UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Ingeniería en Informática

Proyecto Fin de Carrera

“Diseño e implementación del juego StarCraft mediante la herramienta Microsoft XNA 3.1”

Autor: Iván Yusty Tenorio

Tutor: Juan Peralta Donate

Departamento de Informática

Junio, 2011

Resumen

Si bien no existe acuerdo acerca de cuál fue el primer videojuego, sí se tiene la certeza de que tuvo su origen entre 1950 y 1970. Aquella época aportó títulos como Lanzamiento de misiles[r1], Tennis for Two[r2], Spacewar![r3] o Pong[r4], que utilizaban más principios electromecánicos que software. A lo largo de los años, los descubrimientos en materia de hardware, las mejoras a nivel de software y, sobre todo, el espíritu innovador han impulsado sobremanera el trabajo del sector.

A lo largo de los últimos quince años, los videojuegos se han consolidado como uno de los sectores más rentables de la industria. Esto ha llevado a que se produzca un gran crecimiento y diversificación del negocio. Un sector que hace apenas siete años basaba sus beneficios en el desarrollo de productos comerciales clásicos[1] ha evolucionado introduciendo nuevos modelos: los videojuegos casuales[2], bajo demanda[3], arcade[4], indie[5], sociales[6]…

Este proyecto tiene como objetivo el desarrollo de un cliente del juego **StarCraft**[r5], realizado en 1998 por Blizzard Entertainment. Esta producción nace como un juego de estrategia en tiempo real. En ella se narra la historia del futurístico conflicto entre tres razas espaciales, dejando al jugador el control total sobre cualquiera de ellas a su elección.

Para la creación de StarCraft se tomaron ideas de Dune 2[r6], que fue el primer desarrollo en tiempo real que permitió jugar con total libertad de acción: se podían mover las unidades por todo el mapa, se podía atacar cualquier objetivo y se podían construir tantos edificios o unidades como fuera necesario. Otras creaciones que inspiraron la obra central de este proyecto fueron Command & Conquer[r7], Age of Empires[r8] y WarCraft II[r9]: de éstas se tomaron ideas de gráficos, menús, misiones, interfaz… Desde su publicación, StarCraft ha tenido una base de jugadores numerosa y participativa. Recientemente se ha publicado su segunda parte, StarCraft 2[r10], siendo uno de los eventos más importantes del sector durante el año 2010.

El desarrollo planteado en este proyecto consiste en la implementación de un cliente de juego de StarCraft. En él se implementan las opciones disponibles para el jugador local, dejando como líneas futuras el trabajo en red y la inclusión de inteligencia artificial. Las tecnologías utilizadas para la implementación son los productos de Microsoft **XNA Game Studio**[7], **.NET Framework**[8] y **Visual C#**[9]. Una vez desarrollado, el producto puede ser clasificado como videojuego indie de estrategia en tiempo real[10].

Índice general

Índice de figuras

Índice de tablas

Índice de código

Capítulo 1

Introducción

En este capítulo se presenta el proyecto: se tratan aspectos como la motivación inicial, los objetivos planteados y los contenidos recogidos en este documento.

El primer apartado de este capítulo, **1.1. Motivación del proyecto**, expone los motivos que llevaron al alumno a seleccionar este proyecto en concreto.

El segundo apartado, **1.2. Objetivos**, detalla los objetivos planteados inicialmente. Será necesario verificar el cumplimiento de dichos puntos de control para dar por finalizado el proyecto.

Por último, el tercer apartado, **1.3. Contenidos de la memoria**, introduce la distribución y contenidos de este documento, así como una visión general de cada capítulo.

1.1. Motivación del proyecto

El videojuego es, hoy por hoy, la máxima expresión del ocio electrónico. La llamada "industria del videojuego" es el sector empresarial encargado del éxito que actualmente tienen los juegos de ordenador, de consola o de móvil. Esta industria está compuesta por multitud de empresas que realizan labores de diseño, desarrollo, comercialización, distribución, venta, mantenimiento… de hardware o software relacionado con videojuegos. Como se verá, el sector ha experimentado en los últimos años una profunda evolución. El desarrollo de la computación, el aumento de la capacidad de cálculo, la evolución de las redes o el auge de las redes sociales son sólo algunos ejemplos de los factores que han acompañado el proceso de cambio.

Al adaptarse satisfactoriamente al cambiante mercado, las tasas de crecimiento y de penetración sólo pueden ser comparadas a las que reflejan los beneficios. Hoy en día, el sector es uno de los más pujantes en lo que al ocio se refiere. Este hecho destaca especialmente cuando se habla de las costumbres del mercado en general, superando ampliamente la actividad de sectores similares (cine, música…). Por ejemplo:

- El récord de taquilla el día del estreno es ostentado por la película "Harry Potter y el Misterio del Príncipe", y se cifra en 104 millones de dólares. Su coste de producción fue de unos 250 millones de dólares [r11]. Como contrapartida está "Call of Duty: Modern Warfare 2" que, en las mismas condiciones, consiguió 400 millones de dólares mediante la inversión de unos 50 millones de dólares [r12].
- La película que más ha recaudado en toda la historia ha sido "Avatar", rondando los 2.776 millones de dólares [r13]. El videojuego más vendido, "Mario Bros.", supera los 193 millones de copias: a 55 dólares la copia (aproximadamente 40 euros en tiempo de redacción de este documento), la recaudación asciende a 7.720 millones de dólares [r14].

Realizando un análisis del sector en España se pueden apreciar cifras similares. Según destaca la Asociación de Desarrolladores y Editores de Software de Entretenimiento [r15] (AdEsE):

- España es la cuarta potencia europea en consumo de videojuegos.
- La cuota de mercado del videojuego es la opción de ocio electrónico más elegida desde 2002, y mantiene la mayoría absoluta desde 2007 frente al resto de oferta: cine, películas (venta y alquiler) y música.
- El crecimiento del sector y sus beneficios, de manera pareja a la cuota de mercado, han representado la opción mayoritaria y han ido aumentando a lo largo de los años.

- Se ha observado un crecimiento muy positivo del sector desde 2002 a 2010 [r16]. La media del dato se sitúa en un 12,1%, con picos superiores al 20%, aunque también se registra una caída importante: -16,2% entre 2008 y 2009.
- Durante el año 2010 el crecimiento ha sido levemente positivo (5% [r17]), aunque se prevé una mejora de este dato debido a las festividades navideñas.

El crecimiento y estabilidad comentados muestran un futuro estable en el mundo de los videojuegos. Esta solidez es el fruto de la evolución y expansión del modelo económico. Durante los años 80, 90 y principios del 2000, la industria del videojuego tenía su mayor fuente de ingresos en la salida de nuevas consolas y tendía a decaer rápidamente hasta el comienzo de una nueva iteración del modelo (nueva consola, nuevo terminal…). Para paliar esta inestabilidad, la industria se ha implicado muchísimo, adaptándose gradualmente a los nuevos tiempos: se proporciona contenido descargable, se mejoran y revenden juegos antiguos, se utilizan los nuevos servicios (Internet, redes sociales…), se diseña pensando en lanzar nuevos terminales que posibiliten nuevos modos de juego… En definitiva, la industria ha conseguido crecimiento y estabilidad a través del aumento del público objetivo y de la dinamización del modelo productivo.

Una de las situaciones que han potenciado este cambio de modelo de negocio ha sido el hecho de que los propios usuarios han comenzado a expresar sus gustos e ideas en forma de juegos casuales y/o indie. Este curioso fenómeno espontáneo viene inspirado por el acercamiento entre el mundo tecnológico y la población en general. Con los años, las diferentes tecnologías hardware y software han perdido su carácter oscuro. Al mismo tiempo, han pasando de ser raras a ser útiles, y de ahí a ser imprescindibles. Sea como fuere, la industria ha tomado nota de estas ideas en la medida de lo posible. Como consecuencia se tienen, por ejemplo, los juegos incrustados en páginas web de gran afluencia, los desarrollos para móviles, los videojuegos descargables de las consolas de última generación…

En una suerte de simbiosis, el desarrollo de juegos casuales y/o indie por parte de la comunidad de jugadores se ha visto potenciado gracias al interés mostrado por parte de la industria. Las empresas, observando cierta inquietud acerca del software libre[11] y la creación de juegos por parte de aficionados, decidieron desarrollar diferentes librerías. Estos desarrollos, aunque son propietarios[12], facilitan el acceso de los programadores aficionados a las plataformas comerciales. Y esto, aunque es una visión limitada, representa un gran cambio en la lógica de negocio del sector. Las empresas dan un paso hacia los desarrolladores y jugadores, quienes sienten que son bien recibidos y que pueden colaborar con las creaciones que se les ocurran.

Este es el punto donde tiene su base este proyecto: es el primer paso de un desarrollador aficionado en la industria del videojuego. Para este caso se utilizará la colección de herramientas ofrecidas por Microsoft: XNA Game Studio, Visual C# y .NET. Este sistema permite desarrollar para las plataformas PC/Windows, XBOX360, Zune, Windows Phone y Windows Mobile. El objetivo de este proyecto es analizar, diseñar y

construir un cliente del videojuego “StarCraft” de Blizzard Entertainment. Al concluir, el resultado se ajusta a la descripción de juego indie de estrategia en tiempo real de tipo wargame[13].

1.2. Objetivos

En este punto se recogen los objetivos a cumplir mediante la realización de este Proyecto de Fin de Carrera. Estos requisitos son la base para planificar el trabajo y sus etapas, por lo que se desglosan hasta el máximo nivel de detalle. De esta manera se crea la base necesaria para evaluar el progreso a lo largo del tiempo. El proyecto estará terminado cuando, finalmente, todos los objetivos hayan sido superados.

En primer lugar, el objetivo que dirige el proyecto es el deseo de completar las enseñanzas curriculares exigidas para obtener el título de Ingeniero Superior en Informática. Partiendo de este requisito se decidió llevar a cabo el Proyecto de Fin de Carrera, ya que éste es parte fundamental y final del programa de la carrera.

En segundo lugar, se desea llevar a cabo un proyecto informático de gran envergadura. Tras dar por válido el primer objetivo, se analizó el conjunto de opciones disponible:

- Es posible realizar el Proyecto de Fin de Carrera mediante la participación en un proyecto de investigación dentro de la Universidad. Esta idea se aleja de la concepción inicial del alumno, por lo que fue descartada.
- Por otro lado, se puede participar en un proyecto dentro de una empresa. Esta opción, inicialmente más recomendable, fue descartada debido a la escasez de ofertas interesantes, así como por el deseo de experimentar de primera mano todos los eventos de un proyecto completo.
- La tercera opción, como se ha comentado, consiste en la realización de un proyecto informático en toda su envergadura. Esta opción fue escogida por ser capaz de mostrar al alumno las contingencias que ocurren en un proyecto real: necesidad de planificar y presupuestar, desviación de tiempo y costes, toma de requisitos, cambios en los requisitos, decisión de la solución tecnológica aplicada, problemas técnicos asociados…

En tercer lugar, se quiere dotar al alumno de la experiencia necesaria para ingresar en un sector laboral interesante y estable. Este objetivo se cree de bastante importancia dados los tiempos que corren, aunque también esta relevancia es inherente al término de unos estudios de segundo ciclo universitario. Las ideas observadas fueron diversas:

- El estudio y aplicación de la tecnología al procesamiento distribuido de datos. Esta alternativa fue descartada porque, si bien es interesante para el alumno y

había proyectos asequibles, no se apreció demasiado impacto en el mundo laboral.

- El desarrollo de aplicaciones para plataformas móviles[14]. Esta opción, actualmente en auge, fue descartada por no haber ningún proyecto libre ni perspectivas concretas sobre cuándo podía haber alguno.
- El desarrollo de un videojuego. Este caso, finalmente seleccionado, tiene como motivación principal el interés personal del alumno por el mundo del ocio electrónico. Otras causas que han llevado a realizar esta elección son el crecimiento estable del sector y que, en cierta medida, se descubrió compatible el desarrollo de videojuegos con el desarrollo de aplicaciones para plataformas móviles.

En cuarto lugar, habiendo decidido la temática general del proyecto a realizar, se optó por implementar el cliente de un videojuego conocido: StarCraft. Esta opción se decidió de manera conjunta con el tutor del proyecto: su idea consistía en realizar un videojuego de estrategia en tiempo real. Por otro lado, el alumno escogió personalmente el título por preferencia personal, así como por el reconocimiento internacional que acumula.

En quinto lugar, se detectó la necesidad de conocer el mundo del videojuego (historia, plataformas, títulos…), tanto en general como en el caso particular de StarCraft. El cumplimiento de este requisito permite realizar una redacción óptima del capítulo 2, **Estado de la cuestión**.

Finalmente, en sexto y último lugar, es objetivo del proyecto construir una adecuada base de conocimiento sobre las herramientas de diseño y construcción de videojuegos. De esta manera se podrán distinguir los distintos tipos de herramientas, cada una asociada a una determinada actividad del proyecto: sistemas de sonido, sistemas gráficos, entornos de desarrollo, herramientas de edición o de pruebas… En este sentido, también se considera provechoso profundizar en el conocimiento de Visual Studio, de .NET y de C#, ya que son tecnologías muy utilizadas que se han trabajado poco a lo largo de la carrera.

