programming.*

**[6]**. **García López, Wendy y Hechemendía González, Alexis.** Sistema de Generación de Ficheros para Entornos Virtuales. Ciudad Habana : s.n., 2007.

**[7]**. **McGuire, Morgan.**BSP technical details for Dummies. [En línea] febrero de 2008. http://qxx.planetquake.gamespy.com/bsp.

**[8]**. **Pipho, Evan.** Chapter 7: The 3DS Models. *Focus on 3D Models.*

**[9]**. **Pipho, Evan** Chapter 3: Quake ll`s MD2 Models. *Focus on 3D Models.*

**[10]**. **Hawkins, Kevin y Astle, Dave.** Chapter 20 - Building a Game Engine. *OpenGL Game Programming.*

**[11]**. **DevMaster.net.**3D Engines Database. *DevMaster.net.* [En línea] diciembre de 2007. http://www.devmaster.net/engines/list.php?start=30&fid=14&sid=0.

**[12]**. **Skilton, Frank.** 3D Engines Database. *DevMaster.net.* [En línea] diciembre de 2007. http://www.devmaster.net/engines/.

**[13]**. **McGuire, Morgan.** *G3D Manual and Library source code.* [En línea] diciembre 2007. http://g3d-cpp.sourceforge.net/html/index.html.

**[14]**. **Proudfoot, Kekoa.** Unofficial Quake 3 Map Specs. [En línea] abril de 2008. http://graphics.stanford.edu/~kekoa/q3/.

**[15]**. **Larman, Craig.** *PARTE II FASE DE PLANEACION Y DE ELABORACION. UML y Patrones. Introducción al Anílisis y Diseño Orientado a Objeto. Prentice Hall* 1999.

**[16]**. **Larman, Craig.** *PARTE IV FASE DE DISEÑO. UML y Patrones. Introducción al Anílisis y Diseño Orientado a Objeto. Prentice Hall* 1999.

**[17]**. **Gamma, Erich, y otros.** *Design Patterns. Elements of Reusable Object-oriented Software.* 1998

## Consultada

-.**Hernández León, Rolando Alfredo y Coello González, Sayda.** *El Paradigma Cuantitativo de la Investigación Científica.* Ciudad de la Habana : Universidad de las Ciencias Informáticas : s.n., 2002.

-.**Henry, David.** MD2 file format(Quake 2's models). [En línea] diciembre de 2004. http://tfc.duke.free.fr/coding/md2-specs-en.html.

-.**Astle, Dave y Hawkins, Kevin.** *More OpenGL Game Programming.* 2005.

-.**Booch, Grady.** *Object-oriented analysis and design with applications 2da Edición.* Santa Clara, California : s.n., 1998.

-.**Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *Proceso Unificado de Desarrllo de Software.* 2000.

-.**Zerbst, Stefan y Duvel, Oliver.** *3D Game Engine Programing.*

**-.DeLoura, Mark.** *Game Programming Gems 2.* 2002.

# Glosario de Términos y Siglas

**A:**

**Algoritmo:** Conjunto finito de pasos o instrucciones que se deben seguir para realizar una determinada tarea.

**Aproximación:** la curva se ajusta a los puntos de control.

*****

**B:**

**Bípedo:** No es más que el animal que utiliza dos extremidades para desplazarse. En la Realidad Virtual es el esqueleto que cumple la característica anterior y que se utiliza en las animaciones.

***Bug:*** Error en la escritura de un programa lo que produce un defecto en el sistema.

*****

**C:**

**CPU:** Acrónimo utilizado para abreviar Central Processing Unit que significa Unidad de Procesamiento Central. Parte de un computador que controla todas las demás partes.

*****

**D:**

**Disco Duro:** Se llama disco duro o disco rígido al dispositivo encargado de almacenar información de forma permanente en una computadora.

**DFS:** DEPTH-FIRST SEARCH, siglas en ingles, es un algoritmo usado en teoría de Grafos para la búsqueda de todos los caminos de un nodo inicial a un nodo final.

**Drag and Drop:** (arrastrar y soltar) es un término muy usado en informática que se refiere a la acción de arrastrar y soltar con el ratón objetos de una ventana a otra o entre partes de una misma

ventana o programa. Los objetos arrastrados son habitualmente archivos, pero también pueden ser arrastrados otros tipos de objetos en función del programa.

