ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

FÚTBOL ES ASÍ

Alberto Cejas Sánchez

16 de febrero de 2013

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

Fútbol Es Así

- Departamento: Ingeniería Informática
- Director del proyecto: Manuel Palomo Duarte
- Autor del proyecto: Alberto Cejas Sánchez

Cádiz, 16 de febrero de 2013

Fdo: Alberto Cejas Sánchez

Agradecimientos

Aunque el desarrollo ha sido largo y por momentos tedioso, agradecer al tutor Manuel Palomo por la motivación necesaria a adentrarme a un proyecto de esta envergadura, del cual sin duda he aprendido bastante. Debo mencionar también a Mat Buckland, autor de [1], el libro que he usado como base para todo este proyecto.

Licencia

Notación y formato

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

Ogre.

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

```
print "Hola mundo"
```

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

```
sudo apt-get install
```

Índice general

Indice de figuras

Capítulo 1

Introducción

1.1. Motivación

La idea de desarrollar un videojuego de fútbol auna mis dos intereses principales: la informática y el balompié. Además de incorporar un componente creativo y paralelamente iniciarme un poco en el mundo artístico del diseño 3D.

También he de añadir que tras conocer abiertamente el mundo del Software libre, gracias a la importancia que se le presta en la Universidad de Cádiz. Se decidió que el proyecto fuera software libre bajo licencia GPL v3. Y así cualquier persona interesada en el desarrollo de videojuegos y en el software libre en general, pudiera usar los recursos del proyecto libremente.

1.2. Objetivos

El objetivo principal del proyecto es realizar un videojuego sencillo de fútbol en tres dimensiones, que me permita aprender los distintos campos que componen el desarrollo de videojuegos y las simulaciones.

Los destinatarios principales del videojuego son los jugadores casuales, que dedican poco tiempo jugando, debido a que carece de trama argumental u objetivos/trofeos.

1.3. Estructura del documento

Este documento esta compuesto por las siguientes partes:

- **Introducción**: pequeña descripción del proyecto, así como los objetivos y estructura del documento.
- **Descripción general**: descripción más amplia sobre el proyecto, así como todas las características relevantes que tendrá.
- **Planificación**: exposición de la planificación del proyecto y las distintas etapas que esta compuesto el mismo.
- **Análisis**: fase de análisis del sistema, empleando la metodología seleccionada. Se definirán los requisitos funcionales del sistema, diagramas de caso de uso, diagramas de secuencia y contrato de las operaciones.
- **Diseño**: realización del diseño del sistema, diagramas de secuencia y clases aplicadas al diseño.

- **Implementación**: aspectos más relevantes durante la implementación del proyecto. Y problemas que han aparecido durante el desarrollo de este.
- **Pruebas y validaciones**: pruebas realizada a la aplicación, con el fin de comprobar su correcto funcionamiento y cumplimiento de las expectativas.
- **Conclusiones**: conclusiones obtenidas tras el desarrollo de la aplicación.
- **Apéndices**:
  - **Herramientas utilizadas**: explicación de todas las herramientas usadas a lo largo del desarrollo del proyecto.
  - **Manual de instalación**: manual para la correcta instalación del proyecto en el sistema.
  - **Manual de usuario**: manual de usuario para el correcto uso de la aplicación. nuevos estados a los futbolistas.
- **Bibliografía**: libros y referencias consultado durante el desarrollo del proyecto.
- **Licencia GNU GFDL**: texto completo sobre la licencia GNU GFDL.

Capítulo 2

Descripción general del proyecto

2.1. Descripción

El proyecto consiste en un juego sencillo y dinámico de fútbol en tres dimensiones, con equipos de 5 vs 5, en el que el jugador deberá competir contra el ordenador.

Figura 2.1: Descripción: Logo FEA

2.2. Características del videojuego

El videojuego ofrece una alternativa libre y gratuita para jugar a un juego de fútbol en tres dimensiones, teniendo en cuenta que en los repositorios oficiales de distribuciones UNIX como Ubuntu no existe ninguna opción.

El único modo de juego disponible es jugador contra ordenador, que auna la complejidad de ambos y su interacción en el flujo de la ejecución del videojuego. Este modo de partido rápido tendrá una duración de 5 minutos.

2.2.1. Elementos de juego

En esta sección se hará una pequeña descripción de los distintos elementos que encontraremos a lo largo del juego.

Futbolistas

Los elementos básicos del juego, son los encargados de interactuar entre sí y con el balón. Dependiendo de si pertenecen al equipo local o al equipo visitante emplearán un uniforme u otro y serán manejados por el jugador o el ordenador. Cada uno podrá tener características propias tales como la velocidad o la potencia de golpeo.

Figura 2.2: Descripción: Futbolista de FEA.

Balón

Un elemento básico en el fútbol y que en este proyecto podrá interactuar con los futbolistas y con el entorno, reaccionando cuando sale del campo, entra en la portería o choca con un poste de ésta.

Figura 2.3: Descripción: Balón FEA (modelos High y Low Poly).

2.3. Colaboradores

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

Desarrollador Alberto Cejas Sánchez

Diseñador Gráfico Erik Castillo, Alberto Cejas Sánchez, Blender-models.com

- Terreno de juego: Erik Castillo.
- Montañas: blender-models.com (a mountain village).
- Modelado, Diseño, Animación, Texturizado futbolistas: Alberto Cejas Sánchez.
- Modelado, Texturizado balón: Alberto Cejas Sánchez.
- Modelado, Texturizado portería: Alberto Cejas Sánchez.

Música paper planes pilot(Jamendo), freesfx.co.uk

Capítulo 3

Planificación

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

3.1. Fase inicial

La primera fase consistió en plantear la idea del proyecto, con la ayuda del tutor. Tras varias propuestas y la deliberación sobre las mismas, se decidió realizar este proyecto debido a las motivaciones escritas anteriormente.

También se pensó en que lenguaje se desarrollaría el proyecto, así como las principales bibliotecas que se usarían durante la realización del mismo. Buscando siempre opciones libres y multiplataforma.

3.2. Fase de análisis

Esta etapa está dividida principalmente en las dos partes siguiente:

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

3.3. Fase Aprendizaje

Dado que el proyecto abarca un campo que hasta ahora no había tocado como es el mundo de los videojuegos, tenía que aprender los principales conceptos de éstos, así como cada una de las librerías empleadas, que a su vez incluían conceptos matemáticos y físicos. Paralelamente debía aprender a modelar, texturizar y animar en 3D.
Esta fase se extendió a lo largo de bastante tiempo y para la cual leí varios libros, intenté entender código ya escrito y ver videotutoriales acerca del manejo en general y en particular de herramientas 3D.
Por lo tanto esta fase se dividió en tres etapas principales:

- **Aprendizaje del desarrollo de videojuegos**: Al no tener experiencia previa en este campo decidí adentrarme leyendo [1], que además contenía código fuente para añadirle un punto más práctico. Y es que este libro toca muchos palos de los videojuegos, desde la arquitectura de éste para hacerla funcional y extensible como la propia inteligencia artificial con muchas ideas para aplicar sobre un juego de fútbol.

- **Aprendizaje del arte**: fue la tarea más ardua, ya que no había tenido ningún conctacto antes con los conceptos, ni con las herramientas. Para optimizar el tiempo, decidí aprender a la vez que diseñaba el arte del juego. Con lo que esta etapa se subdivide a su vez en tres:

  - **Modelado.** Se trata de la composición del personaje sobre un boceto a papel a través de polígonos, intentando usar el menor número de ellos para así, optimizar el rendimiento. Un ejemplo del proceso sobre la vista lateral de la cara del futbolista:

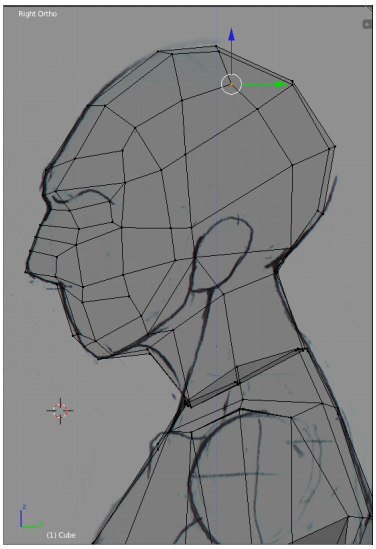


Figura 3.1: Planificación: Aprendizaje modelado (cara futbolista).

  - **Texturizado.** Este proceso a su vez se subdivide en dos partes. La primera consiste en envolver al personaje 3D sobre una manta 2D, e ir marcando con zonas cerradas qué parte del plano 2D corresponde a las zonas del cuerpo del futbolista. Como se puede apreciar en la imagen siguiente:

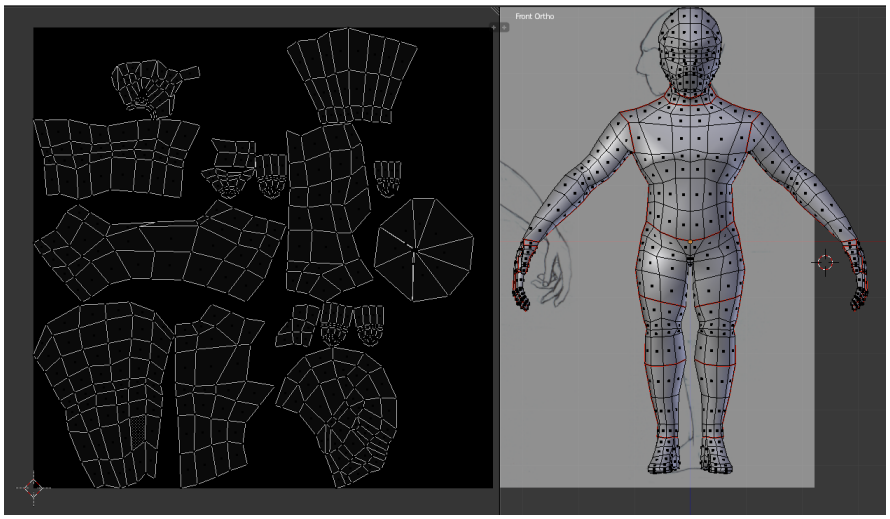


Figura 3.2: Planificación: Aprendizaje texturizado-seaming.

Una vez conocemos ya la subvdivisión de islas del plano 2D se procede al texturizado como tal:

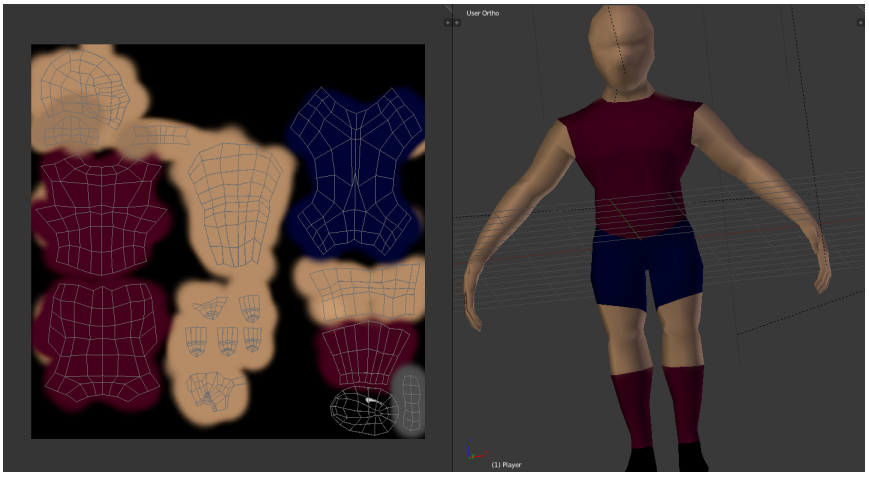


Figura 3.3: Planificación: Aprendizaje texturizado.

- **Animación.** Al igual que el anterior, este proceso a su vez se subdivide en otras dos partes. La primera consiste en crear una malla con movilidad que permita usarse como esqueleto del modelo que se ve. Además cada "hueso"del esqueleto debe tener un peso concreto en los vértices del modelo, y algunos deberán moverse en cadena.

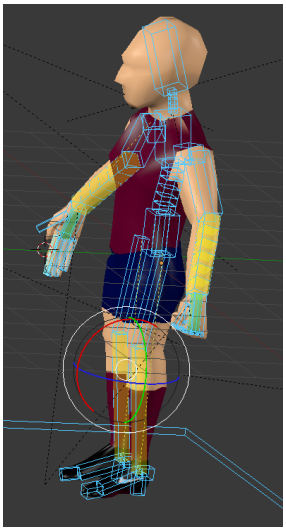


Figura 3.4: Planificación: Aprendizaje rigging.

Una vez el cuerpo reacciona de manera adecuada a los movimientos del esqueleto se crean las animaciones realizando interpolaciones entre disntos "keyframes"para cada "hueso".

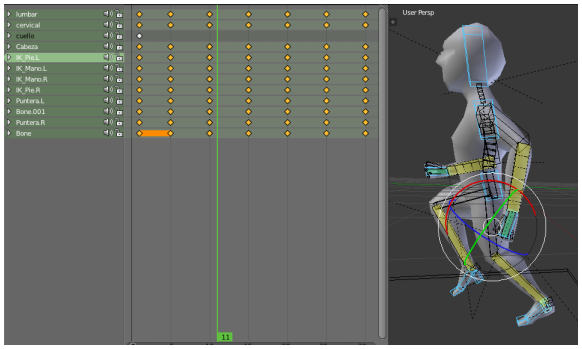


Figura 3.5: Planificación: Aprendizaje animación.