Para conseguir los objetivos planteados se ha confeccionado, con ayuda del tutor, la siguiente serie de requisitos intermedios:

- **Profundizar en el conocimiento de Visual Studio, C# y XNA.** Al comenzar el proyecto se detectó cierta falta de conocimientos en este aspecto. Esto viene motivado por haber trabajado en pocas o ninguna ocasión con estas herramientas a lo largo de la carrera. Por tanto, el primer paso es afianzar conceptos y profundizar en los conocimientos y la mecánica de trabajo de dichas herramientas.
- **Efectuar un análisis de requisitos.** Se debe realizar un análisis de requisitos para el juego que determine las funcionalidades que incluye: qué interacciones habrá entre juego y jugador, qué unidades y edificios será posible controlar,

qué comportamiento tendrán las unidades y los edificios, cómo serán los controles, cómo será la presentación en pantalla…

- **Realizar el desarrollo y la validación.** Durante el desarrollo se cubrirá un conjunto de hitos. Estos hitos, una vez completados, serán entregados como versiones intermedias incrementales al tutor. La verificación del cumplimiento de los requisitos planteados generará la información necesaria para el buen desarrollo del proceso y del producto.
- **Estimar y planificar los costes y tareas del proyecto.** Esta fase, consistente en el análisis y elaboración de una planificación detallada de hitos del proyecto, debe ser abordada en cuanto sea posible. Será un objeto de referencia a la hora de comprobar el avance del proyecto, por lo que también debe ser actualizada tan pronto como se detecte cualquier suceso que pueda afectarla.

Se desglosan a continuación los hitos más importantes del proyecto. Estos hitos servirán para marcar y guiar el avance del proyecto. Por tanto, siguiendo una filosofía de análisis por reducción, se han enfocado como subconjuntos exclusivos de los objetivos intermedios.

- Conocer los métodos básicos que ofrece XNA y la secuencia típica de llamadas de un videojuego.
- Conocer las fuentes de recursos que se pueden utilizar en XNA e introducirlas en Visual Studio.
- Diseñar el sistema de menús: qué páginas contiene, cómo se organizan las páginas, para qué sirve cada una, cómo se traduce el diseño a clases codificables.
- Diseñar la barra de control: de qué se compone, cómo se ha de actualizar, qué interacciones debe mostrar.
- Diseñar el mapa: generación del mismo, actualización de sus elementos e integración del jugador.
- Diseñar las unidades: movimiento, selección y acciones específicas de cada unidad. Integración con el jugador. Estudio de los recursos gráficos necesarios.
- Diseñar los edificios: selección y acciones específicas de cada edificio. Comparación con el diseño de unidades. Integración con el jugador. Estudio de los recursos gráficos necesarios.
- Diseñar el movimiento de las unidades. Detección de colisiones y búsqueda de caminos. Estudio de los recursos gráficos necesarios.
- Diseñar otros recursos y su integración: elementos cohesivos de sonido, iconos diversos, pantallas de transición…
- Evaluar el desarrollo: el juego, la usabilidad, el comportamiento de los enemigos…

Para concluir la sección es necesario destacar dos objetivos de carácter global. Estas cláusulas deben ser observadas a lo largo de todo el proyecto.

- **Viabilidad del proyecto.** Se debe realizar el estudio previo de todos los elementos que afectan al proyecto. Esto servirá para determinar la viabilidad del proyecto, así como para crear una planificación adecuada. Para más información sobre este objetivo, consultar el **Anexo A Planificación**.

- **Estudio de desviaciones sobre planificación:** De manera similar al objetivo anterior, es necesario analizar las desviaciones de tiempo respecto del calendario originalmente estimado, aportando conclusiones de valor que permitan evitar futuros problemas.

1.3. Contenido de la memoria

El contenido del presente documento está organizado en forma de capítulos. Cada uno de ellos profundiza sobre un aspecto concreto relacionado con el proyecto. A continuación se enumeran los capítulos creados junto a un breve resumen del contenido de cada uno.

El capítulo 1, **Introducción**, se presenta una visión global del proyecto y de su situación. Tienen cabida aspectos generales como son la idea en torno a la cual gira el proyecto, un estudio de la industria del videojuego, un análisis de posibilidades futuras y el estudio de los objetivos marcados para la realización de este proyecto.

El capítulo 2, **Estado de la cuestión**, presenta aspectos que tienen una relación directa con el proyecto y que sirven para orientar al lector acerca del producto que se desarrolla. Por tanto, se realiza una completa taxonomía del videojuego, una reseña histórica de la evolución del género de videojuegos de estrategia en tiempo real y un análisis de StarCraft y de Blizzard Entertainment. Después se estudia la plataforma de desarrollo, Microsoft XNA: fundamentos, ventajas, inconvenientes, alternativas y los motivos que han conducido a utilizar este instrumento.

El capítulo 3, **Fase de Análisis**, se recoge el catálogo de requisitos de usuario del proyecto. Después se explotan adecuadamente, dando lugar a los requisitos software y a los casos de uso. Se concluye el capítulo con los diagramas de actividad y de secuencia del programa, elementos que dan una información global acerca del estado del programa y de las funcionalidades a implementar.

El capítulo 4, **Fase de Diseño**, es el lugar donde se expone todo conocimiento útil para implementar las funcionalidades que componen el proyecto. En este capítulo tienen cabida elementos como los bocetos de diseño de interfaces, las arquitecturas de alto y bajo nivel del sistema y el diagrama de clases que se extrae de la información recogida en el capítulo 3.

El capítulo 5, **Fase de Implementación**, es el lugar donde se detallan los aspectos más importantes del desarrollo. En vez de explicar la actividad y el cometido de cada función y de cada variable, en este capítulo se ha escogido aportar una visión general de los elementos implementados: el tratamiendo de imágenes, la definición, dibujado y desplazamiento del mapa y la utilización de unidades y edificios.

El capítulo 6, **Conclusiones**, está dedicado a recoger las reflexiones del autor tras realizar el proyecto. En este capítulo se evalúa la consecución de los objetivos planteados. Estos objetivos también son utilizados como guión para analizar el desarrollo del proyecto, volcando los pensamientos asociados que han surgido.

El capítulo 7, **Líneas Futuras**, es el lugar dedicado a plasmar las ideas que surgen a partir de lo desarrollado en este proyecto. Este capítulo se ofrece como reflexión para orientar futuros desarrollos, comentando el juego realizado y las tendencias más importantes dentro del sector.

Para concluir el documento se ofrece el conjunto de Anexo: en primer lugar, el Anexo A recoge la **Planificación** asociada al proyecto; después, el Anexo B contiene el **Manual de Usuario**; por último, el Anexo C recoge el **Glosario** del proyecto, así como las **Referencias** y la **Bibliografía** empleadas.

Capítulo 2

Estado de la cuestión

Una completa taxonomía del videojuego, una reseña histórica del género de estrategia en tiempo real, un análisis del videojuego StarCraft y un análisis de XNA como herramienta de desarrollo componen este capítulo.

El primer apartado, **2.1. Taxonomía del videojuego**, detalla la clasificación y rasgos principales de los géneros de videojuegos existentes.

El segundo apartado, **2.2. Historia del videojuego de estrategia en tiempo real**, contiene una crónica histórica detallada de este género de videojuegos.

El tercer apartado, **2.3. El videojuego StarCraft**, incluye el análisis realizado sobre el juego que sirve de base para este proyecto.

Por último, el cuarto apartado, **2.4. XNA Game Studio**, presenta las distintas herramientas existentes para el desarrollo de videojuegos y justifica la elección de XNA.

2.1. Taxonomía del videojuego

Hoy en día existen miles de videojuegos: cada uno de ellos entraña infinidad de detalles a considerar. Muchos se muestran como meros clones de las grandes sagas pero, a pesar de ello, la mayoría tiene algo único que los diferencia. Con la ingente cantidad de juegos que existe, se ha creído de gran ayuda hacer una clasificación general, observando las diferencias y similitudes entre géneros, así como las diferentes "especies" que conviven dentro del mismo género.

La taxonomía elaborada posibilita la visión de los videojuegos desde una nueva perspectiva. Es más, por su propia naturaleza, deja al descubierto las leyes elementales del campo que se estudia. Destacando la similitud con otro campo, el estudio taxonómico de Charles Darwin durante su análisis de la fauna le catapultó al desarrollo de la Teoría de la Evolución de manera inevitable. Y, aunque no se cree que analizar las características de los videojuegos pueda llevar tan lejos, sí es cierto que aporta conclusiones de alto valor al diseño software de este tipo de creaciones.

2.1.1. Shoot´em Up

Género traducido literalmente como "Dispárales". También es conocido como "Shooter" o "Juego de disparos". El objetivo en estos juegos es matar a base de disparos todo objetivo evitando morir uno mismo en el intento. El género fue popularizado por los clásicos juegos de naves (Gradius[r18], Galaxian[r19]…). En la actualidad existen tres variantes principales.

Figura 1. Escena de Metal Slug X

FPS o **First Person Shooter (Disparos en primera persona):** Este subgénero ha conseguido mucha popularidad en los últimos años gracias a la saga Halo[r20], de

Microsoft. La mecánica básica es la misma que en cualquier juego de disparos pero, como indica el nombre, se disfruta la acción desde un punto de vista subjetivo en primera persona que sumerge al jugador en la historia narrada.

Figura 2. Escena de Call of Duty 7: Black Ops

TPS o **Third Person Shooter (Disparos en tercera persona):** A pesar de que históricamente la mayoría de los juegos de disparos presentaban la acción desde un punto de vista externo, este subgénero es relativamente reciente. El uso del punto de vista subjetivo situado detrás del jugador, mencionado siempre como "tercera persona", ha adquirido cada vez más popularidad junto al auge de sagas como Max Payne[r21] o Gears of War[r22].

Figura 3. Escena de Max Payne 2: The Fall of Max Payne

Rail Shooter (Disparos sobre raíles): Este subgénero engloba desde juegos clásicos (R-Type[r23]) hasta no tan clásicos (saga The House of The Dead[r24], saga Time Crisis[r25]). En este caso, la acción puede ser desarrollada en primera persona, en tercera o en una combinación de ambas, dado que estos títulos tienen otra característica común: el jugador tiene un control relativo sobre su movimiento. Como indica el nombre del subgénero, el movimiento por el escenario se realizará sobre raíles, pudiendo únicamente modificar levemente la posición propia dentro de un recorrido predefinido.

Figura 4. Escena de Time Crisis 4

2.1.2. Beat´em Up

Género traducido literalmente como “Pégales”. Esta categoría de videojuegos se caracteriza por su elemento principal: el combate físico entre los propios personajes y los enemigos. Dicho enfrentamiento, como se verá, se encuentra cada vez menos limitado, ya que desarrollos actuales incluyen diversas armas u objetos. El objetivo de este tipo de juegos es sobrevivir con el propio personaje a todos los eventos que puedan presentarse.

Figura 5. Escena de Street Fighter 4

Lucha uno contra uno: También conocidos como "Juegos de lucha". Este subgénero engloba títulos emblemáticos como la saga Tekken[r26], la saga Mortal Kombat[r27] o la saga Street Fighter[r28]. El elemento principal son las luchas entre personajes uno a uno, aunque pueden desarrollarse de diferentes maneras. Existen juegos en los que hay más de dos combatientes: la saga King of Fighters[r29], por ejemplo, incluye un sistema de relevos. Otros desarrollos, como Super Smash Bros.[r30], muestran una filosofía de todos contra todos. El objetivo se mantiene inalterado: agotar la resistencia del enemigo antes de que el enemigo haga lo propio con la resistencia del jugador.

Figura 6. Escena de Tekken 5

Lucha con progreso: Es el subgénero original de lucha, así como el más conocido. El objetivo es avanzar linealmente con el personaje propio por los distintos escenarios, mientras salen al paso diferentes personajes o elementos enemigos. Eventualmente se encontrará un enemigo que, al ser vencido, provocará un avance en la historia: de ahí el nombre del subgénero. Algunas figuras clásicas son Double Dragon[r31], Final Fight[r32] o Battletoads[r33]. Actualmente, este género ha visto pocos desarrollos relacionados, por lo que está prácticamente extinto.

Figura 7. Escena de Battletoads

Hack & slash: Traducido literalmente como "Destroza y corta", este subgénero es una evolución muy característica del anterior (Lucha con progreso). Las principales diferencias estriban en su no linealidad: se tiene mayor libertad de acción, de armas… También se suele incorporar, como parte de la historia del juego, cierta evolución en las habilidades del personaje propio. Actualmente, los mejores ejemplos del subgénero pueden encontrarse en Devil May Cry[r34] o en la saga God of War[r35].

Figura 8. Escena de Devil May Cry 4

2.1.3. Estrategia

Los juegos de estrategia son el único género que, hasta la fecha, no ha conseguido ser implementado en una consola sin perder parte del encanto que lo hace tan popular entre los usuarios de plataformas de sobremesa. El objetivo que se plantea es siempre el mismo: completar un conjunto de hitos verificables. Para ello, el jugador debe controlar los recursos puestos bajo su mando, gestionando las tareas que se

realizan y regulando la producción según sea necesario. La acción suele disfrutarse desde un punto de vista elevado, alejado de la acción, desde el que se puede controlar una amplia zona de juego.