*****

**E:**

**Entorno Virtual:** Referido a Mundo virtual. Simulación de mundos o entornos denominados virtuales en los que el hombre interacciona con la máquina en entornos artificiales semejantes a la vida real. Ejemplo de aplicaciones desarrolladas sobre mundos virtuales son los simuladores y los videojuegos.

**Engine 3D:** Conocido también como Game Engine (Motor de Juego), es el software principal (core o núcleo) de un videojuego.

**Editor de Pistas de Carreras:** Herramienta usada para definir el grafo de caminos (ver Grafo de Caminos) de un entorno virtual.

*****

**F:**

***Frame:*** cada uno de las imágenes que componen una animación.

**Fichero:** Bloque lógico de información considerado como una unidad por el usuario. Todos los datos que se almacenan en el ordenador se denominan ficheros. El ordenador puede guardar textos, imágenes, piezas musicales, sonidos y demás. Según su contenido, se almacenan con distintos formatos.

**Fichero *.TRK:** Fichero binario en el que se almacena la información del grafo de caminos editado en el Editor de Pistas.

*****

**G:**

**Grafo:** Es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas.

**Grafos de Caminos:** Grafo que se usa en entornos virtuales para brindar información acerca de la estructura del entorno y para la aplicación de algoritmos de inteligencia artificial como la búsqueda de caminos.

**G3D:** Graphics Three Dimensional. Engine 3D disponible como código abierto (del inglés Open Source) y libre (del inglés free) con licencia BSD desarrollado en el lenguaje de programación C++. G3D Proporciona una base sólida y altamente optimizada para el desarrollo en general de aplicaciones gráficas y extiende las funcionalidades de OpenGL y sockets incluyendo formatos de modelos 3D y bibliotecas utilitarias.

**GPU*:*** Acrónimo utilizado para abreviar Graphics Processing Unit, que significa unidad de procesado de gráficos. Procesador dedicado exclusivamente al procesamiento de gráficos, para aligerar la carga de trabajo del procesador central en aplicaciones 3D interactivas como los videojuegos.

*****

**H:**

***Hardware gráfico:*** Dispositivos necesarios para crear aplicaciones gráficas.

**Heurística:** Regla que permite orientar un algoritmo hacia la solución de un problema. Técnica de programación que permite a un sistema la creación gradual de un valor óptimo para una variable específica por medio del registro de los valores obtenidos en operaciones anteriores. Técnica empleada en los sistemas de inteligencia artificial.

**Herencia:** Es uno de los mecanismos de la programación orientada a objetos, por medio del cual una clase se deriva de otra de manera que extiende su funcionalidad. Una de sus funciones más importantes es la de proveer el Polimorfismo (ver concepto de polimorfismo).

*****

**I:**

**Interfaz de usuario:** Es la parte del programa informático que permite el flujo de información entre varias aplicaciones o entre el propio programa y el usuario.

***Interpolación:*** algoritmo matemático que a partir de varios puntos en el espacio, describe una función que contiene a los puntos intermedios.

**Informática:** Es la ciencia que estudia el tratamiento automático y racional de la información.

*****

**J:**

**John D. Carmack II:** Figura ampliamente reconocida en la industria de los videojuegos. Co-fundó id Software, una empresa de desarrollo de videojuegos, en 1991. Creador de los primeros juegos de acción en primera persona (First Person Shooter o FPS), Wolfenstein 3D, Doom y Quake, entre otros.

*****

**K:**

**Knots:** Son los puntos de enlace entre los distintos trozos de la curva se les suele llamar nudos contando como tales los puntos de inicio y fin de la curva.

*****

**L:**

**Lighting:** La iluminación es la acción o efecto de iluminar. En la técnica se refiere al conjunto de dispositivos que se instalan para producir ciertos efectos luminosos, tanto prácticos como decorativos. Con la iluminación se pretende, en primer lugar conseguir un nivel de iluminación, o iluminancia, adecuado al uso que se quiere dar al espacio iluminado.

*****

**M:**

**Memoria:** Se refiere a una forma de almacenamiento de estado sólido conocido como memoria de acceso aleatorio (RAM por sus siglas en inglés). Ver concepto de RAM.

**Mesh:** Término en ingles que significa malla que no es más que la forma de representar un modelo 3D a partir de polígonos.