- **Familiarización con las bibliotecas**: Para llevar a cabo el desarrollo de este juego se optó por utilizar *Ogre3D* como motor gráfico, *Bullet+OgreBullet(wrapper)* como motor de físicas, *OIS* para manejar la entrada del usuario, *MyGUI* como sistema para la interfaz y *SDL-Mixer* para el sonido. Pero antes de decantarme estuve haciendo pruebas para ver qué me ofrecían con respecto a las alternativas. Para el aprendizaje de la utilización de *Ogre* leí [2] y [3].

3.4. Fase de desarrollo

Tras la consecución de las etapas anteriores, se comenzó el desarrollo del proyecto. Esta etapa del desarrollo es la más extensa de todas, como es comprensible. Y es posible llevarla a cabo gracias a que ya tenía los modelos que había empleado en el aprendizaje y alguna animación disponible.

- **Motor básico**: implementación de las necesidades básicas del proyecto, arquitectura general, máquinas de estados para el propio juego, los equipos, los jugadores, control del ratón, teclado y joystick, carga de recursos y movimiento de los futbolistas.

- **Motor de físicas y colisiones**: El empleo de un motor de físicas se me hacía imprescindible en casos como disparos a portería o pases donde influyen la gravedad y la fricción. Además de las colisiones del balón con las porterías y los futbolistas o entre éstos últimos.

- **Creación de menús e interfaz**: implementación de la breve interfaz de menús de la que está compuesto el juego (menú principal, créditos, pausa, resultado actual del partido, etc).

- **Inteligencia artificial**: posiblemente el punto más complejo del proyecto ya que tendrá comportamientos específicos de los jugadores, así como de los equipos. Dotará al juego de facilidad de ampliación.

- **Integración del sonido**: La insercción de música en el menú principal o efectos de sonido al golpear el balón.

3.5. Pruebas y correcciones

Una de las etapas más importantes, si no es la que más, del desarrollo de cualquier proyecto. Esta etapa se realizaría en paralelo a la de desarrollo, ya que conforme se implementan nuevas funcionalidades, debían ser probadas exhaustivamente en cualquiera de las posibles situaciones que pudiera suceder hasta obtener el comportamiento esperado.

3.6. Redacción de la memoria

La redacción de la memoria se ha redactado conforme se iba avanzando en el desarrollo del proyecto. Pero tras la finalización de este, se le ha dedicado más tiempo a su finalización, corrigiendo puntos que finalmente no se han adecuado al producto final.

3.7. Diagrama de Gantt

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

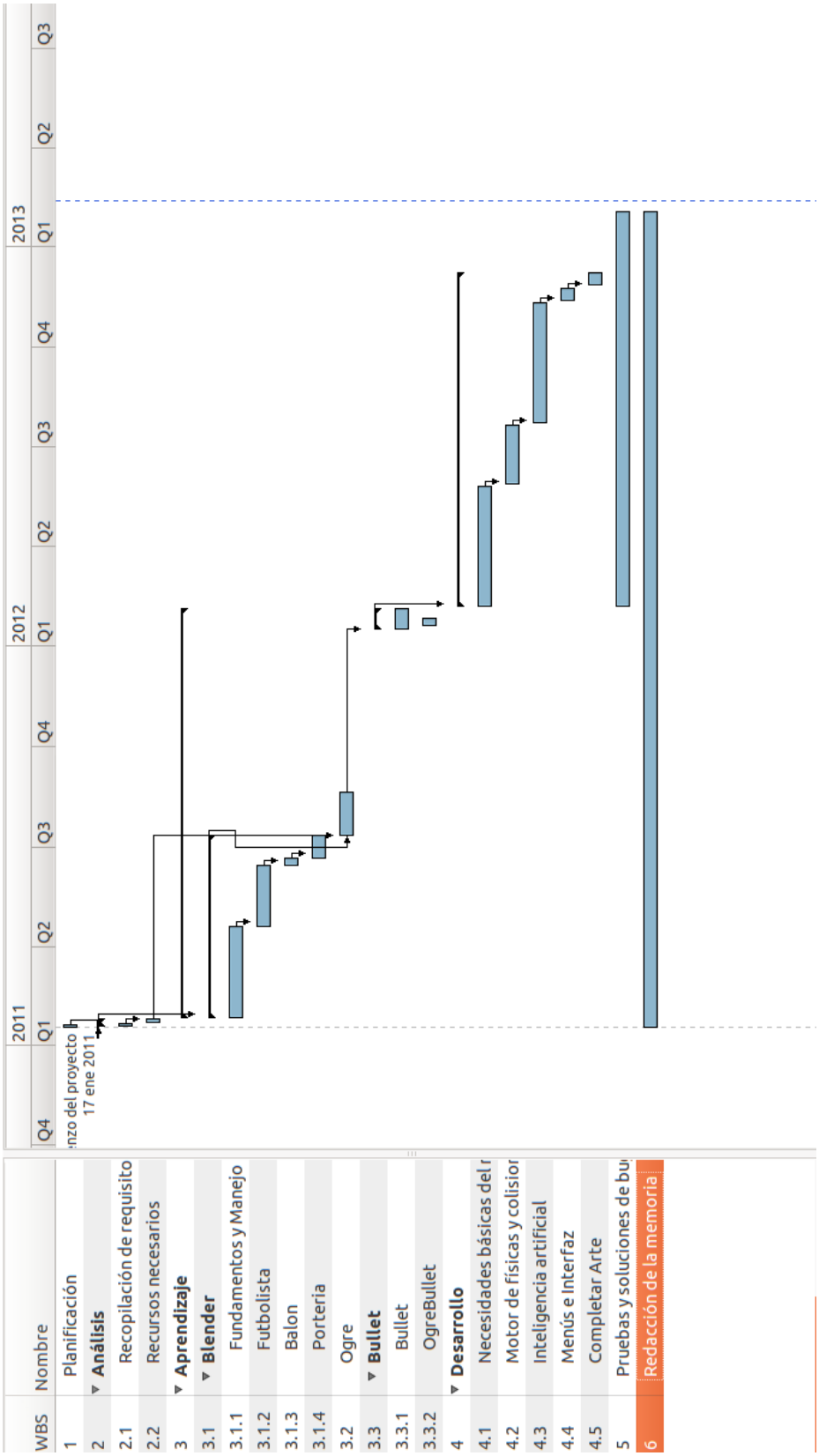


Figura 3.6: Planificación: Diagrama de Gantt.

Capítulo 4

Análisis

4.1. Especificación de requisitos del sistema

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

4.1.1. Requisitos de interfaces externas

En este apartado se describirá los requisitos de conexión del software y el hardware, así como la interfaz de usuario.
La conexión entre el software y el hardware se encarga la librería *Ogre*, un motor de renderizado en 3D, en el lenguaje de programación C++. Por lo que al ser un sistema preestablecido, no será necesario realizar el diseño, ni el análisis, sólo haremos uso de él.
Así que pasamos a definir la interfaz entre el usuario y el videojuego. Todas las ventanas de la aplicación tendrán una resolución de 800x600 píxeles a pantalla completa. En la aplicación encontraremos:

Ventana de menú principal La ventana del menú principal (figura 4.1) muestra el menú de inicio de *Fútbol Es Asi*, así como todas las opciones generales del juego disponibles, que son las siguientes:

- Jugar
- Créditos
- Salir

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

Figura 4.1: Análisis: Boceto del menú principal

Ventana de créditos En esta pantalla (figura 4.2)se mostrarán los creadores de *Fútbol Es Así*. Tendremos a nuestra disposición un botón para volver al menú principal.

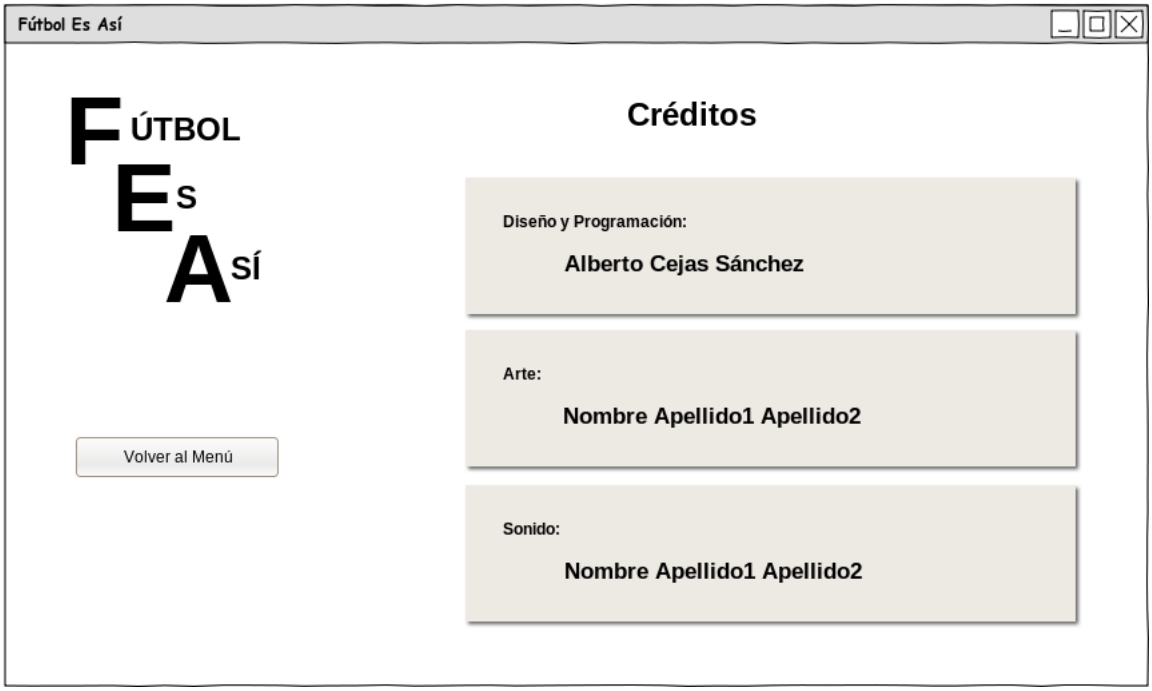


Figura 4.2: Análisis: Boceto de la pantalla de créditos

Ventana de juego Ventana principal de todo el juego (figura 4.3). Mostrará el partido actual que se esté

dispuntando, así como el marcador y el cronómetro. Se podrá acceder al menú de pausa pusando el botón pause del joystick.

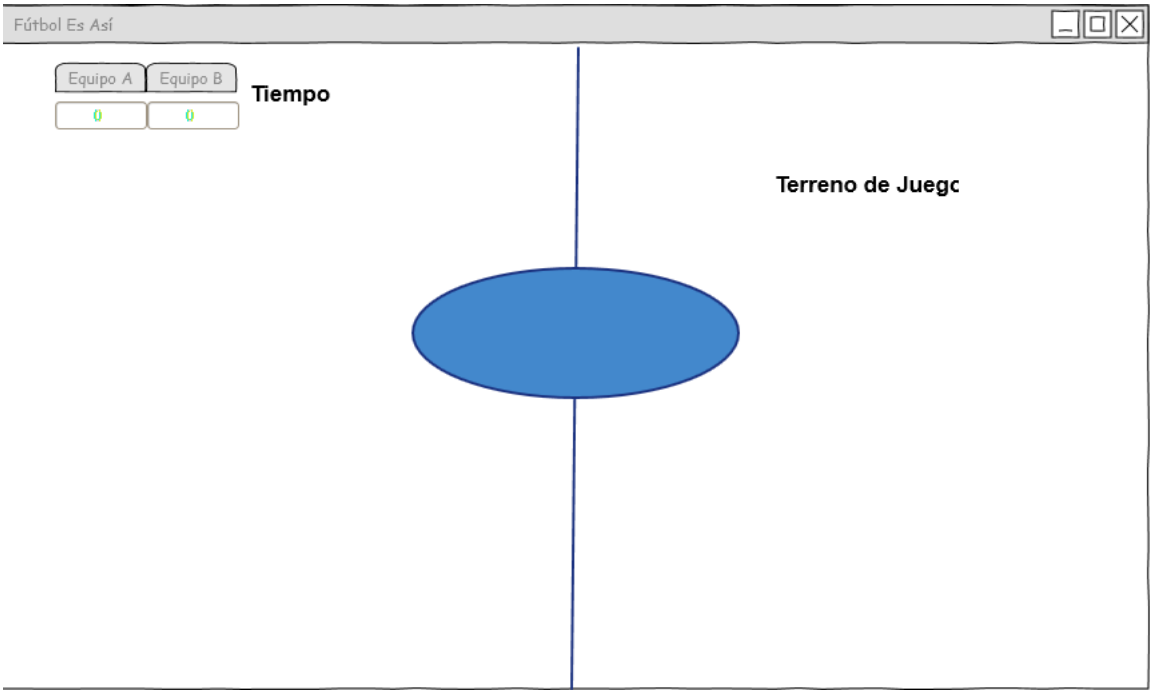


Figura 4.3: Análisis: Boceto de ventana de juego

Ventana de pausa Únicamente accesible desde la ventana de juego (figura 4.4). Esta nos permitirá detener el juego en curso, siendo posible reanudar el juego o volver al menú principal.