Figura 9. Escena de Command & Conquer 3

Real Time Strategy (RTS o **Estrategia en Tiempo Real):**

Los juegos de tipo RTS se caracterizan por el desarrollo de la acción en tiempo real. Esto no quiere decir que se tarden varios años en realizar las acciones: se hace referencia a la ausencia de pausa automática. Algunos juegos que exhiben este comportamiento son Command & Conquer, Warhammer 40.000: Dawn of War[r36] o el propio StarCraft, objetivo de este proyecto y que será comentado en profundidad más adelante.

Figura 10. Escena de StarCraft 1

Turn Based Strategy (TBS o **Estrategia Basada en Turnos):**

Al contrario que ocurre con los juegos de estrategia en tiempo real, estos juegos implementan un sistema de turnos más o menos rígido que determina la libertad de acción del jugador. Las pausas se realizan de manera automática, aunque es común poder realizarlas, además, voluntariamente.

El mecanismo de turnos estará basado en unas reglas, a saber: dependerá de los parámetros de cada jugador, se determinará aleatoriamente al comienzo de la partida, será alterable por los propios jugadores… Cada jugador dispondrá de un crédito limitado en su turno para ordenar acciones (mover, atacar, construir, cambiar formación…). Este crédito también suele verse modificado por diversos factores: ambientales, raciales, basados en logros o en experiencia… Finalmente, los eventos propiamente dichos pueden ser resueltos en tiempo real o por turnos. Como ejemplo del género cabe destacar una de las sagas más longevas, exitosas y conocidas: Heroes of Might & Magic[r37].

Figura 11. Escena de Heroes of Might & Magic 5

Simuladores estratégicos:

Esta vertiente de la estrategia recoge los juegos que tratan de emular una cierta realidad (ciudades, clubes de fútbol, parques de atracciones, compañías aéreas, colonias de hormigas…) sin atenerse a las reglas de juego en tiempo real o por turnos. La velocidad de juego será configurable en todo momento, pudiendo variar entre cero (juego pausado) y el límite máximo que haya sido implementado.

Este subgénero suele mantener un cierto progreso del jugador en base a objetivos o misiones. Sin embargo, a diferencia de otros subgéneros, no suele haber un

final establecido. El jugador puede jugar tanto tiempo como desee mientras cumpla los objetivos planteados.

El rol desempeñado por el jugador siempre será preponderante, teniendo el control de todo lo que sucede. El título concreto puede variar entre unos u otros cargos en función de la habilidad o de la voluntad del jugador.

Algunos títulos representativos son la saga SimCity[r38], Football Manager[r39], Theme Park World[r40], la saga Los Sims[r41].

Figura 12. Escena de SimCity 4

2.1.5. Conducción

Los juegos de carreras son tan antiguos como el ocio electrónico en sí: en todo momento de la historia ha existido algún desarrollo de referencia de este tipo, tanto en salones recreativos como en máquinas de sobremesa o portátiles. En estos juegos, el usuario toma el control de algún tipo de construcción de velocidad dirigible (automóvil, moto, avión, helicóptero, bicicleta, monopatín…) y tiene por objetivo ganar todas las carreras o eventos existentes.

Figura 13. Escena de Need for Speed Carbono

Simuladores de conducción realista:

Este subgénero se caracteriza por el realismo que se desea transmitir al jugador. Incluso dentro de un objetivo tan concreto, existe una amplia diversidad de títulos: pueden estar centrados en moto GP, fórmula 1, rally, navegación aérea, operaciones de rescate… Esta es la categoría propia de juegos como la saga Gran Turismo[r42] y la saga Flight Simulator[r43].

Figura 14. Escena de Gran Turismo 5

Simuladores de conducción arcade:

Este subgénero trata de realizar simulaciones de conducción, aunque aparece la marca de la palabra arcade: se prima la diversión por encima del realismo. Por tanto, los juegos de este subgénero permitirán realizar maniobras imposibles, atravesar diferentes dimensiones, atacar… con el objetivo de hacer más dinámica y divertida la experiencia del usuario. Algunos títulos representativos son la saga Mario Kart[r44], la saga Need for Speed[r45] y Midnight Club[r46].

Figura 15. Escena de Mario Kart Wii

2.1.6. Deportivo

El género de juegos deportivos data de los inicios del sector. Cada deporte, tan característico por sí mismo, podría considerarse un subgénero propiamente dicho. Sin embargo, hay multitud de deportes, por lo que se ha visto la necesidad de retratar las características típicas que reúnen los juegos deportivos en general.

Figura 16. Escena de Tiger Woods PGA Tour 12

Simuladores deportivos:

Estos juegos tienen por objetivo realizar una simulación realista de los eventos deportivos. De esta manera, el jugador siente que realmente está jugando al deporte en cuestión, sea desempeñando un rol fijo o uno variable. Los jugadores sentirán cansancio, el tiempo transcurrirá más o menos deprisa, la moral influirá en el rendimiento del equipo, los efectos ambientales podrán jugar malas pasadas… Elementos representativos de esta categoría son la saga Pro Evolution Soccer[r47], la saga FIFA[r48] y la saga NBA Live[r49].

Figura 17. Escena de NBA Live 10

Arcade deportivo:

En este subgénero se quieren englobar todos aquellos juegos que simulan la realización de una actividad deportiva de manera arcade. En esta categoría, por tanto, entran los títulos en los que prima la diversión y la accesibilidad por encima precisión, realismo y otros elementos. Los títulos suelen retratar una imagen fantástica del deporte en cuestión. Sagas como NBA Jam[r50], FIFA Street[r51], y Blitz[r52] pertenecen a este subgénero, en el que cada día se pueden identificar más ejemplares.

Figura 18. Escena de Tony Hawk's Pro Skater 4

2.1.7. Role-Playing Game

Género traducido literalmente como "Juegos de Rol", y referido popularmente como RPG. Estos juegos tienen su base en los juegos de rol de mesa tradicionales, evolucionando a partir de ellos. El objetivo, por tanto, es desarrollar a uno o varios

personajes a lo largo de la historia que narra el juego. Este perfeccionamiento vendrá influenciado por la superación de retos, la consecución de logros… que darán lugar a recompensas y a avances en la historia del juego. El jugador se convierte en el director de una especie de película parcialmente interactiva, lo que contrasta levemente con lo observado en los juegos de rol en la vida real.

Figura 19. Escena de Baldur's Gate

RPG tradicional:

Subgénero también conocido como RPG occidental, ya que Europa y Estados Unidos son los principales puntos de desarrollo y venta. Este subgénero es más purista que los demás: los juegos son una implementación directa de las reglas de algún juego de rol conocido con una o varias historias de fondo. Las batallas pueden ser por turnos o en tiempo real: este aspecto se suele dejar a juicio del jugador. Grandes exponentes de este subgénero son la saga Baldur´s Gate[r53], la saga Dragon Age[r54], la saga IceWind Dale[r55], la saga Neverwinter Nights[r56] y las dos primeras entregas de Fallout[r57].

Figura 20. Escena de Dragon Age: Origins

RPG de acción:

Este subgénero mantiene todos los elementos de su categoría, pero las facultades adquiridas durante el juego no son lo único: también se tiene en cuenta la habilidad del propio jugador. La principal característica de este subgénero es que las batallas y la exploración siempre tienen lugar en tiempo real. Algunos títulos representativos son Mass Effect[r58], The Elder Scrolls III: Morrowind[r59], The Elder Scrolls IV: Oblivion[r60], The Legend of Zelda[r61] y Fallout 3[r62].

Figura 21. Escena de The Legend of Zelda: Ocarina of Time

RPG japonés:

Subgénero de los juegos de rol caracterizado por ser extraordinariamente popular en Japón, aunque su distribución y consumo actual es a nivel mundial. Sus principales características son la llamativa estética y el particular diseño de los combates. Visualmente, los juegos suelen presentar muchos elementos característicos: robots humanoides, personajes dibujados al estilo manga/anime, temáticas orientales… Los combates se desarrollan siempre por turnos, en escenarios separados de la aventura principal, y el jugador tiene muy poco margen de acción sobre lo que sucede. En este apartado tienen cabida títulos como la saga Final Fantasy[r63], Star Ocean[r64], Lost Odyssey[r65] y Wild Arms[r66].

Figura 22. Escena de Final Fantasy X-2

RPG estratégico:

Subgénero que comparte varios elementos con los RPG japoneses: su popularidad en el país, su estética y el diseño de los combates. La diferencia más significativa es que, fuera de las batallas entre personajes, la acción suele ser bastante limitada: navegar por menús, adquirir equipo… Se puede pensar en ellos como juegos de rol en los que sólo cuentan las batallas. Algunos títulos emblemáticos son Front Mission[r67], Final Fantasy Tactics[r68] y Fire Emblem[r69].

Figura 23. Escena de Fire Emblem: Radiant Dawn

2.1.8. Aventura gráfica

Este género, uno de los más antiguos, tiene su origen en las clásicas aventuras conversacionales (Colossal Cave Adventure[r70]) alrededor del año 1975. El objetivo, como en la mayoría de los juegos, será avanzar en la historia narrada. En este caso, la mecánica básica de toda aventura gráfica es la de investigar y resolver pequeños acertijos utilizando diferentes acciones básicas (explorar, leer, empujar…). Aunque en los años 90 este género disfrutó de una enorme popularidad, en la actualidad está de capa caída, por lo que existen pocos títulos recientes del mismo.

Figura 24. Escena de The Longest Journey

Apuntar y hacer clic

Subgénero conocido también por su traducción inglesa: “Point & click”. Es la rama que más aspectos comparte con las aventuras conversacionales y, por tanto, la más purista dentro del género. La mecánica consiste en avanzar por la historia a base

de clics de ratón, rebuscando en cada rincón de cada escenario para coger objetos, utilizarlos, entregarlos, combinarlos… abriendo así nuevos capítulos. También se deberá interactuar con otros personajes, intentando escoger en cada momento la frase adecuada. Algunos buenos ejemplos son la saga Monkey Island[r71], la saga Simon The Sorcerer[r72] y The Longest Journey[r73].

Figura 25. Escena de The Secret of Monkey Island

Película interactiva:

Es la vertiente más reciente del género. El planteamiento es similar al subgénero de Apuntar y hacer clic, aunque la elección de acciones se ha visto muy mejorada. En este subgénero, las acciones se pueden elegir al situarse cerca de un objeto adecuado o de personajes activables. Adicionalmente, este subgénero suele estar caracterizado por una narrativa muy cinematográfica y gran cantidad de escenas de vídeo. Algunos títulos destacados son Grim Fandango[r74], Shenmue[r75], Fahrenheit[r76] y Heavy Rain[r77].

Figura 26. Escena de Grim Fandango

2.1.9. Puzzle

Género de videojuegos también conocido como "Juegos de habilidad". Los productos de este género, tan antiguo como el sector, están cada vez más de moda gracias al aumento del público casual y la proliferación de juegos diseñados para ser ejecutados en cualquier navegador Web. Uno de los pocos elementos comunes que tienen los juegos de este tipo es que el tiempo suele ser el principal enemigo, ya sea por una cuenta atrás que indica el final de la partida o por un sistema de puntuación basado en el tiempo. Títulos como Arkanoid[r78], Tetris[r79] (y sus clones), Zuma[r80], la saga Pang![r81], la saga Bust-A-Move[r82], o la saga Lemmings[r83] son desarrollos típicos de esta categoría.

Figura 27. Escena de Lemmings

2.1.10. Juego musical

Este género se basa en la inclusión de temas musicales como parte central del sistema de juego. El objetivo ronda siempre de alguna manera el ritmo o la letra de las canciones del estilo favorito del jugador, de manera que éste ha de cantar siguiendo la letra, tocar algún instrumento… La época dorada de este tipo de juegos tiene lugar a partir del año 2000. A pesar de tener su origen en los años 90, la necesidad de incluir dispositivos adicionales para jugar, así como la menor afición por la cultura del videojuego hicieron que su época dorada se haya visto retrasada entre cinco y diez años. Hoy en día, títulos como RockBand[r84], la saga Guitar Hero[r85], DJ Hero[r86], Singstar[r87], Beatmania[r88]… son conocidos por multitud de jugadores y no jugadores, llegando incluso a aparecer en programas de televisión o en prensa no especializada.

Figura 28. Escena de Guitar Hero Metallica

2.1.11. Juego de movimiento

Este género se caracteriza por la fuerte componente interactiva que introduce. Aunque algunos de los títulos podrían caer dentro del género puzzle, este tipo de juegos se centra en el movimiento corporal del jugador. Normalmente se usa algún periférico especial (cámara digital, alfombrilla, mandos…) en combinación con la plataforma de juego propiamente dicha. Los títulos más recientes incluyen Wii Fit[r89] y Wii Sports[r90], aunque Dance Dance Revolution[r91], Just Dance[r92] o Eye Toy Play[r93] tuvieron su momento de gloria con anterioridad.

Figura 29. Escena de Dance Dance Revolution: Hottest Party

2.1.12. Party game

Este género se traduce literalmente como "Juegos de fiesta", aunque no refleja el espíritu del término original. Una acepción más fiel podría ser "Juegos de reunión", entendiendo reunión como el hecho de que varios amigos pasen el rato en un ambiente distendido y divertido. Los títulos de este género se caracterizan por la variedad y la diversión. Su temática suele ser similar a la de los juegos de tablero: se debe avanzar hasta la última casilla superando determinadas pruebas y retos. Siempre permiten la participación de múltiples jugadores, por lo que es habitual influir en las partidas de tus compañeros/contrincantes. Algunos juegos que pertenecen a este género son la saga Mario Party[r94], Sonic Shuffle[r95] y Fuzion Frenzy[r96].