***Modelo:*** Prototipo para la animación.

*****

**N:**

**Nodo:** Estructura que representa una posición en el mapa o en el Grafo.

***Normalizar*:** Transformar coordenadas o parámetros que tengan un valor predeterminado.

*****

**O:**

**OpenGl:** Open Graphics Library, Biblioteca gráfica 3D desarrollada por Silicon Graphics Incorporated.

*****

**P:**

**Plataforma:** Combinación de hardware y software usada para ejecutar aplicaciones.

**Polimórfico:** Se define un objeto como polimórfico a la entidad que puede contener valores de diferentes tipos durante la ejecución del programa.

**Puntos de control:** Conjunto de puntos que indican la forma general de la curva.

*****

**Q:**

**Quake:** Juego de acción en primera persona que fue publicado por id Software el 31 de mayo de 1996.

*****

**R:**

**RAM:** Se trata de una memoria de semiconductor en la que se puede tanto leer como escribir información.

**Realidad Virtual:** Simulación de un medio ambiente real o imaginario que se puede experimentar visualmente en tres dimensiones. La realidad virtual puede además proporcionar una experiencia interactiva de percepción táctil, sonora y de movimiento.

**Render *ó* Renderización:** Es el proceso de generar una imagen desde un modelo. La palabra renderización proviene del inglés render. En términos de visualizaciones en ordenador, más específicamente en 3D, la "renderización" es un proceso de cálculo complejo desarrollado por un ordenador destinado a generar una imagen 2D a partir de una escena 3D.

**RUP*:*** Rational Unified Process (Proceso Unificado de Desarrollo). Metodología para el desarrollo de Software.

**RayTracing:** Técnica que permite generar imágenes tridimensionales por ordenador mucho más realistas que las generadas por los actuales motores de juegos.

*****

**S:**

**Serialización:** Proceso de codificación de un Objeto (programación orientada a objetos) en un medio de almacenamiento (como puede ser un archivo, o un buffer de memoria) con el fin de transmitirlo a través de una conexión en red como una serie de bytes.

**Spline:** Curva definida a trozos mediante polinomios, en los problemas de interpolación, se utiliza a menudo la interpolación mediante Splines.

**Shaders:** Conjunto de instrucciones gráficas destinadas para el acelerador gráfico que definen el aspecto final de un objeto. Estos determinan materiales, efectos, color, luz, sombra y etc.

*****

**T:**

**Tarjeta gráfica:** Es una tarjeta de circuito impreso encargada de transformar las señales eléctricas que llegan desde el microprocesador en información comprensible y representable por la pantalla del ordenador.

**TrackEditor:** Software utilizado en la Realidad Virtual para editar los grafos de caminos de los mapas de la misma.

*****

**U:**

**UML:** Unified Languajes Process (Lenguaje Unificado de Procesos). Es una notación estándar para modelar objetos del mundo real como primer paso en el desarrollo de programas orientados a objetos. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema de software.

*****

**V:**

**Videojuego:** Programa informático creado para el entretenimiento, basado en la interacción entre una o varias personas y un aparato electrónico llamado consola que ejecuta el videojuego.

*****

**Z:**

**Z-Buffer:** En los gráficos por computadora, el z-buffering es la parte de la memoria de un adaptador de video encargada de gestionar las coordenadas de profundidad de las imágenes en los gráficos en tres dimensiones (3D).

*****

**Índice de Tablas**

Tabla 1 Características de G3D Engine. ....16
Tabla 2 Tipos de dato en un archivo *TRK. ....25
Tabla 3 Estructura del Bloque Nodes del fichero TRK. ....25
Tabla 4 Estructura del Bloque Edges del fichero TRK. ....26
Tabla 5 Actor del Sistema. ....32
Tabla 6 CU1 Cargar Mapa BSP. ....32
Tabla 7 CU2 Cargar Modelos. ....32
Tabla 8 CU3 Editar Pista de Carreras. ....33
Tabla 9 CU4 Activar Modo Exploración. ....33
Tabla 10 CU5 Gestionar Fichero con Trayectoria de la Pista de Carrera. ....33
Tabla 11 Descripción CU Cargar Mapa BSP. ....35
Tabla 12 Descripción CU Cargar Modelos. ....36
Tabla 13 Descripción CU Editar Pista de Carreras. ....41
Tabla 14 Descripción CU Activar Modo Exploración. ....42
Tabla 15 Descripción CU Gestionar Fichero con Grafo de Caminos. ....43
Tabla 16 Descripción de la sección Cargar Modelo MD2. ....43
Tabla 17 Descripción de la sección Cargar Modelo 3DS. ....44
Tabla 18 Descripción de la sección Guardar Fichero .trk. ....45
Tabla 19 Descripción de la sección Cargar Fichero .trk. ....46
Tabla 20 Descripción de la clase controladora App. ....70