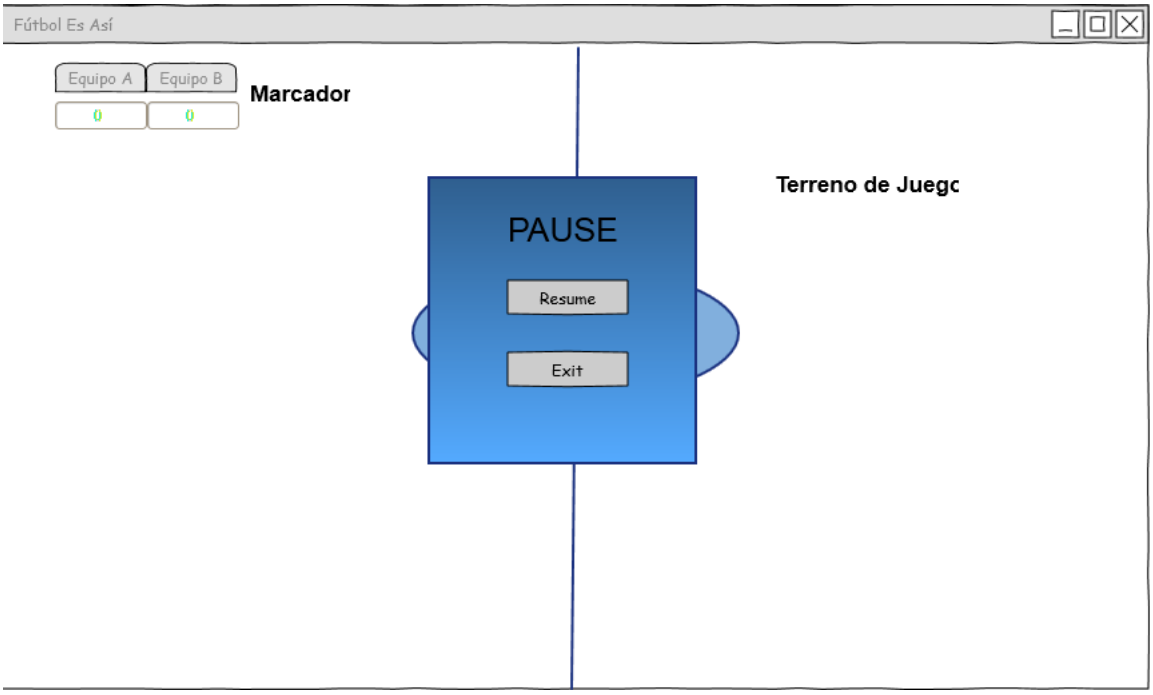


Figura 4.4: Análisis: Boceto de menú de pausa

4.1.2. Requisitos funcionales

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

- Salir de la aplicación desde cualquier ventana.
- Ver créditos.
- Jugar.
- Pausar el juego.
- Pasar balón.
- Chutar balón.
- Mover Jugador.

Los distintos tipos de jugadores son:

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

Existe un único modo de juego:

- **Partido rápido**: consiste en la realización de un partido de 5 minutos, disputado entre el ordenador y la máquina.

4.1.3. Requisitos de rendimiento

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

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

4.1.4. Restricciones de diseño

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

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

4.1.5. Requisitos del sistema software

La aplicación deberá cumplir los siguiente requisitos del sistema:

- Deberá ser multiplataforma, al menos en los siguiente sistemas:
  - **Microsoft Windows**.
  - **GNU/Linux**: usando la distribución Ubuntu 12.10 como principal sistema para pruebas.
- El código con el que se desarrolle la aplicación no debe ser dependiente del sistema en el que se desarrolle.
- El código debe ser mantenible y fácilmente ampliable para futuras versiones.

4.2. Modelo de casos de uso

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

4.2.1. Diagrama de los casos de uso

En primer lugar mostramos el modelo de casos de uso (figura 4.5), que representa la funcionalidad completa de la aplicación. Se ha usado el siguiente esquema:

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

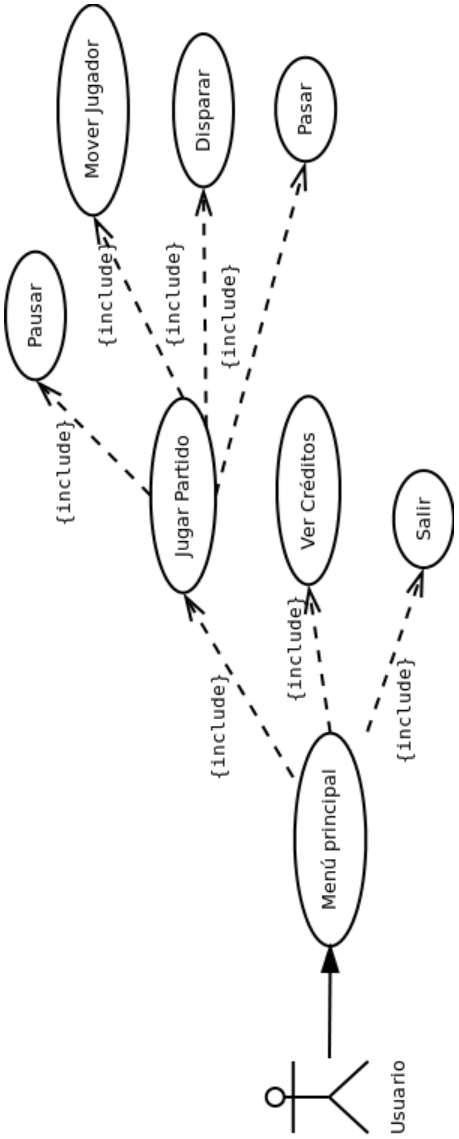


Figura 4.5: Análisis: Diagrama de casos de uso

4.2.2. Descripción de los casos de uso

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

Caso de uso: Menú principal

Caso de uso Menú principal

Descripción Se muestra el menú principal de la aplicación, donde es posible elegir entre jugar, ver los créditos o salir.

Actores Usuario

Precondiciones Ninguna

Postcondiciones Ninguna

Escenario principal

1. El usuario inicia la aplicación
2. El sistema muestra el menú principal del juego en pantalla.
3. El usuario selecciona la opción **jugar**.
4. El sistema inicia el juego.

Extensiones — flujo alternativo

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

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

1. El sistema inicia la opción créditos

2c El usuario selección la opción **salir**.

1. El sistema sale de la aplicación.

Caso de uso: Jugar partido

Caso de uso Jugar partido

Descripción El usuario juega un partido

Actores Usuario

Precondiciones Ninguna

Postcondiciones El usuario completa un partido.

Escenario principal

1. El sistema carga el escenario, los jugadores, el balón y las porterías.
2. El sistema muestra la pantalla de juego.
3. El usuario y el sistema interactúan durante el partido.
4. El usuario completa el partido.

5. El sistema muestra un mensaje fin de partido y vuelve al menú principal.

Extensiones — flujo alternativo

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

Caso de uso: Pausar

Caso de uso Pausar

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

Actores Usuario

Precondiciones Se está jugando una carrera

Postcondiciones Ninguna.

Escenario principal

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

Extensiones — flujo alternativo

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

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

1. El sistema vuelve al menú principal.

Caso de uso: Mover jugador

Caso de uso Mover jugador

Descripción El usuario desplaza al futbolista seleccionado por el terreno de juego.

Actores Usuario

Precondiciones Se está jugando un partido y hay un futbolista seleccionado.

Postcondiciones Ninguna.

Escenario principal

1. El usuario pulsa sobre una de los botones de movimiento de la cruceta del joystick.
2. El sistema mueve al futbolista en la dirección seleccionada.
3. El sistema comprueba que no ha colisionado con otro cuerpo.

Extensiones — flujo alternativo

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

1a El usuario pulsa el botón R1 al mismo tiempo que la dirección.

1. El futbolista se mueve a mayor velocidad en la dirección indicada.

3a El sistema detecta que ha colisionado con un oponente, compañero o portería.

1. El sistema corrige la posición del futbolista.

Caso de uso: Pasar Balón

Caso de uso Pasar Balón

Descripción El usuario ordena al futbolista seleccionado a ejecutar un pase en su orientación actual.

Actores Usuario

Precondiciones Se está jugando un partido, hay un futbolista seleccionado y éste posee el balón.

Postcondiciones Ninguna.

Escenario principal

1. El usuario pulsa el botón X del joystick.
2. El sistema ejerce una fuerza sobre el balón en la dirección del miembro del mismo equipo que recibe, siendo éste el más próximo a la orientación actual del pasador.
3. El receptor controla el balón.

Extensiones — flujo alternativo

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

2a El sistema detecta que no hay ningún miembro del equipo cercano a la orientación actual del pasador.

1. El sistema ejerce una fuerza sobre el balón en la dirección a la que actualmente está mirando el pasador.

3a El balón es interceptado por un miembro del equipo contrario.

1. El oponente controla el balón.

3b El balón es interceptado por otro miembro del mismo equipo.

1. El otro jugador controla el balón.

Caso de uso: Chutar Balón

Caso de uso Chutar Balón

Descripción El usuario desplaza al futbolista seleccionado por el terreno de juego.

Actores Usuario

Precondiciones Se está jugando un partido, hay un futbolista seleccionado y éste posee el balón.

Postcondiciones Ninguna.

Escenario principal

1. El usuario pulsa el botón cuadrado del joystick durante un tiempo concreto.
2. El sistema ejerce una fuerza sobre el balón, acorde al tiempo que se ha mantenido pulsado el botón cuadrado, en la dirección en la que está mirando el lanzador.
3. El portero contrario intercepta el disparo y controla el esférico.

Extensiones — flujo alternativo

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

3a El balón es interceptado por un jugador de campo y su potencia o altura impiden el control por parte del futbolista.

1. El sistema hace rebotar al balón en la orientación adecuada.

3b El balón es interceptado por un jugador de campo y su potencia o altura permiten el control por parte del futbolista.

1. El futbolista controla el balón.

3c El balón entra en la portería.

1. El marcador del equipo del lanzador se incrementa en una unidad y el equipo contrario saca de centro.

3d El balón sale por la línea de fondo.

1. El portero oponente saca de puerta.

3d El balón sale por la línea lateral.

1. El equipo contrario saca de banda en la posición por la que salió el balón.

Caso de uso: Ver Créditos

Caso de uso Ver Créditos

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

Actores Usuario

Precondiciones Ninguna.

Postcondiciones Ninguna.

Escenario principal

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

Extensiones — flujo alternativo

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

Caso de uso: Salir

Caso de uso Salir

Descripción El usuario desea cerrar la aplicación.

Actores Usuario

Precondiciones Ninguna

Postcondiciones Se sale de la aplicación.

Escenario principal

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

Extensiones — flujo alternativo

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

4.3. Modelo conceptual de datos

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

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

4.3.1. Diagrama de clases conceptuales

En la siguiente imagen podemos ver el diagrama de clases asociado a los requisitos obtenidos. Se muestran las clases relacionadas con la pantalla de juego y su relación con el menú principal y créditos así como las clases principales que intervienen en la gestión de la pantalla de juego.

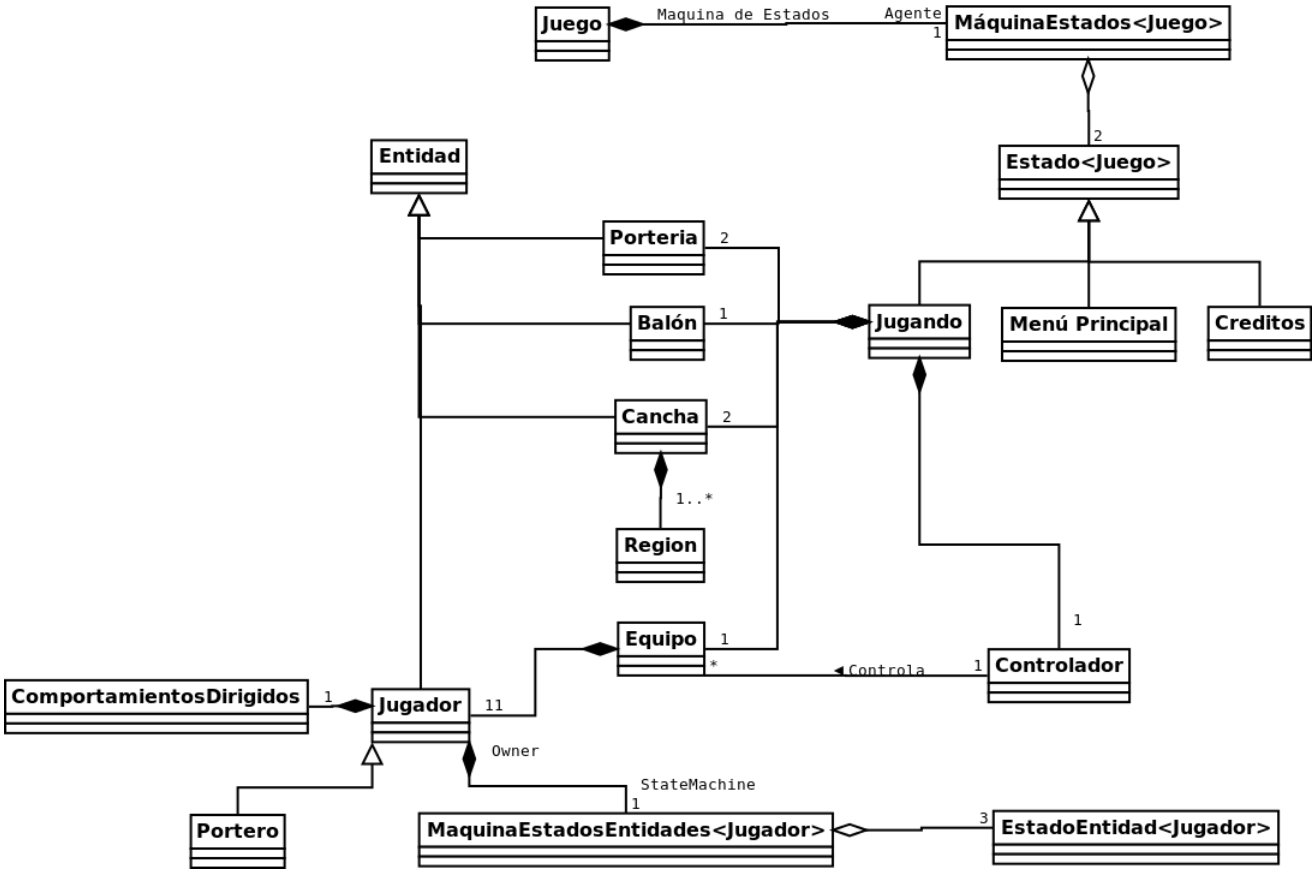


Figura 4.6: Análisis: Diagrama de clases conceptuales

4.4. Modelo de comportamiento del sistema

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

- Diagramas de secuencias del sistema: muestran la secuencia de eventos entre el usuario y el sistema.