Figura 30. Escena de Mario Party 8

2.1.13. Juego de preguntas

Este género se caracteriza por retratar con la mayor fidelidad posible algún concurso televisivo de habilidad cognitiva: 50x15, Pasapalabra, Cifras y letras, Trivial… El jugador suele competir solo o contra otros jugadores con la única ayuda de su propio conocimiento. Este tipo de juegos se centra en pocos temas relacionados y suelen requerir algún dispositivo especial para pulsar al responder Este género adquirió más popularidad gracias a la exitosa saga Buzz[r97] de PlayStation 2. Algunos títulos adicionales son Scene It[r98] y Disney´s Think Fast[r99].

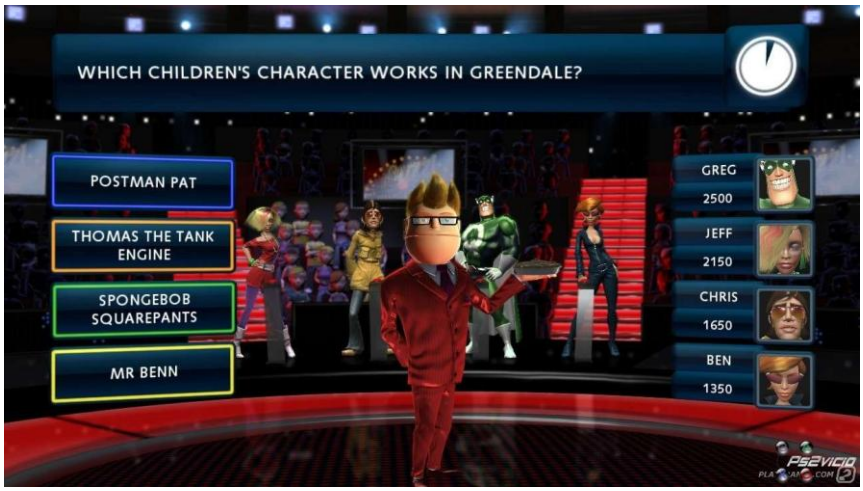

Figura 31. Escena de Buzz Quiz TV

2.1.14. Plataformas

Este género se caracteriza por su mecánica básica: saltar de plataforma en plataforma. Esta idea le da nombre e inspira la mayor parte de sus escenarios. El objetivo suele consistir en avanzar por los distintos niveles superando obstáculos, recogiendo objetos, eliminando enemigos… mediante la combinación ingeniosa de saltos y plataformas. Los títulos que se proponen para representar este género son la saga Mario Bros.[r100], la saga Prince of Persia[r101] y Mirror's Edge[r102], aunque el conjunto de los mismos es inmenso.

Figura 32. Escena de Super Mario Bros. 3

2.1.15. Training

Este género de juegos es, con mayor o menor variación, la versión digital de los acertijos y juegos lógicos vistos en el mundo real. El objetivo del jugador es resolver las pruebas planteadas, recibiendo una puntuación en base a los fallos cometidos, pistas pedidas, tiempo invertido… Dichas pruebas son de la más diversa índole: matemáticas, inglés, vocabulario, velocidad de reacción, capacidad de deducción…

Los desarrollos, por su proximidad con los juegos tradicionales en papel, han observado mayor evolución en las pequeñas consolas de mano o en dispositivos móviles. Algunos títulos destacables son la saga Brain Training[r103], Mind Quiz[r104], English Training[r105], Training For Your Eyes[r106] o Big Brain Academy[r107].

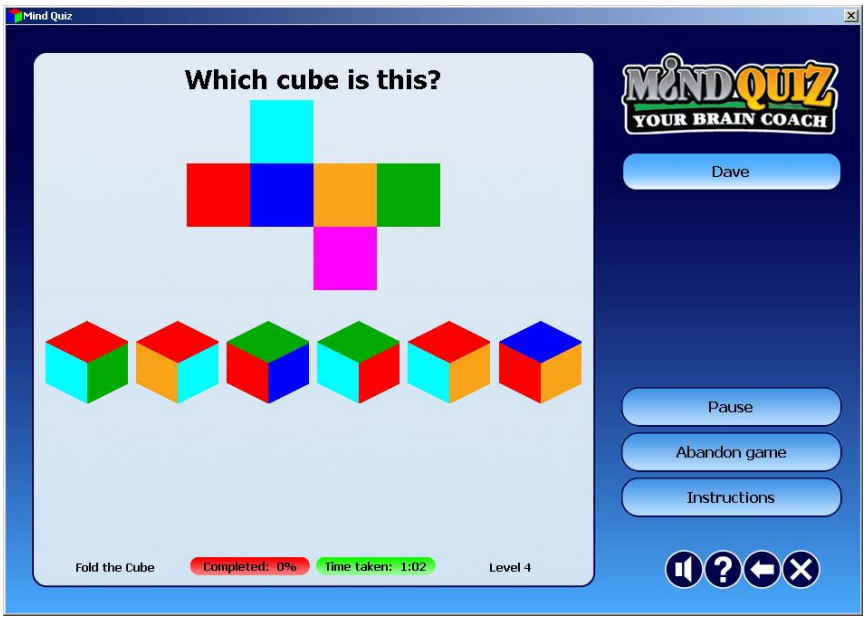

Figura 33. Escena de Mind Quiz

2.1.16. Sandbox

Esta categoría se traduce literalmente como “Caja de arena”, aunque se la conoce popularmente como “Juegos de mundo abierto”. Los títulos que caen en esta categoría son de muy diversa índole, por lo que en principio no pueden ser catalogados con el proceso utilizado hasta ahora. Sin embargo, la cada vez mayor cantidad de títulos que surgen y su gran éxito hacen que merezca la pena tenerlos en consideración. Las principales características que distinguen estos desarrollos son:

- La acción tiene lugar en un entorno más o menos extenso: una ciudad, un estado… Esto aumenta el campo de acción del jugador, otorgándole un cierto grado de libertad.
- Suele existir un vasto conjunto de tareas secundarias diseminadas por todo el terreno de juego. Estas aventuras, más o menos relacionadas con la historia principal, suelen dar variedad y ventajas al jugador interesado.
- Es común que el jugador tenga acceso al uso de vehículos. Esto es independiente de la historia de fondo, y es un hecho generalmente poco común en los videojuegos en general.

Siendo este género tan especial como se ha descrito, los títulos representativos se caracterizan por el otro género que trabajan. Así, se puede tener un Sandbox más orientado hacia RPG como la saga Fallout, o un Sandbox más orientado hacia FPS como la saga Far Cry[r108].

Figura 34. Escena de Far Cry 2

2.1.17. Conclusión

Aunque los juegos siguen siendo catalogados dentro de los géneros que se crearon hace décadas, en la actualidad la mayoría de títulos conviven entre diversas

fuentes y suelen abarcar varios géneros. Así, por ejemplo, se tiene que Bioshock[r109] estará catalogado como FPS, aunque tiene una importante condición de RPG y de Aventura.

Figura 35. Escena de Bioshock 2

2.2. Historia del videojuego de estrategia en tiempo real

Esta sección describe una parte de la industria del videojuego: aquella dedicada a los videojuegos de estrategia en tiempo real. Este tipo de videojuegos se caracteriza, en general, por permitir al jugador controlar un conjunto de elementos de juego (unidades, edificios y recursos) con total libertad y en tiempo real. El objetivo suele ser la eliminación de otros bandos similares en su composición y de facción enemiga.

El objetivo principal de esta sección es proporcionar la información necesaria para comprender qué es StarCraft, por qué tiene la implementación que tiene y, sobre todo, qué relevancia tienen estos aspectos en la elección del tema central de este Proyecto de Fin de Carrera.

Para cumplir el objetivo mencionado, la sección se ha estructurado como una crónica histórica del género: se parte de los desarrollos originales, meros esbozos de lo que sería un género, y se concluye destacando los juegos más actuales. Entre estos dos hitos se describen elementos clave de la historia, como pueden ser los aspectos novedosos, heredados o rescatados de algún juego o las claves del éxito de las creaciones destacadas.

2.2.1. Prólogo: orígenes del género

Para realizar una crónica del género de estrategia en tiempo real es necesario comprender la situación del mundo en que nació. A mediados de la década de 1980, Nintendo había apostado por una débil industria del videojuego con el lanzamiento de la Nintendo Entertainment System[r110]. Apple y Microsoft introdujeron en el mercado la noción de un ordenador personal. La tensión provocada por la Guerra Fría estaba comenzando a desaparecer y los países más ricos empezaban a pensar que un estilo de vida relajado era deseable. Los juegos más conocidos de la época, Tetris[r111]y PacMan[r112], fueron la base del ocio digital de esta generación. En el mundo del videojuego, todo era tan sencillo como emocionante.

Con lentitud, la complejidad fue ganando terreno. En 1992 vio la luz Wolfenstein 3D[r113], base del género First-Person Shooter. Los usuarios comenzaron a familiarizarse con la tecnología, haciendo de ella algo común. Algunos, maravillados con la compleja realidad modelada en SimCity[r114], darían vida al género de Simulación.

El lanzamiento de las consolas Super Nintendo Entertainment System y Sega Genesis fue la señal que esperaban los desarrolladores: la industria del videojuego definitivamente revivía, el público objetivo aumentaba y había una creciente sed de nuevas experiencias de juego.

Los primeros desarrollos de esta época tuvieron como base algo muy conocido y que daba lugar a multitud de tramas con relativa facilidad: la guerra. Algunos juegos utilizaron sus principios en general, mientras que otros la implicaban directamente en la experiencia de juego. En cualquier caso, un altísimo porcentaje de creaciones tuvieron la dinámica bélica como motivo principal. Algunas de estas creaciones intentaron trasladar los juegos de mesa o de tablero, como Risk[r115], a las plataformas digitales, dando lugar a toda una gama de fracasos. Se hizo patente la necesidad de un cambio de paradigma.

Sin embargo, este cambio no se produjo de la noche a la mañana. Durante algún tiempo, los juegos fueron simplemente derivados de otros, aportando gráficos diferentes o una historia novedosa e intrigante. Se analizarán los tres más significativos: Herzog Zwei[r116], Mega-Lo-Mania[r117]y Stonkers[r118]. Como se verá en apartados subsiguientes, estos desarrollos son la clave del éxito de Dune 2, por lo que se pueden considerar la raíz del género de estrategia en tiempo real.

Herzog Zwei

Juego desarrollado y publicado por TechnoSoft en 1989. El título, en alemán, puede ser traducido como “Duque Dos”. Es la continuación del juego Herzog[r119], que únicamente vio la luz en los mercados japoneses y que no será analizado en este documento. Fue concebido para la plataforma Sega Genesis.

El aspecto más novedoso de este juego, rescatado multitud de veces por creaciones posteriores, es el hecho de que el jugador es una unidad. Por supuesto, si hubiera una única unidad por jugador el género carecería de buena parte de los factores estratégicos. El jugador, entonces, es la unidad central: una nave nodriza capaz de "producir" otras unidades y enviarlas a realizar el cometido que el jugador necesite en cada momento.

Otro aspecto reseñable es la dinámica de juego, más propia de una creación arcade. El juego fue diseñado para ser manejado con los sencillos controles de Sega Genesis, por lo que los jugadores se ven inmersos en un constante ir y venir de órdenes y de movimientos: están constantemente pulsando botones. Esto contrasta claramente con la dinámica pensativa y pausada de otros juegos, sean de estrategia en tiempo real o, simplemente, de mesa.

Figura 36. Escena de Herzog Zwei

Mega-Lo-Mania

Juego desarrollado por Sensible Software y publicado por diversas compañías. Salió al mercado por primera vez en 1991. Aunque precede cronológicamente a otras creaciones importantes del mismo género, este juego fue creado originalmente para la plataforma Amiga, lo que le dio en su momento la ventaja competitiva en términos de calidad gráfica y de sonido.

Esta creación aporta multitud de elementos que, si bien no serían replicados en un futuro, sí expandieron las expectativas de los usuarios, creando rápidamente un nivel de calidad estándar. Los elementos son:

- Se introduce el concepto de historia. El juego estructura las partidas a lo largo de una línea temporal. Esta línea temporal se divide en épocas. Cada época se

caracteriza con elementos de la historia humana: edad de piedra, del bronce, del hierro…

- Se incorpora una gran cantidad de terrenos jugables. Cada época tiene un mapa dividido en sectores. Cada sector puede estar vacío u ocupado por las unidades de un jugador.
- Se introduce el concepto de construcción. Las unidades y los recursos de cualquier jugador son, simplemente, humanos. Siguiendo una lógica cíclica, los humanos pueden realizar acciones: construir edificios, construir herramientas, construir catapultas, atacar, recolectar minerales y reproducirse. Para realizar las acciones, el jugador deberá construir inicialmente las herramientas necesarias, preparando así a sus humanos. Cada jugador recibe una cantidad inicial de humanos y edificios.

Figura 37. Escena de Mega-Lo-Mania

Stonkers

Juego desarrollado y publicado por Imagine Software en 1993 para la plataforma 48K ZX Spectrum. En él se incorporan los elementos básicos de la estrategia en tiempo real: el uso de diferentes tipos de unidades y la acción continuada, sin turnos.