**Índice de Figuras**

Figura 1 Editor de Niveles – GtkRadiant. ....7

Figura 2 Mensaje de Trayectoria Incorrecta. ....8

Figura 3 Editor de Pistas de Carreras. ....9

Figura 4 Creación de un Árbol BSP. ....10

Figura 5 Estructura del fichero BSP. ....11

Figura 6 Estructura del fichero 3DS. ....12

Figura 7 Estructura del fichero MD2. ....13

Figura 8 Arquitectura de G3D Engine. ....14

Figura 9 Animal Race: Videojuego producido por la Universidad de Ulm, usando G3D, ODE, y Animadead. ....16

Figura 10 Proceso de edición de una pista de carreras. ....21

Figura 11 Dinámica de un Editor de Pistas de Carreras. ....22

Figura 12 Grafo de Caminos de una región. ....23

Figura 13 Estructura del fichero TRK. ....24

Figura 14 Modelo de Dominio. ....28

Figura 15 Diagrama de Casos de Uso del Sistema. ....34

Figura 16 Menú Developer Tools. ....47

Figura 17 Menú Open File. ....47

Figura 18 Menú Save Track. ....48

Figura 19 Menú Control Pane. ....49

Figura 20 Diagrama de Clases del Análisis de CU Cargar Mapa BSP. ....51

Figura 21 Diagrama de Clases del Análisis de CU Cargar Modelo. ....52

Figura 22 Diagrama de Clases del Análisis de CU Editar Pista de Carreras. ....52

Figura 23 Diagrama de Clases del Análisis de CU Activar Modo Exploración. ....52

Figura 24 Diagrama de Clases del Análisis de CU Gestión de Fichero TRK. ....53

Figura 25 Diagrama de colaboración del análisis CU Cargar Mapa BSP ....54

Figura 26 Diagrama de colaboración del análisis del CU Cargar Modelos. Sección Cargar Modelo MD2. ....54

Figura 27 Diagrama de colaboración del análisis del CU Cargar Modelos. Sección Cargar Modelo 3DS.....54

Figura 28 Diagrama de colaboración del análisis para el CU Editar Pista de Carreras.....55

Figura 29 Diagrama de colaboración del análisis para el CU Activar Modo Exploración. .....55

Figura 30 Diagrama de colaboración del análisis del CU Gestión de Fichero con Trayectoria de la Pista de Carrera. Sección Guardar Fichero .TRK.....56

Figura 31 Diagrama de colaboración del análisis del CU Gestión de Fichero con Trayectoria de la Pista de Carrera. Sección Cargar Fichero .TRK.....56

Figura 32 Diagrama General de Paquetes de Clases del Diseño.....59

Figura 33 Diagrama de Clases del Diseño del Paquete Loader**.** .....60

Figura 34 Diagrama de Clases del Diseño del Paquete TrackEditor. .....61

Figura 35 Diagrama de Secuencia del Diseño del CU Cargar Mapa BSP. .....62

Figura 36 Diagrama de Secuencia del Diseño del CU Cargar Modelo. Sección Cargar Modelo MD2. .....63

Figura 37 Diagrama de Secuencia del Diseño del CU Cargar Modelo. Sección Cargar Modelo 3DS. .....63

Figura 38 Diagrama de Secuencia del Diseño del CU Editar Pista de Carreras. Sección Crear Nodo. .....64

Figura 39 Diagrama de Secuencia del Diseño del CU Editar Pista de Carreras. Sección Eliminar Nodo. .....65

Figura 40 Diagrama de Secuencia del Diseño del CU Editar Pista de Carreras. Sección Modificar Información de un Nodo.....66

Figura 41 Diagrama de Secuencia del Diseño del CU Editar Pista de Carreras. Sección Modificar Información de una Arista. .....67

Figura 42 Paquetes del Diagrama de Componentes.....71

Figura 43 Diagrama de Componentes del Paquete Loader.....72

Figura 44 Diagrama de Componentes del Paquete TrackEditor.....73

Figura 45 Diagrama de Subsistemas. .....74

Figura 46 Proceso de Edición de una pista de carreras. .....77

Figura 47 Proceso de Prueba de la Pista Editada, modo manual.....78

Figura 48 Proceso de Prueba de la Pista Editada, modo automático. .....79