- Contrato de las operaciones del sistema: describen el efecto que producen las operaciones en el sistema.

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

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

Caso de uso: Menú principal(escenario principal)

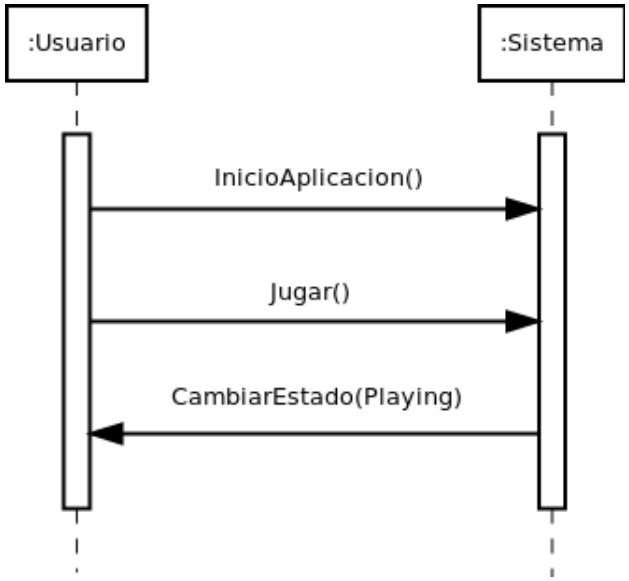


Figura 4.7: Análisis: Diagrama de secuencia Menú principal (escenario principal)

Operación InicioAplicacion()

Actores Jugador, sistema.

Responsabilidades inicia la aplicación y muestra el menú principal.

Precondiciones Ninguna

Postcondiciones

- El sistema inicia todos los subsistemas necesarios para la correcta ejecución de la aplicación.
- El sistema cambia el estado de autómata juego a 'MainMenu'.

Operación Jugar()

Actores Jugador, Sistema.

Responsabilidades salir del menú principal y acceder a la pantalla de juego.

Precondiciones

- El estado del juego es 'MainMenu'.

Postcondiciones

- Se cambia de estado el juego a 'Playing'.

Caso de uso: Menú principal(escenario 2a)

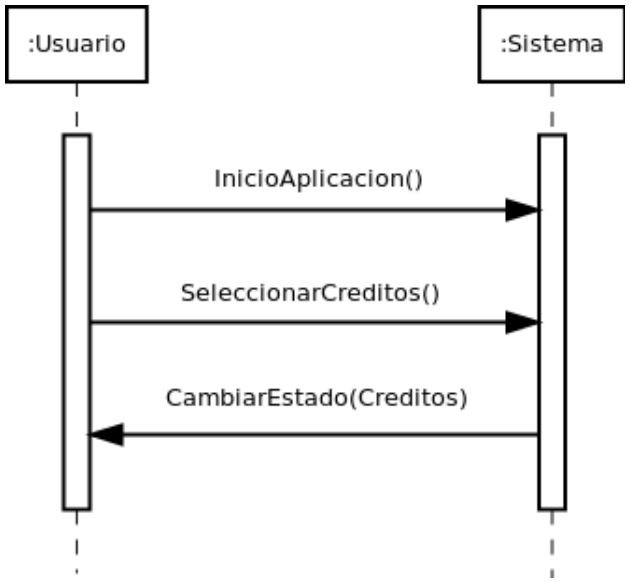


Figura 4.8: Análisis: Diagrama de secuencia Menú principal (escenario 2a)

Operación SeleccionarCréditos()

Actores Jugador, sistema.

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

Precondiciones

- El autómata juego está en estado 'MainMenu'.

Postcondiciones

- El autómata juego cambia al estado 'Credits'.

Caso de uso: Menú principal(escenario 2b)

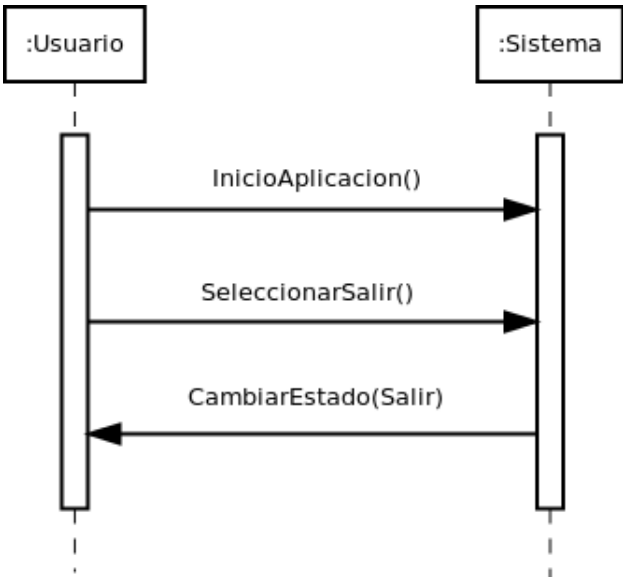


Figura 4.9: Análisis: Diagrama de secuencia Menú principal (escenario 2b)

Operación SeleccionarSalir()

Actores Jugador, sistema.

Responsabilidades Salir de menú principal y salir de la aplicación

Precondiciones

- El autómata juego está en estado 'MainMenu'.

Postcondiciones

- Se elimina el objeto 'Game'.

Caso de uso: Jugar partido (escenario principal)

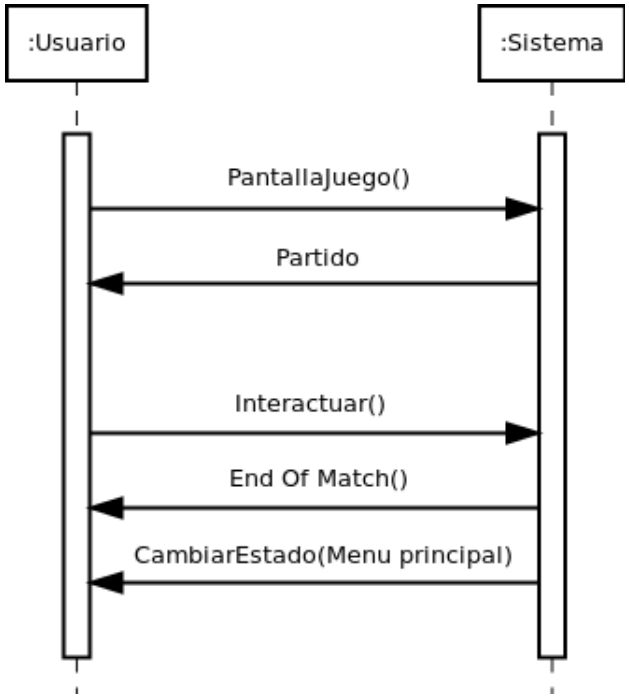


Figura 4.10: Análisis: Diagrama de secuencia Jugar partido (escenario principal)

Operación PantallaJuego()

Actores Jugador, sistema.

Responsabilidades carga, muestra e inicia la pantalla de juego.

Precondiciones Ninguna

Postcondiciones

- Se crea un objeto de Playing.
- Se crea un objeto de SoccerPitch.
- Se crea un objeto de Player por cada futblista especializando en los porteros en GoalKeeper.
- Se crea un objeto de Ball.
- Se crean dos objetos de Goal.

Operación Interactuar()

Actores Jugador, sistema.

Responsabilidades permite al jugador interactuar con el mundo 3D y los elementos que este posee.

Precondiciones

- El juego está en estado Playing.
- Existen objetos Player, estando uno de ellos seleccionado por el Jugador.

Postcondiciones Ninguna.

Caso de uso: Pausar (escenario principal)

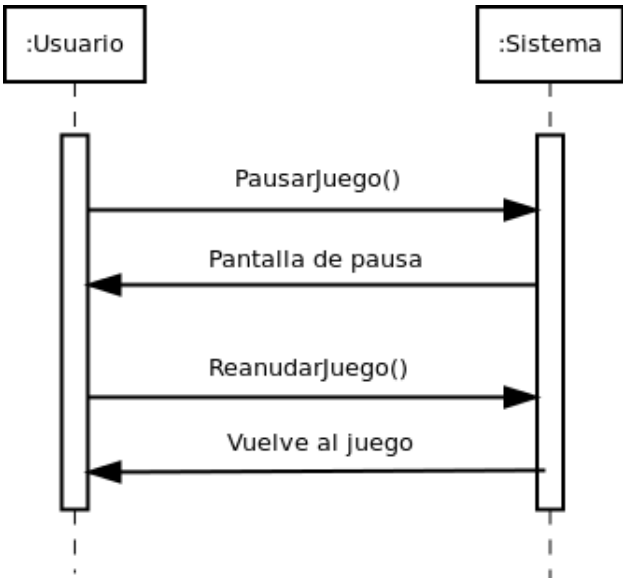


Figura 4.11: Análisis: Diagrama de secuencia Pausar(escenario principal)

Operación PausarJuego()

Actores Jugador, sistema.

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

Precondiciones

- El objeto juego está en estado Playing.

Postcondiciones

- Se activa y se hace visible el panel de pausa.

Operación ReanudarJuego()

Actores Jugador, sistema.

Responsabilidades reanuda la partida y quita el menú de pausa.

Precondiciones

- El objeto juego está en estado Playing.
- El partido estaba pausado.

Postcondiciones

- Se desactiva el panel de pausa y se hace invisible.

Caso de uso: Pausar (escenario *b)

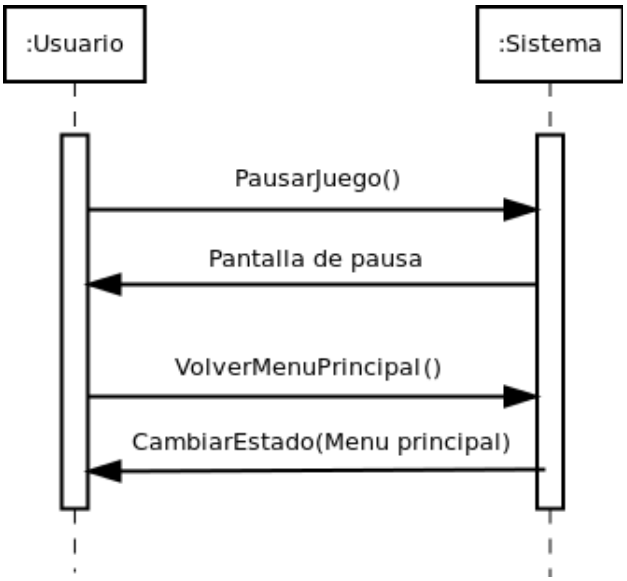


Figura 4.12: Análisis: Diagrama de secuencia (escenario b)

Operación VolverMenuPrincipal()

Actores Jugador, sistema.

Responsabilidades se el partido actual y se vuelve al menú principal.

Precondiciones

- El objeto juego está en estado Playing.
- El panel de pause está activado y visible.

Postcondiciones

- Se libera la memoria ocupada por los elementos del partido y se desactiva el menú de pausa.

Caso de uso: Mover futbolista (escenario principal)

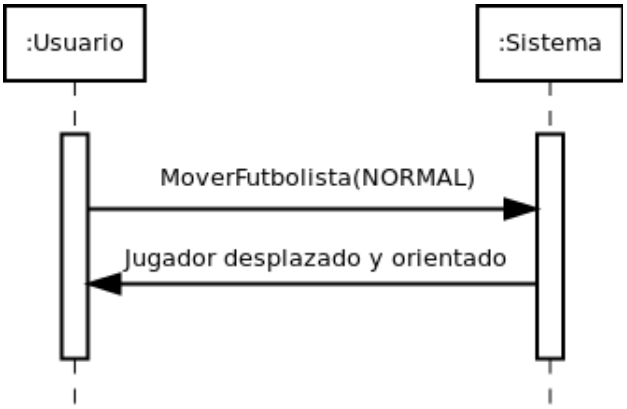


Figura 4.13: Análisis: Diagrama de secuencia Mover futbolista (escenario principal)

Operación MoverFutbolista()

Actores Jugador, sistema.

Responsabilidades mueve al personaje por el mundo 3D.

Precondiciones

- El objeto juego está en estado Playing.
- Existe un objeto de InputManager.
- Existe un objeto de SoccerTeam s.
- Existe un objeto de Player seleccionado perteneciente al equipo s.

Postcondiciones

- Se modifica la posición del objeto Player.