El aspecto más novedoso que aporta esta creación son las reglas del juego. Otros desarrollos siguen reglas fijas arbitrarias o funcionan al azar, mientras que Stonkers sigue el clásico sistema de “Piedra, papel y tijera”. En términos del juego, “Armor” (unidades similares a carros de combate) gana a “Artillery” (unidades similares a baterías antiaéreas), “Artillery” gana a “Infantry” (unidades similares a soldados rasos) e “Infantry” gana a “Armor”.

En otros aspectos, Stonkers no era tan novedoso, pero sí se oponía en cierta medida a lo que se esperaba de él. Esto le proporcionaría la notoriedad que, con el

paso de los años, le haría servir de inspiración para una nueva generación. Algunos de estos aspectos revolucionarios fueron:

- Se elimina la necesidad de adquirir y posicionar las unidades. Esto se realiza de manera aleatoria al inicio de cada partida.
- Se elimina la necesidad de construir los edificios. Cada jugador tiene un "Port" (puerto) y un "HQ" (centro de mando). No es posible construir más edificios ni recuperar aquellos que se hayan perdido.
- Se implementa únicamente un mapa. En él figuran un río, un puente y dos zonas de tierra, una por cada bando. Algunas zonas del mapa varían la velocidad de movimiento de las unidades. Esto hace destacar enormemente la importancia del factor estratégico en un juego de este incipiente género.
- Se propone una nueva idea respecto a los recursos. Cada jugador recibe cuatro cargamentos de recursos en su puerto a lo largo de la partida. Estos recursos sirven únicamente para mantener vivas las unidades que controla el jugador.
- Se introducen algunos combates obligatorios. Dos unidades comenzarán a pelear si los contendientes están próximos entre sí.

Figura 38. Escena de Stonkers

2.2.2. Época dorada: los años de gloria

Si Stonkers, Mega-Lo-Mania, y Herzog Zwei fueron el combustible para la imaginación de los primeros desarrolladores de estrategia en tiempo real, Dune 2 fue la chispa que encendió el fenómeno.

Estas cuatro creaciones, al definir los conceptos básicos del género, ofrecieron a los desarrolladores la plantilla necesaria para llevar a cabo sus propias ideas. La buena acogida que tuvieron convenció a las empresas de software, que no se contentaron simplemente con producir meras copias. En su lugar, tomaron la fórmula investigada

por Dune 2 y la expandieron mediante multitud de innovaciones. Al final de esta época, cuando se empezó a sentir la influencia de los primeros juegos en tres dimensiones, la apariencia inicial de Dune 2 había sido casi olvidada. Esto no quiere decir que los nuevos juegos no fueran tan buenos: de hecho, la mayoría eran mejores.

El género se vio sometido a una rápida serie de innovaciones. Algunos juegos enfatizaron un mayor realismo gráfico o histórico. Otros, como WarCraft[r120] y WarCraft 2, se especializaron en los combates cuerpo a cuerpo. La incipiente saga Command and Conquer, por ejemplo, sedujo a sus seguidores con creaciones modernas, tramas intrigantes y unos estupendos cortometrajes de altísima calidad que ilustraban el progreso del jugador. Parecía posible la existencia de un juego diferente adaptado a la imaginación de cada jugador.

Dune 2

Juego producido por Westwood Studios y publicado por Virgin Interactive en 1992. Considerado el primer juego del género, inspiró directa o indirectamente todas las creaciones posteriores, por lo que define la mayor parte de los elementos que componen un juego de género RTS. Mientras que desarrollos anteriores tendían a imitar aspectos concretos de la guerra, Westwood dio un paso adelante: en Dune 2 se tenía el control total del ejército.

Como desarrollo software, se trató de balancear la carga, utilizando algunos aspectos externos para poder centrar el esfuerzo en otros. Por ejemplo, se aprovechó el mundo de Dune[r121], una creación literaria de ciencia ficción de Frank Herbert[r122]. Esto evitó la necesidad de imitar la realidad y de depurar un mundo completo. Al mismo tiempo se consiguió un amplio espectro de posibilidades: tramas históricas, unidades, construcciones… Además, al existir varias facciones personalizadas, los jugadores podían sentirse más identificados con unas u otras.

Como se enunció, heredar el mundo de Dune hizo posible innovar en la forma de jugar. Herzog Zwei no permitía controlar a más de una unidad. Stonkers no permitía adquirir nuevas unidades. Mega-Lo-Mania no permitía controlar unidades de los otros sectores de un mapa. Todos estos factores se percibieron como restricciones hacia el usuario. Tirando la casa por la ventana, Dune 2 fue más allá: permitió el control total sobre las fuerzas del usuario. Las unidades pueden ser enviadas a cualquier parte del mapa. Pueden disparar a objetivos específicos. Pueden moverse y patrullar. Pueden ser construidas tantas como sea necesario. Es posible construir cualquier edificio en cualquier momento. Como resultado, todas las funcionalidades ya conocidas (recolección de recursos, envío de recursos a las tropas, investigación de nuevas tecnologías…) se vieron eclipsadas, pasando a ser simplemente complementarias.

A continuación se comentan algunos de los aspectos más novedosos o trabajados de Dune 2, ya que forman la base del resto de creaciones posteriores:

- Definición de facciones o bandos. Las unidades de los distintos bandos no son similares entre sí, sino que cada una tiene un nombre, una función, un gráfico asociado, un conjunto de sonidos… Esto favorece que los jugadores, dependiendo de su modo de juego, sean mejores con un bando u otro, aunque también implica más trabajo por parte de la compañía, que debe crear más contenido y balancear con igualdad el poder de cada bando.
- Creación de la necesidad de recolectar recursos a lo largo y ancho del mapa. Anteriormente, los recursos se generaban periódicamente en puntos específicos (Herzog Zwei), eran generados automáticamente (Mega-Lo-Mania) o, simplemente, no se generaban (Stonkers). En cualquier caso, esta nueva fórmula es simple y brillante: fuerza a los jugadores a salir de su base, les obliga a combatir por los recursos e introduce la estrategia económica dentro de las reglas del juego. Esta es la novedad más potente de Dune 2, y la idea se ha transmitido fielmente hasta la actualidad.
- Introducción del árbol tecnológico y del cambio dinámico de las reglas de juego. En vez de ajustarse a las simples unidades conocidas hasta entonces, se apuesta por un conjunto de unidades capaces de ser mejoradas por avances tecnológicos. Estos avances pueden ser leves, saltando entre ramas hermanas, o más graves y poderosos, ascendiendo por el tronco del árbol. Tácticamente, la exploración del árbol es un punto clave, ya que se modifican automáticamente las normas de juego. Por ejemplo, un jugador puede tener el control de la batalla gracias a sus tanques, y descubrir, de pronto, que el enemigo puede sabotearlos y controlarlos. Estas sorpresas son comunes durante el juego, y obligan a los jugadores a meditar sus acciones y a dividir los recursos de forma apropiada.
- Introducción de la "shroud" (“capa”). El mapa, inicialmente, está totalmente oscuro, siendo necesario explorarlo para conocer todos los elementos que incluye. Es la primera vez que se implementa el concepto, tan simple como novedoso, de que el jugador no tiene el conocimiento perfecto de las andanzas de su enemigo.
- Extinción de límites de construcción o producción. Siguiendo la historia narrada en el mundo de Dune, el jugador es capaz de construir cualquier edificio en cualquier lugar del mapa, excepto en los terrenos arenosos. De la misma manera, el jugador puede ordenar a cualquier construcción que produzca una unidad, sin más restricciones que el hecho de que cada unidad se produce solamente en un tipo de edificio concreto.

Cabe destacar que esta creación dista de ser perfecta. Aunque en su momento fue el último grito, la óptica actual señala ciertas deficiencias de interfaz: no es posible seleccionar varias unidades, tampoco se pueden dar varias órdenes en secuencia o alternativamente, o tampoco se puede indicar un itinerario concreto al mover una unidad. Con el paso del tiempo se han ido detectando y cubriendo estos aspectos en mayor o menor medida, convirtiéndolos en estándares de facto.

En cualquier caso, Dune 2 consiguió su objetivo, dar al jugador el control total de su ejército: por ello se convirtió en un éxito en su tiempo, y en un modelo a seguir en años por venir. Las siguientes creaciones incluirían otros trasfondos históricos o cambiarían algún concepto a su gusto, pero un alto porcentaje de aspectos serían simples adaptaciones del concepto explorado por esta creación.

Figura 39. Escena de Dune 2

Command and Conquer

Juego desarrollado por Westwood Studios y publicado por Virgin Interactive en 1995. Esta creación es la siguiente publicación del equipo de Dune 2, por lo que abundaron las referencias a ella como "Dune 3". Sin embargo, las únicas similitudes que comparten son las características propias del género. Westwood había establecido un estándar con su primer título y estaba mejorándolo con novedades y aspectos que hubieran quedado por debajo del nivel de calidad esperado.

En Command and Conquer, un meteorito impacta cerca del río Tíber, en Italia. Como uno de los resultados, surge el tiberio: un mineral altamente radiactivo, energéticamente inigualable y con tendencia a convertir sus proximidades en más tiberio. El jugador da comienzo a lo que se conoce como la Primera Guerra del Tiberio: la lucha por controlar el mineral, sus usos y sus zonas de actuación. Todo el juego está ambientado en la línea descrita con el mayor realismo posible: multitud de secuencias de vídeo, mapas, unidades y edificios idénticos a aquellos de la vida real…

Aparte de mantener el concepto de Dune 2, Command and Conquer introdujo un conjunto de ideas innovadoras:

- Se implementa la selección de grupos de unidades. Esto evita la secuencia "desplazar mapa, seleccionar unidad, desplazar mapa, dar orden" para cada

unidad, por lo que los jugadores pueden utilizar más tiempo para planear su táctica u observar los resultados.

- Se incluye la función de asignar grupos de unidades a números del teclado, pudiendo seleccionar estas unidades en cualquier momento de la partida. Es decir, el jugador es capaz de crear sus propias compañías, batallones… manteniendo el control en todo momento.
- Se acusa más el balance de unidades entre distintos bandos y dentro del mismo bando. Cada facción tiene diferentes unidades, cada una con una función específica. Las unidades de mayor armadura son más caras, y las más rápidas son más baratas. El juego obliga al jugador a encontrar el punto de equilibrio entre las unidades enemigas, las propias unidades y los recursos disponibles.
- Se implementa la interacción entre unidades. Algunas unidades pueden ser combinadas con otras para producir efectos diversos: una unidad más poderosa, o quizá más conveniente, o quizá capaz de esconderse temporalmente...

Figura 40. Escena de Command and Conquer

WarCraft II

Juego desarrollado y publicado por Blizzard Entertainment en 1995. Esta creación salió al mercado pocos meses después de Command and Conquer. Las compañías que respaldan ambos títulos se encontraban bajo la atenta mirada del público, que tendía a verlas como facciones rivales. En este sentido, cada una tenía la obligación de renovarse o morir: debían impactar adecuadamente en el mercado o retirarse de la competición.

La trama histórica que describe WarCraft II fue objeto de no pocas críticas en su momento. Aunque no fuera comparada a la trabajada historia de Command and Conquer, el argumento del juego carece de profundidad e implica poco al jugador. El jugador siente que está cumpliendo un camino lineal de misiones: hay dos diferentes para elegir, pero nada más.

Sin embargo, WarCraft II estuvo lejos de ser un fracaso: las figuras fantásticas de orcos, elfos y humanos capturaron la atención del público, que nunca antes las había visto en un juego. Además, el juego está enfocado al combate cuerpo a cuerpo, lo cual también era difícil de encontrar. Finalmente, este desarrollo cuenta con algunos aspectos innovadores que tendrían repercusión sobre todos los juegos venideros. A continuación se citan estas novedades:

- Se habilita el botón derecho del ratón para realizar la acción por defecto de la unidad seleccionada. Esto aumenta el tiempo de juego en detrimento del tiempo de manejo de interfaz, a la vez que evita a la compañía el tiempo necesario para poner las acciones en la interfaz.
- Se incluyen las batallas navales. Existen recursos navales, unidades navales y mapas basados en mares e islas. Para combinar estos aspectos con la fórmula de juego tradicional se tienen barcos transporte, barcos espía, barcos de ataque a otros barcos y barcos de ataque a tierra.
- Se implementa el juego para varios jugadores tanto en red local como en la novedosa red Internet. De esta manera comienzan los primeros torneos a gran escala, en los que jugadores de todo el mundo tienen su posición dentro de la comunidad y aquellos más premiados sientan su doctrina de juego de una forma casi fanática.
- Se hace posible la construcción lejos de la propia base. Por primera vez, el jugador puede construir sus edificios en cualquier punto del mapa sin necesidad de construir una base primero. Esto da lugar a cambios tácticos radicales: bases secundarias, cuarteles ocultos…
- Se implementa la "niebla de guerra": el terreno que no tiene unidades cerca se encuentra inactivo y oscurecido. Esto difiere de la "shroud" (la "capa") de Dune 2, en la que el mapa quedaba descubierto para siempre en cuanto era visitado. Combinado con la posibilidad de construir cualquier edificio en cualquier punto del mapa, lo que se trajo al mundo de la estrategia en tiempo real fue una total revolución del estilo de juego. Todas las creaciones futuras tendrán esta idea en cuenta de una manera u otra.
- Se implementa un editor de mapas. Este programa, incluido en todas las copias del juego, otorgó el control total del juego a los jugadores: ellos deciden a qué van a jugar. Los mapas más interesantes están en Internet, y cualquier grupo de usuarios puede elegir cómo hacer equipos o qué batalla jugar. También se hace posible recrear escenarios reales en términos del juego.
- Se incluye una forma de interactuar con el mapa en tiempo real. Cada facción tiene una unidad capaz de hacer explotar partes del mapa, haciendo posible la apertura de caminos secundarios. Las posibilidades que se abren entonces incluyen huidas, flanqueos, emboscadas…

Figura 41. Escena de WarCraft II

StarCraft

Juego desarrollado y publicado por Blizzard Entertainment en 1998. Después de comprobar el éxito de las creaciones previas, la apuesta era la siguiente: distanciarse de la competencia creando un título sencillo de aprender, difícil de perfeccionar y que fomentara la competencia y el juego en red. También se debían pulir los defectos observados en WarCraft II. Por tanto:

- Se diseña para ser fácilmente aprendido: los menús tienen opciones intuitivas, no hay apenas submenús y el árbol de tecnologías es reducido tanto en profundidad como en conexiones entre nodos.
- Se crean interfaces de conexión para todos los estándares de red del momento.
- Se incluyen tres facciones totalmente únicas. Acompañando a la historia que narra el juego existen tres razas. Cada una de ellas tiene sus unidades, construcción, manera de ser jugada, debilidades, ventajas…
- Se crea desde cero toda una trama histórica. Esta narración está plagada de vídeos, misiones de juego especiales, locuciones… Además, se implica mucho al jugador, ya que la campaña en solitario obliga a jugar con las tres razas disponibles.
- Se opta por una forma de protección contra copias que se sabía débil, permitiéndose deliberadamente el juego en solitario.
- Se controla fuertemente el juego en red. Se trabaja sobre la idea de que los jugadores conozcan el producto jugando en solitario, y luego deseen combatir contra otros jugadores. Además, se procura ofrecer novedades a través de Battle.net[r123]: mapas nuevos, objetos del juego, reconocimiento a los mejores jugadores…

- Se incluye una completísima herramienta de creación de mapas. Esta herramienta no sólo es capaz de crear mapas, sino que puede añadirles un conjunto arbitrario de hitos lógicos. Esto otorga el control del juego al usuario de una forma total: puede crear sus propios mapas con su propia historia, así como jugar según sus propias reglas. Más tarde, a través de la ingeniería inversa, sería posible modificar los recursos gráficos y de sonido del juego, haciendo modificaciones totales del mismo.

Hoy en día, 15 años después de su publicación, todos los objetivos que se plantearon originalmente se mantienen: es un testimonio de la brillantez del diseño realizado. StarCraft sigue teniendo una comunidad de jugadores amplia y activa. Especialmente notable es el caso de Corea del Sur: en este país, StarCraft es considerado deporte nacional[r124]. Existen jugadores profesionales, los cuales reciben el patrocinio de las grandes marcas del mundo tecnológico. También se televisan asiduamente concursos, eventos y competiciones relacionadas[r125].

Finalmente, el juego es conocido por un enorme conjunto de jugadores, sea por haber disfrutado del título, por referencias recibidas de otros jugadores... Teniendo en cuenta este hecho, no es de extrañar que Blizzard Entertainment decidiera continuar la saga con un segundo título. Esta nueva entrega se divide en tres módulos que, secuencialmente, describen una misma historia. Esto ha sido percibido como una buena manera de explorar en detalle una trama que se sabe interesante e intrigante. El impacto protagonizado por su salida al mercado se cuenta como uno de los hitos económicos más importantes del sector en 2010.

Figura 42. Escena de StarCraft

2.2.3. Las tres dimensiones: una revolución

Al término de la pasada etapa, los años de gloria del género tocaban a su fin. Los desarrolladores pronto se dieron cuenta de que necesitaban algo para tener a los aficionados ya no entretenidos, sino interesados. Necesitaban novedades como las traídas por Dune 2.

Justo en esta situación sale a la luz Total Annihilation[r126]. La compañía, Cavedog Entertainment, eligió confiar en su equipo de desarrollo y dar vida a un experimento: el primer juego de estrategia en tiempo real que utilizó escenarios y unidades en tres dimensiones y simuló las leyes físicas aplicables. Este movimiento fue muy osado, pues fue la primera apuesta comercial que puso en tela de juicio la capacidad de computación de los sistemas de la época: de haber salido mal, hubiera representado un problema muy serio para la compañía.

Respecto a la empresa que lo creó, Total Annihilation fue una creación más: ni un fiasco ni el juego más vendido. Para el mercado y el resto de creaciones, sin embargo, suponía una revolución. Sin embargo, tal acontecimiento no se produjo instantáneamente, ya que hubiera obligado a todos los equipos de desarrollo a cambiar de manera radical la forma de diseñar sus títulos actuales y futuros.

Poco a poco, otros equipos fueron tomando conciencia de lo que sería el futuro. Los primeros títulos tridimensionales que salieron al mercado aprovechaban la prueba conceptual que supuso Total Annihilation. Esto produjo un conjunto de títulos con gráficos excepcionalmente buenos pero vacíos en su interior. Las creaciones tuvieron su parte del mercado y fueron productivas en términos de beneficio, pero pronto cayeron en el olvido, reemplazadas por otras. En pocas palabras, Total Annihilation había sentado una nueva base, pero nada ni nadie había conseguido sacar provecho de ello.

A lo largo de los años, Relic Entertainment haría público Homeworld[r127] y su continuación, Homeworld 2[r128]. Sin ser totalmente revolucionarias, estas creaciones cuestionarían aspectos básicos del género. Por ejemplo, dejarían en evidencia la poca efectividad para conducir una trama argumental, ya que la mayor parte de los juegos tratan sobre la táctica de batalla y no sobre la historia anterior o posterior. También dejarían ver la dificultad de implementar un título desarrollado totalmente en tres dimensiones (espacio exterior), aunque saldría más que airoso de este problema.

Más que conducir el género hacía una larga y penosa agonía, esta crítica fue percibida por usuarios y desarrolladores como un soplo de aire fresco. Homeworld 2 será, por tanto, puerta de esta época hacia la siguiente, sentando la referencia gráfica y de jugabilidad para los desarrollos venideros.

Total Annihilation

Juego desarrollado por Cavedog Entertainment y publicado por GT Interactive en 1997. A pesar de coincidir en tiempo con otros desarrollos de la época, esta creación fue una arriesgada apuesta: es el primer título que calcula en tiempo real las tres dimensiones de las unidades, del terreno y de los proyectiles.

Como se puede suponer, el uso efectivo de las tres dimensiones supone un gran cambio en la experiencia de juego. Los gráficos son más coloridos, las explosiones son más creíbles, los disparos no siempre aciertan... Todo ello contribuye al novedoso realismo de Total Annihilation.

Así mismo, esta creación destaca por ser la prueba de concepto necesaria para poner en marcha el cambio en el sector. Con este juego quedó demostrado que los ordenadores del momento eran capaces de calcular y dibujar los movimientos de cientos de unidades y de miles de proyectiles a una velocidad razonable, o al menos suficiente para ofrecer al usuario la experiencia fluida que esperaba.

Este juego, sin embargo, también tiene su dosis de innovación en el aspecto teórico del género. Estas innovaciones serán luego rescatadas por diversos títulos futuros.

- Se potencia el apartado táctico del juego haciendo de los recursos algo renovable, infinito. Las únicas dudas del jugador son acerca del tiempo que tardan las unidades en llevar a cabo sus cometidos. Además, tampoco es necesario esperar a conseguir cierta cantidad de recursos sin hacer nada.
- Se introducen los recursos reciclables. Cualquier jugador puede aprovechar las ruinas de una unidad vencida para conseguir parte de los recursos que fueron necesarios para construirla.
- Se introduce la artillería de largo alcance. En todos los juegos anteriores, las unidades tienen un rango de ataque relativamente corto. Normalmente no pueden atacar más allá de lo que se muestra en pantalla. Total Annihilation aporta diversas unidades capaces de bombardear cualquier punto del mapa, abriendo multitud de posibilidades tácticas y cambiando sensiblemente la experiencia de juego.
- Se hace posible que las unidades se muevan mientras disparan. Además, las unidades intentarán maniobrar para evitar el fuego enemigo, y cesarán de disparar si existe riesgo de fuego amigo.
- Se rompe la arbitrariedad de las reglas. Se tiene una mecánicas de “piedra, papel y tijera”, pero también se utiliza la puntería, la suerte, el tipo de proyectil y de armadura, las maniobras de las unidades... como herramientas para variar el resultado de las batallas.

- Se potencia la parte para varios jugadores mediante la incorporación de más de 230 unidades en total. Esto aportaría variedad, haciendo del juego un clásico que siempre merece la pena volver a jugar.

Figura 43. Escena de Total Annihilation

Homeworld

Juego desarrollado por Relic Entertainment y publicado por Sierra Entertainment en 1999. Este título representa la segunda revolución del género: es la primera creación que utiliza en toda su profundidad las tres dimensiones del espacio.

En Homeworld se combina una trama argumental interesante con la libertad total de movimientos. La historia narrada ocurre en el espacio exterior y está ambientada de manera futurista. Trata sobre la raza Kushan, que realiza preparativos para encontrar su planeta natal; de pronto, es atacada por el imperio Taiidan, quedando vivo únicamente el jugador y sus tropas.

Como se ha mencionado, todo el juego se desarrolla en el espacio exterior: esto hace posible ciertas novedades tácticas como son la formación de batalla o la existencia de puntos ciegos. Aunque parezca difícil, la interfaz también fue especialmente diseñada para ser sencilla y para no obstruir la visibilidad de las escenas del juego. La parte gráfica de este título es uno de sus aspectos más importantes.

Otra de las novedades que incorpora Homeworld al género es la persistencia. Partiendo de la trama argumental, el jugador se encuentra obligado a llevar todo lo que no haya sido destruido o consumido a la batalla siguiente. Esto puede ser entendido como algo positivo por el factor de realismo que aporta, o bien puede terminar siendo aburrido, ya que el jugador se ve obligado a posponer el final de una batalla hasta asegurarse un remanente para la próxima.

Figura 44. Escena de Homeworld

Warcraft III: Reign of Chaos[r129]

Juego desarrollado y publicado por Blizzard Entertainment en 2002. La idea conceptual que llevó a realizar este desarrollo fue la intención de crear una nueva revolución en el género. El título sería bandera de un nuevo tipo de juego que vendría a llamarse RPS (Role-Playing Strategy, estrategia de juego de rol), basándose tanto en factores tradicionales de la estrategia en tiempo real como en ideas traídas desde el género de juegos de rol. Las novedades que propugnaría serían:

- La existencia de la figura central del héroe, una unidad muy poderosa.
- Los grupos de unidades no podrían moverse con total libertad; en su lugar, lo harían bajo el mando del héroe al que estuvieran asignados.
- La asignación de unidades a héroes se haría mediante edificios. Estos edificios estarían en el mapa sin intervención del jugador, y permitirían adquirir unidades a cambio de recursos.
- Los recursos serían diferentes para cada raza, siempre ajustados al trasfondo histórico que la raza describiera.
- Habría un total de seis razas totalmente diferentes entre sí.

Sin embargo, Blizzard Entertainment nunca se ha caracterizado por ser innovadora. Cierto es que ha sido capaz de coger lo mejor de un género, estudiarlo, y producir una obra de mayor calidad, pero nunca ha destacado por los aspectos novedosos. Este es el caso de, entre otros, WarCraft III. La compañía utilizó ampliamente los conocidos estándares del género.

Aun así, terminó habiendo espacio para la innovación: se implementa la figura del héroe como una unidad regular, aunque más poderosa que las demás. Estas nuevas unidades pueden ganar experiencia, subir de nivel y llevar objetos encima con

la finalidad de aumentar el daño que hacen, el daño que son capaces de recibir o, simplemente, para avanzar en la trama argumental del juego. De manera adicional, también se incluyeron edificaciones neutrales (tiendas de objetos, fuentes de vida, templos de resurrección...) para dar más juego a la figura del héroe. El resultado de esta innovación es que el jugador se involucra más, prefiriendo un bando u otro, o encontrando una forma de juego más entretenida que otra. Cada héroe tiene está totalmente definido: historia, voz, gráficos, actitud, ataques…

Otra novedad digna de mención está relacionada con los recursos. En este aspecto, Blizzard Entertainment da vida a los gastos asociados a la gestión de una economía. Dicho de otra manera, cuantas más unidades tiene el jugador, menos recursos es capaz de ingresar en cada intento. Por este motivo, las tácticas de expansión agresiva y sin control están desaconsejadas en WarCraft III, y es necesario buscar otro tipo de equilibrio.

Finalmente, es necesario hablar de dos aspectos novedosos respecto a la interfaz de usuario. En primer lugar, se implementó la selección de subgrupos dentro de los grupos de unidades. De esta manera se agiliza la gestión al más bajo nivel, haciendo posible dar órdenes sólo a las unidades cuerpo a cuerpo, o sólo a las unidades destinadas a curar. En segundo lugar, también se implementó la posibilidad de hacer algunas habilidades automáticas. Así, por ejemplo, las unidades destinadas a curar lo harían tan pronto como observaran alguna unidad amiga dañada.