Caso de uso: Mover futbolista (escenario 1a)

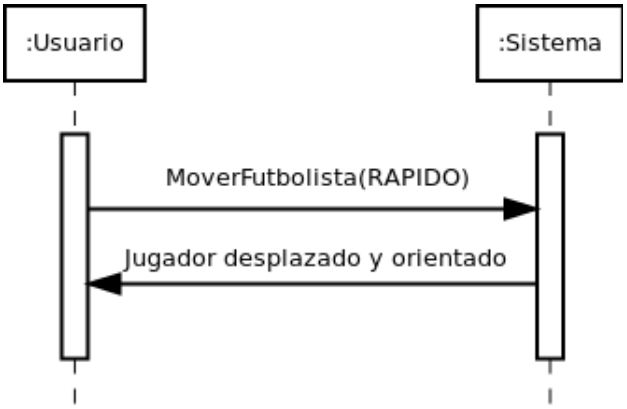


Figura 4.14: Análisis: Diagrama de secuencia Mover futbolista (escenario 1a)

Operación MoverFutbolista()

Actores Jugador, sistema.

Responsabilidades mueve al personaje por el mundo 3D.

Precondiciones

- El objeto juego está en estado Playing.
- Existe un objeto de InputManager.
- Existe un objeto de SoccerTeam s.
- Existe un objeto de Player seleccionado perteneciente al equipo s.
- El usuario tiene pulsado el botón R1.

Postcondiciones

- Se modifica la posición del objeto Player más rápido.

Caso de uso: Pasar Balón (escenario principal)

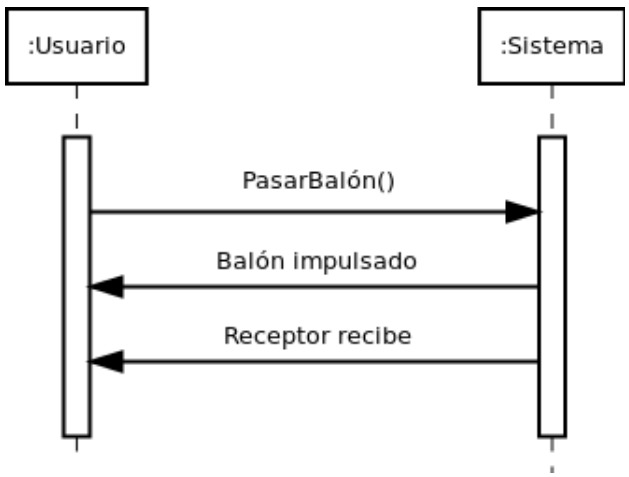


Figura 4.15: Análisis: Diagrama de secuencia Pasar Balón (escenario principal)

Operación PasarBalon()

Actores Jugador, sistema.

Responsabilidades hace que el jugador pase el balón hacia el receptor más cercano en su orientación actual.

Precondiciones

- Existe un objeto p1 de Player perteneciente al SoccerTeam s que posea el balón.
- Existe un objeto p2 de Player perteneciente al SoccerTeam s que no posea el balón.
- El objeto juego está en estado Playing.

Postcondiciones

- El objeto player p1 deja de poseer el balón.
- El objeto player p2 es el nuevo poseedor del balón.

Caso de uso: Pasar Balón (escenario 2a)

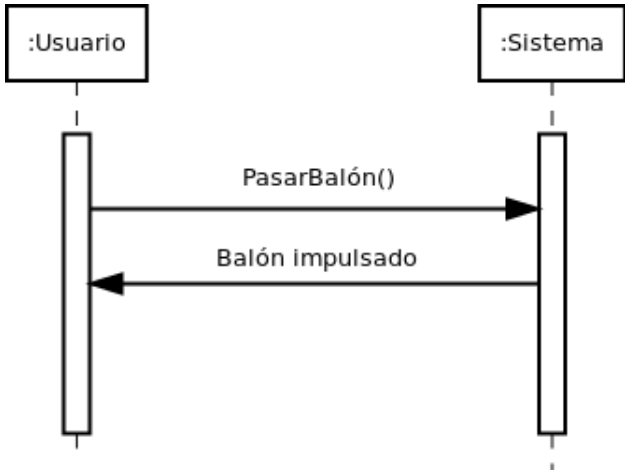


Figura 4.16: Análisis: Diagrama de secuencia Pasar Balón (escenario 2a)

Operación PasarBalon()

Actores Jugador, sistema.

Responsabilidades hace que el jugador pase el balón su orientación actual.

Precondiciones

- Existe un objeto p1 de Player perteneciente al SoccerTeam s que posea el balón.
- El objeto juego está en estado Playing.

Postcondiciones

- El objeto player p1 deja de poseer el balón.
- El balón es impulsado en la dirección en la que está mirando el jugador p1.

Caso de uso: Pasar Balón (escenario 3a)

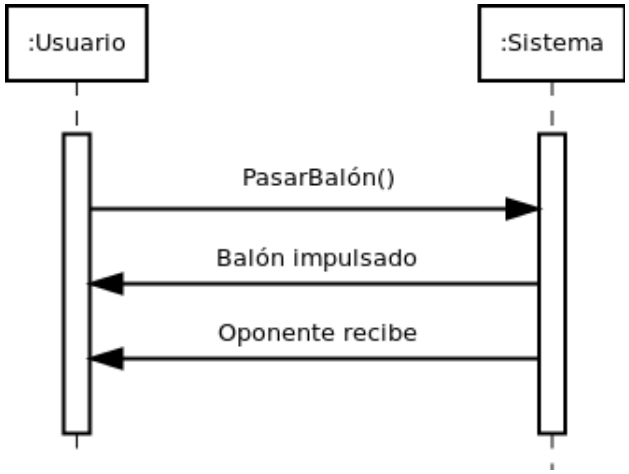


Figura 4.17: Análisis: Diagrama de secuencia Pasar Balón (escenario 3a)

Operación PasarBalon()

Actores Jugador, sistema.

Responsabilidades hace que el jugador pase el balón.

Precondiciones

- Existe un objeto p1 de Player perteneciente al SoccerTeam s1 que posea el balón.
- Existe un objeto p2 de Player perteneciente al SoccerTeam s2 que no posee el balón.
- El objeto juego está en estado Playing.

Postcondiciones

- El objeto player p1 deja de poseer el balón.
- El objeto player p2 intercepta el balón.

Caso de uso: Chutar Balón (escenario principal)

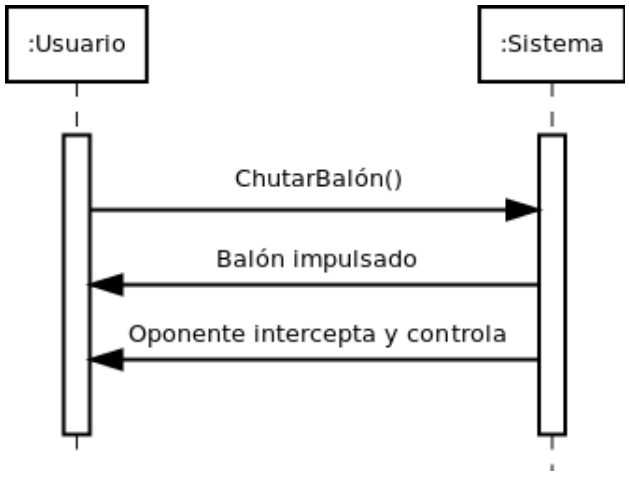


Figura 4.18: Análisis: Diagrama de secuencia Chutar Balón (escenario principal)

Operación ChutarBalon()

Actores Jugador, sistema.

Responsabilidades hace que el jugador chute el balón hacia su orientación actual.

Precondiciones

- Existe un objeto p1 de Player perteneciente al SoccerTeam s1 que posea el balón.
- Existe un objeto p2 de Player perteneciente al SoccerTeam s2.
- El objeto juego está en estado Playing.

Postcondiciones

- El objeto player p1 deja de poseer el balón.
- El objeto player p2 intercepta y controla el disparo.

Caso de uso: Chutar Balón (escenario 3a)

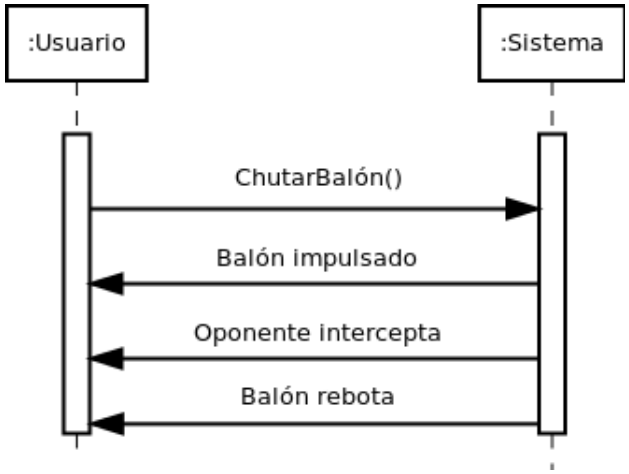


Figura 4.19: Análisis: Diagrama de secuencia Chutar Balón (escenario 3a)

Operación ChutarBalon()

Actores Jugador, sistema.

Responsabilidades hace que el jugador chute el balón hacia su orientación actual.

Precondiciones

- Existe un objeto p1 de Player perteneciente al SoccerTeam s1 que posea el balón.
- Existe un objeto p2 de Player perteneciente al SoccerTeam s2.
- El objeto juego está en estado Playing.

Postcondiciones

- El objeto player p1 deja de poseer el balón.
- El objeto player p2 intercepta el disparo pero no controla el balón y éste sale rebotado.

Caso de uso: Chutar Balón (escenario 3c)

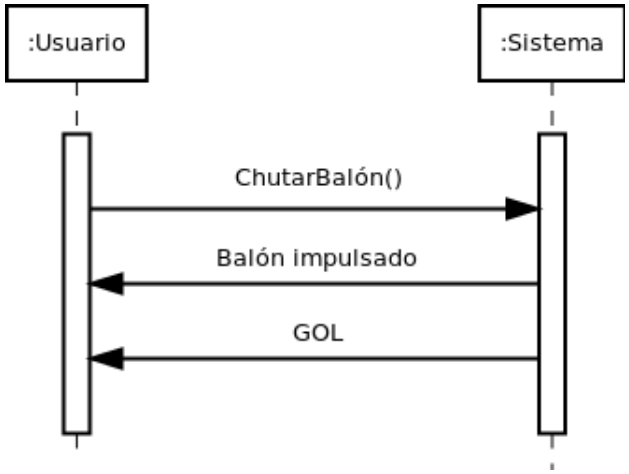


Figura 4.20: Análisis: Diagrama de secuencia Chutar Balón (escenario 3c)

Operación ChutarBalon()

Actores Jugador, sistema.

Responsabilidades hace que el jugador chute el balón hacia su orientación actual.

Precondiciones

- Existe un objeto p1 de Player perteneciente al SoccerTeam s1 que posea el balón.
- El objeto juego está en estado Playing.

Postcondiciones

- El objeto player p1 deja de poseer el balón.
- El marcador del SoccerTeam s1 se incrementa en una unidad.

Caso de uso: Chutar Balón (escenario 3d)

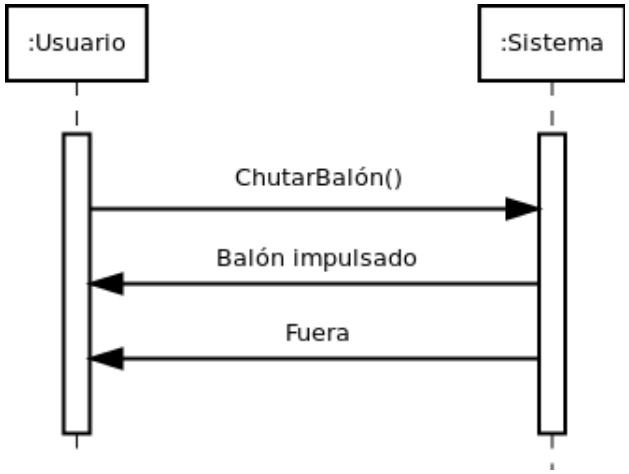


Figura 4.21: Análisis: Diagrama de secuencia Chutar Balón (escenario 3d)

Operación ChutarBalon()

Actores Jugador, sistema.

Responsabilidades hace que el jugador chute el balón hacia su orientación actual.

Precondiciones

- Existe un objeto p1 de Player perteneciente al SoccerTeam s1 que posea el balón.
- El objeto juego está en estado Playing.

Postcondiciones

- El objeto player p1 deja de poseer el balón.
- El balón es impulsado y sale inmediatamente fuera.

Caso de uso: Créditos (escenario principal)

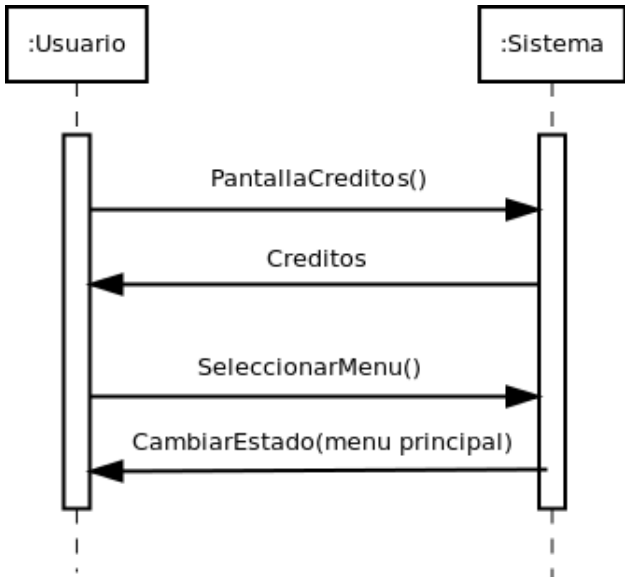


Figura 4.22: Análisis: Diagrama de secuencia Créditos(escenario principal)

Operación PantallaCréditos()

Actores Jugador, sistema.

Responsabilidades crea y muestra la pantalla de créditos

Precondiciones

- El objeto juego está en estado MainMenu.

Postcondiciones

- El objeto juego cambia a estado Credits.

Operación SeleccionarMenu()

Actores Jugador, sistema.

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

Precondiciones

- El objeto juego está en estado Credits.

Postcondiciones

- Se destruye el estado Credits y se pasa a estado MainMenu.

Capítulo 5

Diseño

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

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

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

5.1. Interfaz gráfica

Tras los resultado que se han obtenido en la fase de análisis del sistema, es necesario desarrollar una interfaz sencilla y agradable para el usuario de la misma. En el diseño de estos, se intentará en todo momento que el flujo de ejecución sea lo más intuitivo posible para el usuario.

Figura 5.1: Diseño: Captura de la interfaz del sistema

5.1.1. Diagrama de interacción entre interfaces

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

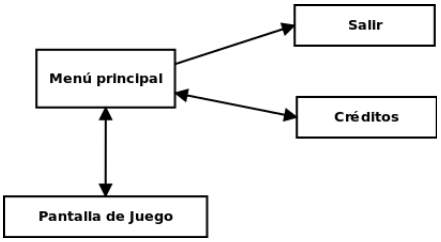


Figura 5.2: Diseño: Diagrama de interacción

5.2. Diagrama de clases de diseño

Dividiremos el diagrama de clases de diseño en dos partes, para verlo con una mayor claridad, dicha separación se basará en el cometido de las clases que interfieran.
En primer lugar mostraremos las clases encargadas de la navegación entre los menús y la pantalla de juego, que corresponden a los distintos estados que puede adoptar el juego.

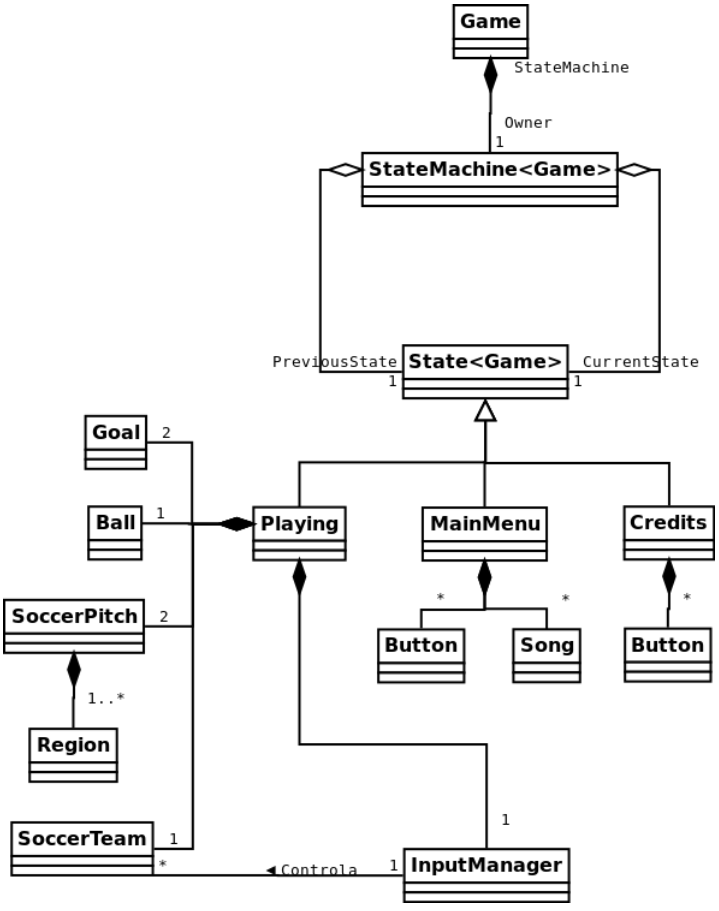
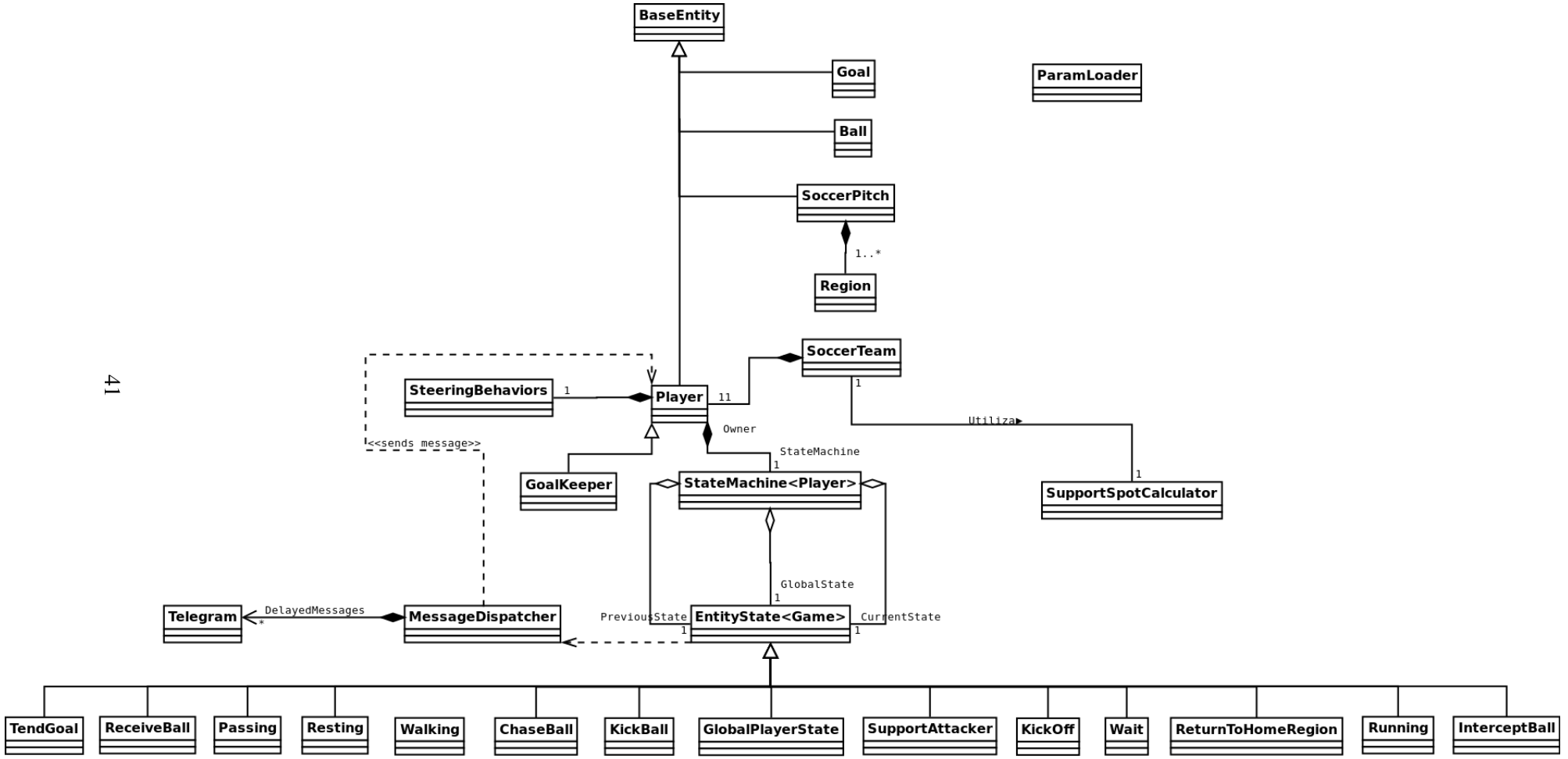


Figura 5.3: Diseño: Diagrama de clases de diseño 1

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


BaseEntity
Goal
ParamLoader
Ball
SoccerPitch
1..*
Region
SoccerTeam
SteeringBehaviors
1
Player
11
1
Utiliza
<<sends message>>
Owner
StateMachine
1
GoalKeeper
StateMachine<Player>
SupportSpotCalculator
1
GlobalState
1
Telegram
DelayedMessages
*
MessageDispatcher
PreviousState
1
EntityState<Game>
CurrentState
1
TendGoal
ReceiveBall
Passing
Resting
Walking
ChaseBall
KickBall
GlobalPlayerState
SupportAttacker
KickOff
Wait
ReturnToHomeRegion
Running
InterceptBall

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

A continuación, se muestra una lista con las diferentes clases necesarias para la realización de sistema. Junto a cada una de las clases habrá una pequeña descripción sobre la labor que desempeña cada una.

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

StateMachine Clase con las necesidades básicas de una máquina de estados para el juego en sí.

EntityStateMachine Clase con las necesidades básicas de una máquina de estados para los jugadores.

State Clase con las necesidades básicas de los estados del juego o los personajes.

Playing Clase encargada de gestionar el juego del partido.

MainMenu Clase que gestiona el menú principal.

Credits Clase que gestiona el menú de selección de personaje.

Base Entity Clase virtual que representa las entidades del juego.

Ball Clase que representa el balón.

Goal Clase que representa la portería.

Player Clase que gestiona al futbolista.

GoalKeeper Clase especializada de Player para gestionar lo exclusivo de los porteros.

InputManager Clase encargada de manejar la entrada del usuario.

ParamLoader Clase encargada de cargar parámetros predefinidos en un documento de texto.

SoccerTeam Clase que gestiona un equipo.

MessageDispatcher Clase encargada de enviar mensajes entre entidades del juego.

Region Clase que representa cada una de las zonas en las que está subdivido el terreno de juego.

SteeringBehaviors Clase encargada de gestionar los comportamientos de los futbolistas.

Song Clase para representar las canciones que suenen en el menú principal.

Telegram Clase que representa el contenido de un mensaje entre entidades.

Attacking Clase que gestiona el comportamiento de un equipo de fútbol cuando está atacando.

Deffending Clase que gestiona el comportamiento de un equipo de fútbol cuando está defendiendo.

PrepareForKickOff Clase que gestiona el comportamiento de un equipo de fútbol cuando hay un saque de banda, corner, centro o portería.

SupportSpot Clase que que representa los puntos de apoyo imaginarios dibujados sobre el terreno de juego.

SoccerPitch Clase que que representa el terreno de juego.

GlobalPlayerState Clase que que representa el estado global de un futbolista.

Resting Clase que que representa el estado de no hacer nada en un futbolista.

Walking Clase que que representa el estado de andar en un futbolista.

ChaseBall Clase que que representa el estado de perseguir el balón en un futbolista..

KickBall Clase que que representa el estado de intentar disparar a puerta en un futbolista..

Dribble Clase que que representa el estado de regatear en un futbolista.

SupportAttacker Clase que que representa el estado de apoyar/desmarcarse en un futbolista..

KickOff Clase que que representa el estado de sacar de banda/corner/centro/portería en un futbolista.

Wait Clase que que representa el estado de esperar en un futbolista.

ReturnToHomeRegion Clase que que representa el estado de volver a la posición que ocupa en el campo en un futbolista.

ReceiveBall Clase que que representa el estado de recibir un pase en un futbolista.

TendGoal Clase que que representa el estado de cubrir la portería en un portero.

InterceptBall Clase que que representa el estado de interceptar un balón en un portero.

Running Clase que que representa el estado de correr en un futbolista.

Passing Clase que que representa el estado de pasar en un futbolista.

Capítulo 6

Implementación

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

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

También esta disponible toda la documentación del proyecto generada con doxygen. Aquí un enlace al directorio de dicha documentación:

https://forja.rediris.es/plugins/scmsvn/viewcvs.php/trunk/doc/html/?root=cusl5-fea

6.1. Físicas y Colisiones

La detección de las colisiones es una de las cosas más básicas de la mayoría de los juegos y más aún en los deportivos ya que la pelota cobra especial protagonismo al interactuar con todos los jugadores, el campo y las porterías. Además, hay que tratar las colisiones entre futbolistas y de éstos con los postes de la portería.

Debido a la complejidad del espacio 3D en el que hay que tener en cuenta factores más complejos como el bote del balón o la dirección en la que sale despedido éste cuando choca con la portería, se optó por *Bullet+OgreBullet* de manera que todo esto se manejase automáticamente. Sin embargo, esta liberación de la responsabilidad de tratar estas colisiones introduce una dificultad adicional: la integración de la librería con el sistema y la interactuación con las imprevisibles actividades realizadas por el usuario.
El principal problema encontrado es la necesidad de conocer cúanto tiempo tardará el balón en alcanzar una posición si se le aplica una fuerza. Esta funcionalidad no es soportada por ninguna librería de físicas libre por lo que al resultar imposible aproximarlo por fórmulas matemáticas, se tuvo que crear un mundo físico paralelo y realizar la simulación en las mismas circunstacias, adelantando así el tiempo que tardará. Esta operación se convierte en la más costosa del sistema.

6.2. Inteligencia artificial

Otro de los aspectos más importantes y complejos de un videojuego de las características de *Fútbol Es Así*, es la inteligencia artificial, ya que uno de los dos equipos será dirigido por el ordenador. Para poder

simular el comportamiento de los futbolistas o de un conjunto de ellos se han implementado un conjunto de estados y comportamientos.

6.2.1. Jugadores

Los comportamientos más básicos de los futbolistas son:

- **Perseguir**. Mueve al cuerpo hacia la posición destino.