Figura 45. Escena de WarCraft III

Homeworld 2

Juego desarrollado por Relic Entertainment y publicado por Sierra Entertainment en 2003. Cuando esta creación fue anunciada, muchos de los seguidores de Homeworld se preguntaron qué podía aportar Relic para mejorar su pasado título, aún

reciente en las mentes de los jugadores. La compañía centró sus esfuerzos en el área gráfica y en la interfaz de usuario.

Tras cerca de cuatro años de desarrollo, el apartado visual recibió una considerable cantidad de tiempo, por lo que se aprecian mejoras en este sentido: mayor realismo, mayor detalle, efectos especiales más definidos... El movimiento de la cámara se mejoró también, evitando quedar excesivamente lejos de la acción o tener alguna unidad que bloqueara la visión del jugador.

Respecto al apartado de interfaz, se habían detectado defectos que fueron solucionados en esta nueva versión. Por ejemplo, en Homeworld era difícil localizar y controlar a una única unidad, cuestión que se solucionó gracias a un ingenioso diseño centrado en la escuadrilla como unidad de gestión. Además, se tomó prestado parte del aspecto de Emperor: Battle for Dune[r130], siendo la interfaz transparente y ocultable.

Aunque el juego en sí es un gran título que destaca entre sus competidores, también es cierto que las innovaciones brillan por su casi total ausencia. Si acaso, conviene reseñar que Homeworld 2 incorpora el concepto de módulos y subsistemas: piezas producidas por el jugador para mejorar ciertas unidades. Si bien este enfoque nunca se había implementado, las ideas originarias tienen más de diez años de antigüedad (Dune 2, Command and Conquer).

Figura 46. Escena de Homeworld 2

2.2.4. Época de madurez: una nueva generación

Al concluir el período iniciado por Total Annihilation, el mundo de la estrategia en tiempo real había quedado muy bien definido. Los cambios descritos en la pasada época, con la salvedad de las novedosas tres dimensiones, son de calado menor.

Además, los jugadores han dedicado mucho tiempo a disfrutar de sus títulos preferidos, por lo que saben con precisión qué esperan ver en un videojuego de género RTS. De esta manera, los desarrolladores pueden enfocar toda su creatividad en dar vida a un conjunto de aspectos relativamente pequeño en lugar de tratar de satisfacer una amplia variedad de gustos.

En esta época comienzan a despuntar los primeros juegos RTT (Real-Time Tactics o Táctica en Tiempo Real). Este tipo de creaciones son juegos de género RTS sin aspectos económicos o políticos: se centran exclusivamente en la táctica de batalla de pocas unidades. Este subgénero, si bien no es excesivamente popular, sí ha influido a otras creaciones, enfocándose todas ellas hacia las batallas de pequeña escala. Esto contrasta con la filosofía inspirada por Total Annihilation: según se observa, este gran título solo pudo transmitir la necesidad de las tres dimensiones.

El enfoque de acciones a pequeña escala ha tenido un gran punto positivo: el aumento de la cifra de ventas. En esta época, el género RTS destaca por su aumento de la cifra de beneficios. El motivo principal de ello es la publicidad: en The Lord of the Rings: Battle for Middle Earth[r131] se recrea el mundo de Tolkien para la plena satisfacción de los aficionados a esta literatura; en Rome: Total War[r132] el usuario puede observar grandes batallas en detalle mientras mantiene una consistencia histórica envidiable; Command and Conquer: Generals[r133] muestra la guerra actual y futura con realismo y detalle… En definitiva, el género de videojuegos RTS observó el potencial del uso de licencias comerciales. A partir de ahí se diversificó, dando a un reducido público objetivo el gusto de tener un juego diseñado prácticamente a su medida.

Por otro lado, el nivel visual de los juegos se ve incrementado hasta ser casi idéntico al de las fotografías. Age of Empires III[r134], por ejemplo, prueba los límites de lo comercialmente computable, ofreciendo así una calidad gráfica sin igual entre los de su género. Y, como sabe cualquier gran compañía, si es más bonito, vende más.

A lo largo de esta época se hace evidente que, sin el respaldo de una gran entidad, los proyectos están condenados a fracasar ante la competencia. Eso sí, aquellos desarrollos financieramente adecuados tendrán la calidad que sólo una enorme cantidad de dinero, de talento y de tiempo pueden conseguir.

Age of Empires III

Juego desarrollado por Ensemble Studios y publicado por Microsoft en 2005. Esta creación es la cuarta de la saga “Age of”, proyectos de Ensemble y Microsoft desde la primera entrega. Está ambientada en el periodo colonial del Nuevo Mundo (América) entre los años 1500 y 1850. El jugador debe seleccionar una de las ocho civilizaciones implementadas, conquistar el nuevo continente y reducir o aliarse con todo enemigo existente.

En general, los aspectos de Age of Empires III son muy similares a los de otros juegos del género, sean pasados o coetáneos. Cada civilización tiene sus ventajas e inconvenientes; aparte de esto, comparten las mismas unidades conceptualmente hablando. Las reglas que dominan el juego son simples: piedra, papel y tijera. O, lo que es lo mismo, caballería, infantería y artillería. Se deben conservar las fuentes de recursos del mapa: comida, madera y monedas. Estos conceptos cubren las necesidades económicas del jugador y deben ser entregados a la Ciudad Base para poder ser utilizados. Existe un árbol tecnológico que da acceso a diferentes estructuras y unidades. Se avanza a través de él por medio de las Edades, que dan nombre a la franquicia ("Age" es el término inglés para "Edad").

Sin embargo, existen otros aspectos que lo hacen único por sí mismo, así como representativo de la época en la que encaja. En primer lugar, el juego sigue la tradición de la franquicia, enfocándose al realismo más preciso. Es por este motivo que incorpora el motor gráfico Havok[r135]: la trayectoria de cada disparo de cada unidad es calculada y animada en tiempo real. De manera similar, si alguna unidad cae o es destruida, los efectos de este suceso podrán verse en toda su magnitud.

Adicionalmente, este título implementa tres aspectos muy bien entrelazados tanto entre sí como con la trama histórica que desarrolla. Estas novedades son las siguientes:

- Se utiliza la Experiencia como recurso. Se obtiene mediante la construcción de elementos propios o la destrucción de elementos de otros jugadores. Sirve para subir de nivel la Ciudad Base del propio jugador.
- Se implementa la Ciudad Base como elemento clave de juego. En ella se investigan las mejoras más importantes, se permite el cambio de Edad y representa conceptualmente el primer asentamiento colonial en el nuevo continente.
- Se incorpora el uso de cartas. Cada jugador recibe estos objetos a lo largo de la partida, con un límite de veinte unidades. La utilidad reside en poder solicitar, mediante el gasto de una carta, el envío de refuerzos militares, económicos o mejoras diversas. Esto representa el enlace de comunicación existente entre el asentamiento colonial y el Viejo Continente.

Figura 47. Escena de Age of Empires III

The Lord of the Rings: The Battle for Middle Earth II

Juego desarrollado y publicado por Electronic Arts en 2006. Este juego trata de recrear la ambientación, personajes y dinámica de la creación de J.R.R. Tolkien, El Señor de los Anillos[r136]. El jugador tiene seis facciones a su disposición, y su tarea es la de conquistar la Tierra Media mediante la destrucción de las restantes facciones.

El título como tal, según se analizará, goza de una gran calidad gráfica y de un amplio abanico de posibilidades y modos de juego. Sin embargo, el motivo de ser expuesto en esta sección es otro: tiene su base en la licencia comercial del mundo de Tolkien. Buena parte del éxito que tiene este producto se fundamenta en este hecho, ya que reúne tanto a los aficionados al género RTS como a aquellos más adeptos a la fantasía que se describe.

Como refuerzo al argumento anterior se ha elaborado una lista de características de esta creación, refiriendo cuando sea posible de qué otro título recibieron inspiración.

- El combate se basa en los principios básicos de “piedra, papel y tijera”, aunque se complica con seis elementos en juego en lugar de tres.
- El recurso llamado “recurso” se obtiene automáticamente al construir el edificio asociado con esta funcionalidad. Esta idea se observa por primera vez en Mega-Lo-Mania, previamente documentado.
- El concepto de “puntos de mando” como limitación al número de unidades controladas data de Dune 2, el primer juego del género.

- Los edificios neutrales, cuya utilidad es adquirir diferentes unidades como ventaja competitiva frente a los demás jugadores, ya se había visto en WarCraft III.
- Los puntos de poder, generados mediante la victoria en batalla y que sirve para activar habilidades especiales de las unidades sigue la implementación propuesta por Command and Conquer: Generals.
- El juego permite jugar a gran escala, decidiendo las batallas automáticamente. Esta funcionalidad es muy similar a la que implementa Shogun: Total War[r137].

Figura 48. Escena de The Lord of the Rings: The Battle for Middle Earth II

2.2.5. Epílogo: conclusiones y tendencias

Tiempo atrás, en una época oscura y casi olvidada, los juegos eran construidos por pequeños equipos de gente creativa. Unos pocos, incluso una sola persona, podían dar vida a la visión del entretenimiento que compartían. Pero los tiempos han cambiado. Ya no hay lugar para los programadores-diseñadores, los desarrolladores-publicistas ni las creaciones que nacen y crecen dentro de las cuatro paredes de una pequeña oficina. En este proceso de cambio también se ha perdido el espíritu inicial.

Quizá consuele a algunos aficionados nostálgicos que los primeros juegos fueran invariablemente malos. Pero nadie de su generación es capaz de no estremecerse por la inevitable pérdida de la genuina idea original. Con el largo camino recorrido se han conseguido increíbles gráficos, una complejidad suprema, una especialización inigualable… pero se ha perdido algo que debió haberse conservado.

Pero la historia del género está plagada de “préstamos” de ideas. Si una idea puede ser tomada de un juego viejo e incorporada en uno nuevo, ¿no podría hacerse lo mismo con el espíritu original del género? Probablemente no. Esto se debe a que no sólo el género ha cambiado, sino que también los jugadores han evolucionado. Son más exigentes, más competitivos y se encuentran más cómodos dentro de la especialización de una franquicia comercial concreta. Como resultado, las creaciones

RTS demandan más recursos que nunca. Y la pregunta que todos los desarrolladores se hacen es ¿qué es lo siguiente que se puede hacer?

Dios de "la Caja"

Tradicionalmente, los usuarios del género RTS habían encontrado un elemento común, "la caja": una idea claramente delimitada que recuerda la concepción de mapa subyacente. Sin embargo, para algunos, el siguiente paso en la evolución es simple: Massively Multiplayer Online RTS (MMORTS, Juegos de Estrategia en Tiempo Real Online Masivamente Multijugador).

El género MMORTS sería abanderado por las nuevas generaciones de aficionados: sin duda es la perfecta antítesis del espíritu original. Los jugadores que conocieron los primeros juegos RTS defienden que no importa la escala para la que se diseñe el juego, sino la sensación de control que emana de él. Al trasladar el número de jugadores a un tamaño masivo, el producto final es incapaz de hacer sentir que se está dirigiendo un ejército, ya que las acciones individuales tienen menor peso.

Aunque la idea de crear un videojuego MMORTS parece increíblemente compleja, necesitada de una ingente cantidad de recursos y que, además, obligaría a los jugadores a especializarse, no parece disminuir como sería lógico a lo largo del tiempo. La posibilidad de librar batallas a lo largo y ancho del planeta o incluso entre varios planetas tiene un enorme atractivo para muchos aficionados.

Éxito del subgénero RTT

Aunque el género se encuentra en una edad madura, esto no quiere decir que se haya estancado. Nuevas e ingeniosas creaciones son incorporadas continuamente. Muchas de ellas son de menor calado, aunque otras han llegado a transformarse en subgénero por sí mismas.

Este es el caso, introducido anteriormente, de RTT. Los jugadores evitan los rutinarios aspectos económicos del género y se dejan llevar por la acción táctica de batalla, más directa. Quizá esto es otro camino para las creaciones que están por venir.

Partida y regreso

Quizá la ingente especialización del género acabe dando lugar a interfaces creativas y accesibles. Y, quizá, esto permita encontrar un lugar cómodo en el mercado de las consolas.

Aunque grandes creaciones como Command & Conquer o StarCraft fueron recodificadas para diversas consolas, pronto se vio que el género se había hecho

demasiado complicado para ser manejado mediante una interfaz tan simple como puede ser el mando de una consola. Se dieron muchos intentos de transportar creaciones exitosas de PC, pero ninguna cosechó más que un modesto triunfo.

La tendencia se mantuvo hasta que Electronic Arts apostó por el sistema XBOX 360 con el lanzamiento de Command and Conquer 3[r138]. Aunque no fue el juego más vendido, sí demostró que se podía manejar una creación RTS actual con relativa eficiencia. Este hecho fue determinante para que las comapñías Bungie y Ensemble Studios publicaran Halo Wars[r139].

Es de destacar la ironía que subyace dentro de la historia del género. Habiendo nacido precisamente en el mundo de las consolas, pronto fue confinado a los límites del teclado y del ratón de un ordenador. Al final, varias creaciones volvieron al mundo de las consolas, obligando a los desarrolladores a pensar si los supuestos básicos que se han mantenido durante años deben sobrevivir mucho más tiempo o no.