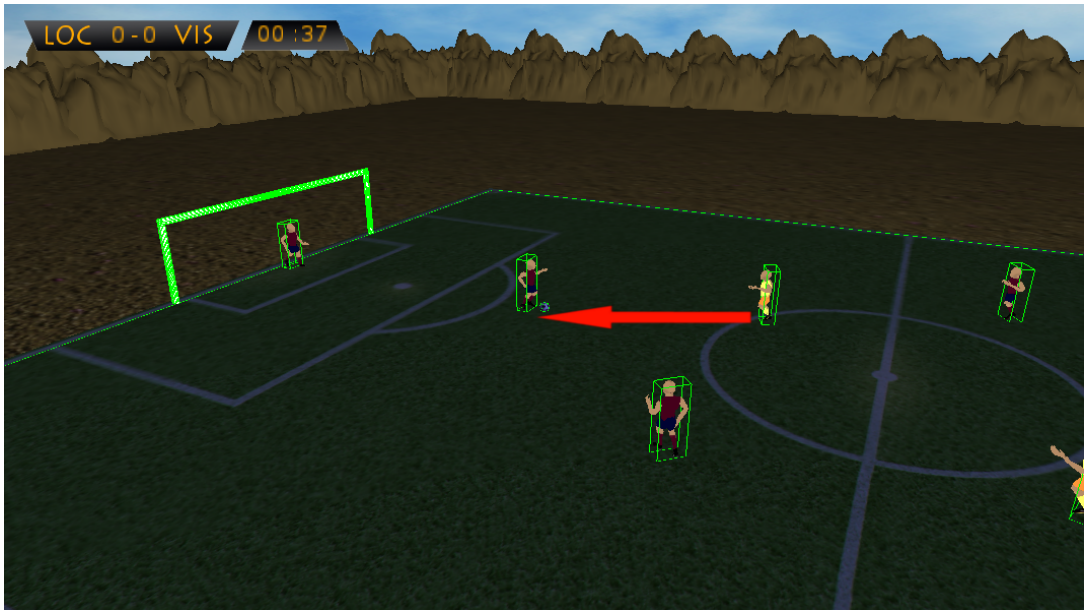


Figura 6.1: Implementación: Seek

  Este comportamiento se usa especialmente en el estado ÇhaseBallza sea porque el oponente posee el balón o porque este último está libre.

- **Predecir**. Mueve al cuerpo en la dirección en la que el balón se encontrará en el futuro según la velocidad de éste y la distancia entre ellos.

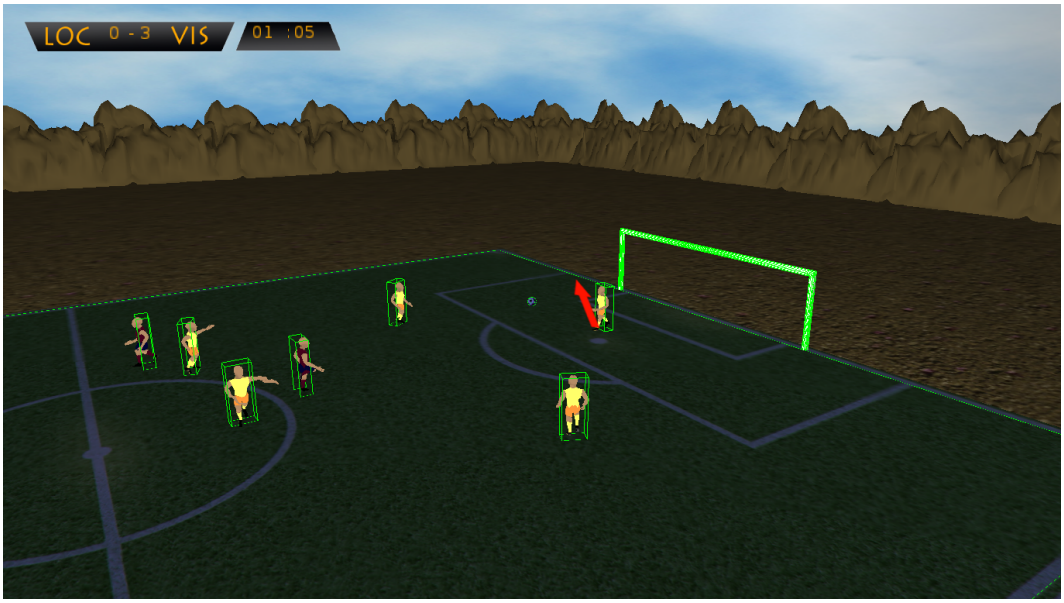


Figura 6.2: Implementación: Pursuit

- **Interponerse**. Mueve al cuerpo a la posición situada entre éste y otro cuerpo.

Figura 6.3: Implementación: Interpose

La IA también implementa la decisión de los futbolistas a la hora de realizar un pase o un disparo, escogiendo el destino y la potencia según la distancia. El flujo de selección del futbolista cuando tiene la posesión del balón es el siguiente:

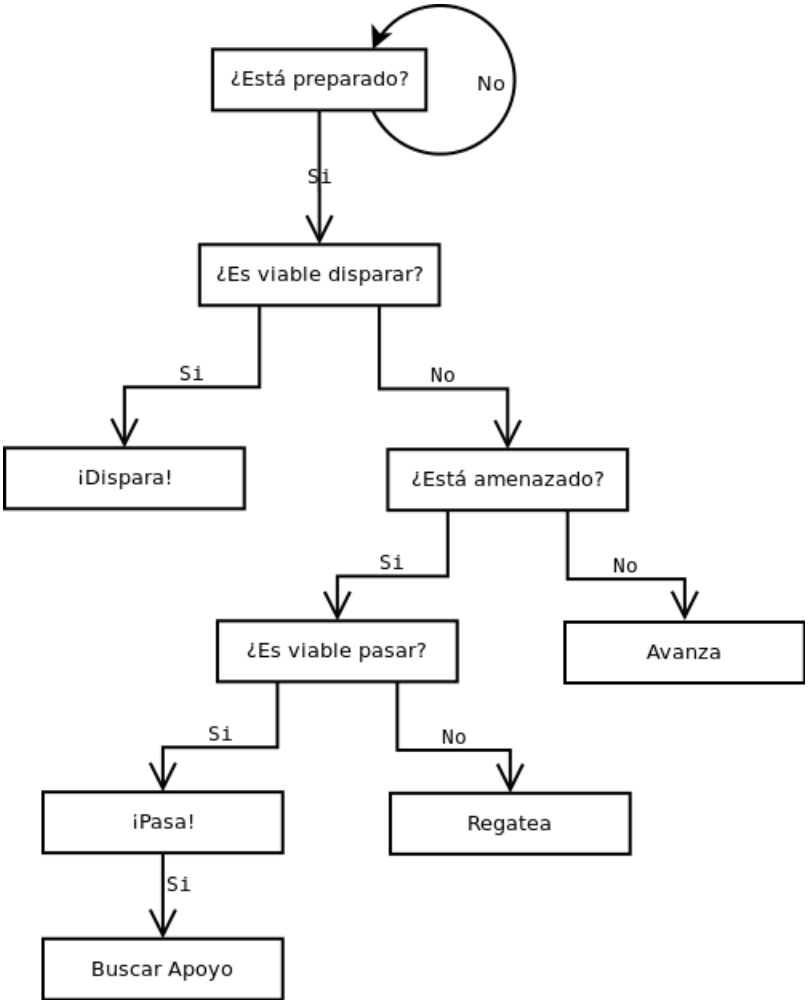


Figura 6.4: Implementación: Flujo posesión

Como se puede apreciar en la imagen anterior, el flujo de ejecución es seleccionado según unas condiciones:

- **¿Está Preparado?.** Indica si el futbolista está preparado para realizar alguna acción ya que siempre que controla el esférico debe dejar pasar un pequeño tiempo para que los movimientos no sean inmediatos.

- **¿Es Viable Disparar?.** Indica si el futbolista tiene una opción certera de disparar, para ello genera tres posibles disparos, estableciendo como objetivos tres posiciones aleatorias de la portería. Es viable si:

  - El balón tiene fuerza suficiente para llegar a la línea de gol.

  - El balón no puede ser interceptado por los futbolistas rivales. Concretamente ningún jugador rival(incluyendo el portero) puede llegar, intersectando perpendicularmente a la línea recta que une al receptor y al pasador, antes que el balón. Tal que:

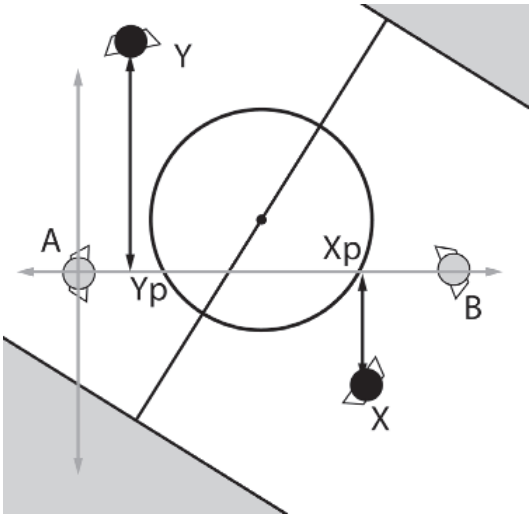


Figura 6.5: Implementación: Disparo viable, fuente: [1]

- **¿Está Amenazado?.** Si hay un oponente lo suficientemente cerca del poseedor del balón, dentro de un radio específico desde la posición actual de éste.

- **¿Es Viable Pasar?.** Indica si el futbolista tiene una opción certera de pasar, para ello comprueba tres posibles pases para cada receptor. El primero hacia la posición actual exacta del receptor, el segundo y tercero tienen como destino las dos tangentes a un radio específico desde la posición actual exacta del jugador. Tal que:

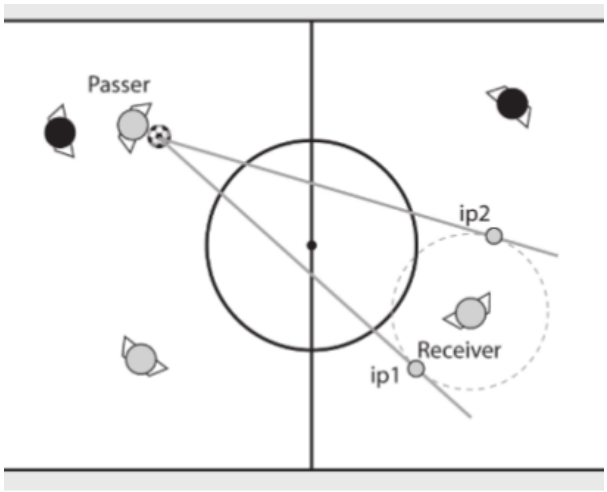


Figura 6.6: Implementación: Pase viable, fuente: [1]

Estos tres pases son viables si:

- El balón tiene fuerza suficiente para llegar al destino.
- El balón no puede ser interceptado por los futbolistas rivales. Concretamente ningún jugador rival puede llegar, intersectando perpendicularmente a la línea recta que une al receptor y al pasador, antes que el balón.

De todos los posibles pases considerados como viables, se elige el que tenga como destino una posición más cercana a la portería contraria.

Además de los saltos condicionales, hay dos operaciones que describen dos comportamientos de los futbolistas:

- **Buscar Apoyo.** Determina el jugador que está más cerca del punto de apoyo que más puntuación tiene en ese momento, y se le mensajea para que cambie su estado a SupportAttacker. Los puntos de apoyo son posiciones preestablecidas que se encuentran dentro del terreno de juego tal que:

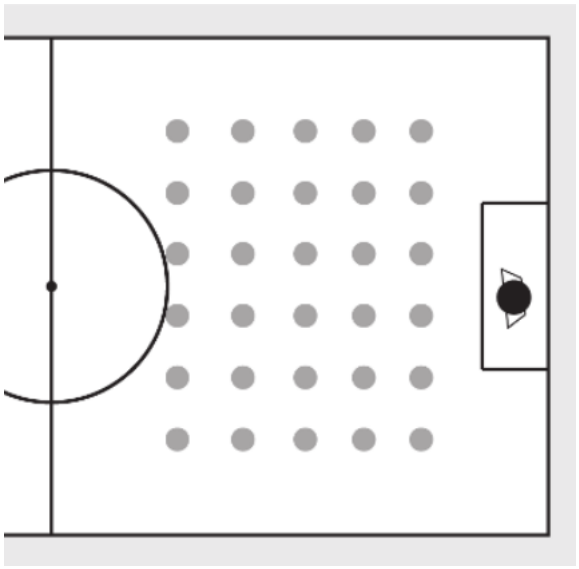


Figura 6.7: Implementación: Puntos de apoyo, fuente: [1]

La puntuación se otorga según:

  - Se puede pasar hacia esa posición desde la posición del receptor: 2 puntos.
  - Se puede marcar desde esa posición: 1 punto (para evitar tiros lejanos).
  - Lejanía respecto del pasador: 2 puntos base creciente según distancia.

- **Regatear.** Permite al futbolista continuar la jugada en caso de que no tenga opciones de pase. Para ello busca el oponente que esté justo delante suya y regatea hacia la izquierda si éste está a la derecha del regateador, o regatea hacia la derecha si el oponente está a la izquierda suya.

Además de buscar apoyo, los jugadores sin la posesión del balón actúan de alguna de las siguientes maneras:

- **Perseguir balón.** Utilizando el comportamiento "Perseguir"descrito anteriormente, persigue al balón si es el jugador de su equipo más cercano al esférico.

- **Volver a su región origen.** Utilizando el comportamiento perseguir, el jugador toma como objetivo su posición original en el campo.

- **Proteger portería.** En el caso de los porteros, haciendo uso del comportamiento Ïnterponerse", se coloca a una distancia fija de la línea de gol protegiendo ésta de las amenazas rivales según la posición de éstos.

- **Interceptar balón.** En el caso de los porteros, haciendo uso del comportamiento "Predecir", interceptar el balón en la posición futura en la que estará este.

- **Esperar.** En caso de que el futbolista no tenga ninguna responsabilidad espera acontecimientos.

6.2.2. Equipos

Además de los comportamientos individuales de los futbolistas, cada equipo puede adoptar tres tipos distintos de comportamiento según las circunstancias del juego:

- **Atacar.** Posiciona a los futbolistas en regiones del campo adelantadas con respecto a su portería.

- **Defender.** Posiciona a los futbolistas en regiones del campo cercanas a su portería.

Figura 6.8: Implementación: Atacar y Defender.

- **Balon parado.** Posiciona a los futbolistas en las regiones adecuadas según el tipo de saque (puerta, centro, banda, corner). Además deshabilita deshabilita el resto de comportamientos hasta que no se saque.

Figura 6.9: Implementación: Balón Parado.

Capítulo 7

Pruebas

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

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

También se realizaron muchas pruebas de integración, ya que había módulos, una vez probado en solitario, debían realizar distintas acciones junto con otros módulos. Como pueden ser las colisiones y físicas o la entrada del usuario. Esto requería de mucho esfuerzo ya que se empleaban librerías externas sumado a que la entrada del usuario es imprevisible y su interactuación con el entorno debe ser aceptable.

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

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

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

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

7.1. Pruebas unitarias

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

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

7.2. Pruebas de integración

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

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

En este apartado los principales problemas se encontraron a la hora de integrar las librerías externas con el sistema, sobre todo la librería de físicas y colisiones, que practicamente supusieron la mayor parte del tiempo de pruebas.

7.3. Pruebas de jugabilidad

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

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

- **Físicas**: se les pedía que comprobarán las potencias de disparo, pase según la distancia y la velocidad de los futbolistas.
- **Colisiones con entre futbolistas**: comprobación de las colisiones entre futbolistas en los distintos lances del juego.
- **Posiciones futbolistas**: correcto posicionamiento de los futbolistas en estados de ataque, defensa o saque a balón parado.
- **Inteligencia artificial**: la agilidad y dificultad de jugar contra la inteligencia de la IA.

La mayor parte de las personas que probaron el juego, opinaron que éste tenía una complejidad elevada para dificultar la victoria al jugador lo suficiente. Aunque opinaron que hacian falta cambios en algunas posiciones defensivas, así como la velocidad de los jugadores. Tras realizar los cambios propuesto, los colaboradores se mostraron más contentos con los resultados.

7.4. Pruebas de interfaz

Conforme se realizaban las distintas pantallas referentes a los menús que podemos encontrar en el juego, se realizaban pruebas exhaustivas sobre la interacción entre éstas. Sobre todo se comprobaban que las interfaces fueran intuitivas y claras. Así como comprobar la coherencia de elementos cambiantes como el sonido o el volumen de éste.

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

Capítulo 8

Conclusiones

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

8.1. Resumen de objetivos

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

También decir que el proyecto me ha ocupado bastante más tiempo del esperado en un principio. Tuve muchos problemas y dudas en algunas fases del desarrollo de proyecto, que me tuvieron bloqueado durante un tiempo hasta encontrar la solución más adecuada para éstos. He sido capaz de adentrarme a mirar

Se puede decir que con este proyecto he aprendido muchísimo en todas y cada una de los aspectos y fases que lo componen, destacando sobre el resto el diseño de este tipo de sistema, cómo interactúan sus componentes y cómo implementar esto de manera elegante y flexible.

8.2. Conclusiones personales

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

Además añadiría que este proyecto me ha preparado para enfrentarme al mundo laboral y concretamente al ámbito en el que me gustaría trabajar, teniendo una idea previa de la arquitectura, herramientas y patrones que se utilizan a menudo para el desarrollo profesional.

Puedo decir que he profundizado y consolidado bastante en el lenguaje de programación c++. Además he aprendido a manejar bibliotecas externas, así como entender su documentación y cómo integrarlas en un proyecto.

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

8.3. Mejoras y ampliaciones

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

- **Modo de dos jugadores**: añadir la posibilidad de que jueguen dos jugadores el mismo partido.
- **Modo en red**: también sería una buena idea añadir un modo de juego en el que pudiésemos jugar en red contra otros oponentes.
- **Mayor diversidad de regates**: añadir regates y animaciones que dotarían al juego de una mejor presencia.
- **Crear ambiente**: añadir un estadio con personas y efectos de sonido simulando un ámbiente futbolístico.
- **Grabación de las mejores jugadas para repetirlas**: implementar una opción que grabara las jugadas de gol almacenándolas en un fichero, y así poder visualizarlas posteriormente.

Apéndice A

Herramientas utilizadas

A.1. Lenguaje de programación

Una de las decisiones más importante que consideraba a la hora de desarrolla el proyecto, era la elección de un lenguaje de programación adecuado y que cumpliera todas las expectativas necesarias. Opté por C++ debido a:

1. Lenguaje que hemos visto y aprendido a lo largo de toda la carrera, y con más profundidad en la asignatura de Programación Orientada a Objetos. Por lo que ya se tenía una soltura y conocimientos previos que no se tenían con ningún otro lenguaje.

2. Lenguaje compilado por lo que su velocidad y eficiencia es mucho mayor que la de otros lenguajes. Aspectos muy a tener en cuenta a la hora de desarrollar un videojuego.

3. Multiplataforma

4. Conjunto de bibliotecas estándar muy amplias y muy bien documentadas.

Esta eficiencia y velocidad se hace aun más necesaria en el desarrollo de videojuegos 3D ya que demandan altas exigencias.

A.2. Motor de renderizado

Tras la decisión final del lenguaje de programación que se usaría a lo largo de todo el desarrollo del proyecto, la siguiente decisión de gran importancia que se debía tomar, era la elección demotor de renderizado que usaríamos en *Fútbol es Así*.

En este caso se tuvo clara la elección desde el momento en el que se decidió usar *C++* como lenguaje principal, me decanté por la biblioteca gráfica *Ogre3D*. Ésta es una biblioteca multiplataforma de código abierto para C++ que permite a los desarrolladores crear, manejar y mostrar escenas tridimensionales utilizando aceleración por hardware. Como ventaja principal, uso extenso uso, la gran comunidad que tiene y la extensa documentación. Además de la integración con software de modelado, texturizado y animación como Blender.

Figura A.1: Herramientas utilizadas: Logo de Ogre3D

Al ser una biblioteca orientada a objetos su manejo es bastante sencillo representando la escena como un árbol. Otra virtud es la flexibilidad que ofrece con su API y la facilidad de instalación ya que se puede encontrar en los repositorios oficiales de distribuciones como Ubuntu.

A.3. Físicas y colisiones

Se optó por *Bullet* como librería de físicas y colisiones ya que es libre, multiplataforma, flexible, optimizada, con gran comunidad y documentación. Posiblemente la biblioteca libre más completa. En este proyecto se utiliza se utiliza en su mayoría su integración en el wrapper *OgreBullet*.

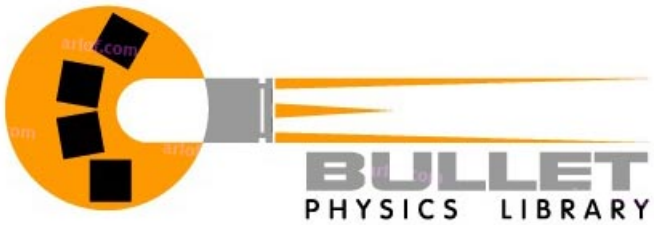


Figura A.2: Herramientas utilizadas: Logo de Bullet

A.4. Gestión entrada de usuario

Para detectar la entrada de usuario y manejarla, sobre todo, el joystick se optó por *OIS(Object-Oriented Input System)* que también cumple con ser libre, multiplataforma, flexible y eficiente.

A.5. Sonido

Para llevar a cabo la reproducción del sonido se empleó *SDLMixer*, una extensión de la famosa librería *SDL*. Destaca por su sencillez y flexibilidad siguiendo el patrón de libre, multiplataforma y eficiente.

A.6. Interfaz

La interfaz de los menús se llevó a cabo a través de *MYGUI*, una librería multiplataforma, libre y muy sencilla. A pesar de no ser la más completa, se ajustaba a las necesidades del proyecto.

A.7. Blender

La elección del programa de edición 3D era también importante ya que debía poder exportarse todo el contenido al motor de renderizado *Ogre*. Tras una investigación previa me decidí por Blender.

Blender es un programa informático libre, multiplataforma, dedicado especialmente al modelado, animación y creación de gráficos tridimensionales. Tiene una extensa comunidad y cuenta con multitud de documentación audiovisual que facilitan el complicado aprendizaje de la herramienta.
En el foro oficial de *Ogre* se desarrolló un exportador de Blender a *Ogre* que aunque cuenta con multitud de bugs, acabó cumpliendo con su cometido básico.

Figura A.3: Herramientas utilizadas: Logo de Blender

A.8. Gimp

GIMP (GNU Image Manipulation Program) es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un programa libre y gratuito. Está englobado en el proyecto GNU y disponible bajo la Licencia pública general de GNU.
Con él se han realizado texturas, botones, y todo el contenido 2D de elaboración propia.

Figura A.4: Herramientas utilizadas: Logo de Gimp

A.9. Sistema de control de versiones

Todo el código y recursos de *Fútbol Es Así* está alojado en el sistema que proporciona RedIris, que consiste en un entorno completo usando el sistema de control de versiones *subversion*.

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

Se evaluaron otros sistema de control de versiones como pueden ser GIT o mercurial , pero finalmente me decanté por subversión, ya que pensé en la inscripción en el Concurso Nacional de Software Libre que requiere usar RedIris.

A.10. Documentación del código

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

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

A.11. Redacción de la memoria.

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

Figura A.5: Herramientas utilizadas: Logo de LaTeX

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

A.12. Realización de diagramas: Dia

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

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

Figura A.6: Herramientas utilizadas: Logo de Dia

Apéndice B

Manual de instalación

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

Debido a la alta complejidad de empaquetar un proyecto en Ogre a un .deb, se procede a explicar como instalarlo a partir del código fuente:

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

En primer lugar hay que descargar SVN, para instalarlo, ejecutamos la siguiente orden en una terminal:

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

En segundo lugar, debes descargar el código fuente del repositorio del juego en la forja de RedIRIS:

```
svn checkout https://forja.rediris.es/svn/cusl5-fea
```

El videojuego cuenta con las siguientes dependencias:

- libogre-dev
- libsdl1.2-dev
- libsdl-mixer1.2-dev
- mygui
- libbullet-dev

Se instalarán las dependencias utilizando el gestor de paquetes.
Posteriormente instalar *OgreBullet* tal y como describo en esta url:
http://osl2.uca.es/iberogre/index.php/Colisiones_y_F%C3%ADsicas_con_OgreBullet
Una vez están instaladas todas las dependencias se procede a ejecutar sobre el directorio del proyecto:

```
make
./futbolesasi
```

Apéndice C

Manual de usuario

C.1. Menú principal

Desde el menú principal se podrá acceder al modo partido de *Fútbol Es Así* y la información sobre los desarrolladores del proyecto.

Figura C.1: Manual de usuario: Menú principal

Debe usar el ratón para seleccionar la opción que desee. Además se podrá silenciar la música del menú pulsado en el icono del volumen.

C.2. Pantalla de juego

Una vez comencemos un partido pasaremos a la pantalla de juego.

Figura C.2: Manual de usuario: Pantalla de juego

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

- **Marcador**: en la parte superior izquierda, el marcador actual del partido.
- **Cronómetro**: en la parte superior izquierda encontramos el tiempo transcurrido del partido.

A partir de este momento el juego se controlará a través de un joystick, con los siguientes controles:

- **Equis**: Pasar balón.
- **Cuadrado**: Golpear el balón de forma gradual según tiempo de pulsación.
- **Start**: Menú Pausa.
- **L1**: Cambiar Jugador.
- **R1**: Correr.
- **Dirección**: Dirección en la que moverse.

Bibliografía

[1] Mat Buckland. *Programming Game AI by Example*, 2004. 495 p. ISBN:978-1556220784.

[2] Gregory Junker. *Pro OGRE 3D Programming*, 2006. 284 p. ISBN:978-1590597101.

[3] F. Kerger. *OGRE 3D 1.7 Beginner's Guide*, 2010. 300 p. ISBN:978-1849512480.

[4] VideoTutoriales Modelado, Texturizado, Rigging y Animación en Blender. http://cgcookie.com/blender/.

[5] Página oficial de la herramienta para documentar código *Doxygen*. http://www.stack.nl/~dimitri/doxygen/

[6] Guía para la generación de la memoria del Proyecto Fin de Carrera. http://osl2.uca.es/wikiformacion/index.php/LaTeX_para_Proyecto_Fin_de_Carrera.

[7] Referencia del lenguaje de programación C++. http://www.cplusplus.com/reference/

[8] Página sobre la herramienta de generación de diagramas *Dia*. http://projects.gnome.org/dia/

[9] Página sobre la herramienta para realizar bocetos *Pencil Project*. http://pencil.evolus.vn/en-US/Home.aspx

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

[11] Jason Gregory. *Game Engine Architecture*, 2009. 864 p. ISBN:978-1568814131.

GNU Free Documentation License

Version 1.3, 3 November 2008

<http://fsf.org/>

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.
The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".
Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.
The "**publisher**" means any person or entity that distributes copies of the Document to the public.
A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

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

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.
If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.
It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

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

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.
You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

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

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.
If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.
However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.
Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.
Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.
"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.
"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.
An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.
The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

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

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

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

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

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.