Figura 49. Escena de Command & Conquer 3

Más allá de la guerra

La pregunta que debe ser contestada algún día se mantiene realmente abstracta. ¿Es necesaria la complejidad para dar profundidad? El realismo, ¿añade o quita profundidad a los juegos? ¿Cuáles son los límites del género? ¿Deben ser cruzados o preservados?

Para responder a estas y otras preguntas similares se podrían investigar creaciones menos tradicionales del género. En 2001, Nintendo hizo público su título Pikmin[r140] para la consola GameCube. Esta creación incorpora estrategia y táctica, pero es considerado mayoritariamente como no RTS. La razón es simple: no hay batallas, ni armas, ni conquistas, ni muertes… El juego trata sobre las criaturas pikmin, que son capaces de crecer, cosechar, curar a otros individuos o defenderse de sus enemigos. Esto abre un conjunto de interrogantes: ¿por qué los RTS deben tratar temas bélicos? ¿Por qué no es posible crear un RTS lleno de divertidas criaturas coloridas? ¿Por qué todos los héroes son guerreros?

Puede ser que los desarrolladores hayan tenido muy estrechas sus miras durante las pasadas décadas. O quizá el género ha estado aprisionado desde el principio por el realismo forzoso de la guerra. En teoría, para un género que se basa en otorgar el control total al jugador, las posibilidades deberían ser infinitas.

Figura 50. Escena de Pikmin

Nuevos horizontes

Durante las pasadas décadas, el crecimiento en popularidad y diversidad del género RTS ha sido enorme. Lo que comenzó con unos pocos programas simples inspirados en juegos de mesa tiene ahora reconocimiento internacional, torneos, diversidad… Conforme las condiciones iniciales fueron evolucionando, la comunidad de jugadores también cambió. Nadie sabe con seguridad hacia dónde avanza este género de videojuegos, si hacia el éxito o hacia el colapso; sea lo que fuere, el tiempo lo dirá.

2.3. El videojuego StarCraft

Esta sección se dedica al análisis de todos los aspectos que rodean a StarCraft, videojuego de estrategia en tiempo real creado por Blizzard Entertainment, base de desarrollo de este proyecto y tema en torno al que gira la mayor parte de los aspectos trabajados.

StarCraft es el nombre que tienen como base todos los productos de la saga; el videojuego analizado aquí es el primer elemento, por lo que es considerado el origen, sin ningún añadido, complemento o precedente. El nombre, inglés, se compone de "star": estrella, astro, cuerpo interestelar… y de "craft": embarcación, nave, creación, el arte de/al realizar determinada actividad… La traducción literal, por tanto, es "Nave estelar" o "Embarcación estelar", aunque también puede traducirse como "Arte estelar" o "Creación estelar". Este juego de palabras sobre "Craft" se introdujo a propósito para otorgar al título un pensamiento similar a "El arte de construir embarcaciones estelares".

A continuación se ofrece una reseña de la compañía creadora de la saga y su evolución en el tiempo, para luego explorar el juego en profundidad. Finalmente, se ofrece la situación actual del juego en el mundo, incluyendo aspectos como torneos, conferencias, expansiones, modificaciones o continuaciones.

2.3.1. La compañía

Activision Blizzard es una empresa estadounidense con sede en Irvine, California, dedicada a la creación de videojuegos, y la compañía creadora de StarCraft. Su fecha de fundación data de 1991 bajo el nombre de Silicon & Synapse. Más tarde, en 1994, la empresa es comprada por la distribuidora Davidson & Associates, que cambiarían el nombre de la empresa por Blizzard Entertainment Inc. En 1998, Blizzard Entertainment Inc. fue vendida a Havas[r141] por problemas de fraude fiscal. Ese mismo año, Havas traspasaría la totalidad de la compañía a Vivendi[r142]. En 2007, Vivendi y Activision se fusionaron, rebautizando a la empresa con el nombre que actualmente ostenta: Activision-Blizzard.

Figura 51. Logotipos actuales de Blizzard Entertainment

Esta empresa se caracteriza por el nivel de calidad de sus productos: prefiere retrasar la fecha de lanzamiento de sus creaciones hasta que alcanzan el nivel deseado, asegurándose así expectación, ventas y una buena experiencia de juego. De acuerdo a esta política de calidad, por ejemplo, se cancelaron algunos desarrollos (WarCraft Adventures: Lord of the Clans[r143], StarCraft: Ghost[r144]) o se ofrece la posibilidad de realizar reserva de lanzamientos próximos. Finalmente, la empresa también administra un popular servicio gratuito de juego a través de internet llamado Battle.net. Este servicio está disponible hoy en día para cualquier copia original de cualquier juego de la compañía que tenga implementada la conexión a este sistema.

A pesar de la mencionada cancelación de StarCraft: Ghost, proyectada continuación de StarCraft, Blizzard Entertainment Inc. hizo público un comunicado en 2007 en el que se aseguraba el desarrollo de StarCraft II. Después de casi diez años, para sorpresa de gran parte de la comunidad, se decidieron a confirmar la continuación de la saga. Este desarrollo conllevaría, aseguraron, mejoras a nivel gráfico y de jugabilidad, así como la continuación de la historia narrada a lo largo de StarCraft. Las promesas se hicieron realidad en 2010 y, con muchísima expectación, largas colas y una enorme cantidad de personas en listas de reserva, StarCraft II: Wings of Liberty vio la luz.

A continuación se enumeran algunas de las creaciones propias de la empresa. Estos datos se muestran cronológicamente ordenados y relacionados con la taxonomía realizada.

- The Lost Vikings (1992): juego de plataformas.
- Rock N' Roll Racing (1993): juego de conducción.
- Blackthorne (1994): juego de plataformas.
- The Death and Return of Superman (1994): juego beat 'em up de tipo lucha con progreso.
- Warcraft: Orcs & Humans (1994): juego de estrategia bélica en tiempo real.
- Justice League Task Force (1995): juego de lucha uno contra uno.
- The Lost Vikings II (1995): juego de plataformas.
- Warcraft II: Tides of Darkness (1995): juego de estrategia bélica en tiempo real.
- Warcraft II: Beyond the Dark Portal (1996): juego de estrategia bélica en tiempo real.
- Diablo (1996): juego de rol de acción.
- StarCraft (1998): juego de estrategia bélica en tiempo real.
- StarCraft: Brood War[r145] (1998): juego de estrategia bélica en tiempo real.
- Warcraft II: Battle.net Edition (1999): juego de estrategia bélica en tiempo real.
- Diablo II (2000): juego de rol de acción.
- Diablo II: Lord of Destruction (2001): juego de rol de acción.
- Warcraft III: Reign of Chaos (2002): juego de estrategia bélica en tiempo real.
- Warcraft III: The Frozen Throne (2003): juego de estrategia bélica en tiempo real.

- World of Warcraft (2004): juego online de rol de acción.
- World of Warcraft: The Burning Crusade (2007): juego online de rol de acción.
- World of Warcraft: Wrath of the Lich King (2008): juego online de rol de acción.
- StarCraft II: Wings of Liberty (2010): juego de estrategia en tiempo real.
- World of Warcraft: Cataclysm (2010): juego online de rol de acción.
- Diablo III (en desarrollo): juego de rol de acción.

2.3.2. StarCraft en detalle

En esta sección se describen en profundidad los aspectos propios del videojuego en que se fundamenta el proyecto. El propósito es dar el debido fondo al lector para que el diseño realizado no le sea extraño en ningún aspecto.

En este punto, por tanto, se tratan aspectos generales como año de creación, formato de distribución, cifra de ventas, secuelas y expansiones… Después se estudian aspectos concretos: los modos de juego, los recursos y tipos de terreno, las razas disponibles y, por último, la historia narrada durante el juego.

Aspectos generales

StarCraft es un videojuego de género RTS creado y publicado por Blizzard Entertainment Inc. en 1997. Su salida al mercado se produjo en 1998 para ordenadores con el sistema Windows en la forma de un novedoso Compact Disc (CD). Un año más tarde saldría a la venta otro CD para Mac OS y, en junio de 2000, tuvo una adaptación en formato cartucho al sistema Nintendo 64 con la participación de Mass Media Interactive Entertainment. En el momento de escritura de este documento la cifra de ventas se sitúa en unos 10 millones de copias, aunque todavía continúa a la venta en tiendas y se espera un aumento debido a la comercialización de su expansión, StarCraft II.

Otros hitos importantes relación a los productos que engloba la saga son:

- En 1998 se puso a la venta la única expansión oficial, StarCraft: Brood Wars.
- Desde 1998 hasta hoy ha recibido cerca de 20 actualizaciones en forma de parches. Esta forma de desarrollo es muy característica de la compañía. El contenido de dichas actualizaciones ha sido diverso: eliminación de defectos gráficos, incorporación de nuevas animaciones, modificación del equilibrio entre razas, mejora de la inteligencia enemiga, nuevos mapas…
- En 2007 Blizzard Entertainment Inc. anunció la secuela oficial del juego: StarCraft II. Este producto representa la continuación de la historia de StarCraft, ya que mantiene las mismas razas protagonistas y retoma el argumento de la primera entrega del juego. Este segundo título está basado en

tres bloques secuenciales (Wings of Liberty para la historia Terran, Heart of the Swarm para la historia Zerg y Legacy of the Void para la historia Protoss) y en varios parches por bloque.

El juego se encuentra en el ambiente fantástico y futurista del siglo XXVI. En él, tres razas provenientes de distintos planetas luchan por la supervivencia en algún lugar de la familiar Vía Láctea. Dichas razas se verán en detalle más adelante.

Modos de juego

En este apartado se describen los diversos modos de juego implementados en StarCraft. Algunos de ellos fueron incluidos originalmente con el lanzamiento del producto, mientras que otros han sido popularizados posteriormente por la comunidad de jugadores.

Campaña: modo jugable que incluye los episodios del juego con sus mapas y misiones individuales. Cada episodio trata sobre una raza y está compuesto por 10 misiones de diverso ámbito (explorar el mapa, construir o destruir una estructura, crear o eliminar una unidad, resistir un asedio…).

Escaramuza: modo jugable que permite combatir contra uno o varios bandos controlados por el ordenador en el mapa deseado. Los bandos pueden ser de cualquier raza. La inteligencia que presenta el ordenador es considerada débil para jugadores expertos, pero puede ser utilizada para aprender aspectos más básicos.

Multijugador: modo jugable que permite combatir contra otros jugadores humanos. Los medios implementados para ello son:

- Internet (a través de Battle.net): por medio de este modo se puede jugar contra cualquier jugador humano que participe en Battle.net. Éste es un servicio gratuito que requiere el número de serie original del juego. La compañía ofrece, adicionalmente, un simple sistema de charla, la posibilidad de crear equipos, un sistema de puntuación y una liga.
- Red local por IPX o UDP: estas dos tecnologías pueden utilizarse para interconectar varios clientes StarCraft conocidos entre sí. Todos los jugadores deben tener instalada su copia del juego y conocer la IP y puerto de destino.
- Módem y conexión directa por cable: estas tecnologías, ya en desuso, fueron las primeras en ser implementadas. Permiten hasta cuatro jugadores, siempre que éstos tengan instalada su copia del juego y sean capaces de configurar la red adecuadamente.

Reglas del juego: modo no jugable complementario para los modos de juego “escaramuza” y “multijugador”. Este modo permite escoger las reglas según las que se desarrollará la partida. Algunas opciones son:

- Arriba contra abajo: la pantalla se divide en dos partes según la cantidad de jugadores.
- Refriega: se permite combinar la aparición de contrincantes humanos y por ordenador.
- Todos contra todos: se inhibe la creación explícita de alianzas.
- Duelo entre dos: este modo permite enfrentarse a dos jugadores hasta que uno pierda toda capacidad de supervivencia.
- Muerte súbita: el jugador gana si mata a una criatura enemiga. Este modo suele activarse cuando se termina el límite de tiempo.
- Usar ajustes del mapa: cada mapa está diseñado para uno o varios modos de juego.
- Tomar la bandera: el jugador ganará si captura un objeto que está en posesión del bando enemigo.
- Avaricia: el ganador de este tipo de partidas es aquel que consiga acumular una determinada cantidad de recursos o controlar todos los generadores de recursos del mapa.
- Matanza: la condición de victoria de este tipo de partidas es que no quede ningún elemento enemigo vivo.

Finalmente, las reglas del juego analizadas se pueden combinar libremente con las siguientes opciones:

- Alianza: dos bandos deciden dinámicamente crear un pacto de no agresión. La interfaz del programa impedirá comportamientos fuera del pacto mientras esté vigente.
- Equipos: dos bandos se configuran estáticamente y para toda la partida como aliados.
- Victoria aliada: el jugador gana si algún aliado gana.
- Victoria por equipos: el jugador gana si algún miembro del equipo gana.
- Límite de tiempo: la partida tendrá una duración concreta. A partir de dicho punto se podrá bien puntuar a los contendientes o bien declarar muerte súbita.

Editor de mapas: modo no jugable. El juego incluye un editor de mapas versátil que ha permitido la creación de mapas con un estilo completamente distinto al del juego original. Por ejemplo:

- Recreación de eventos históricos: estos mapas recrean escenarios reales como son las guerras mundiales, napoleónicas, la Edad Media…
- Mapas de ficción: en estos mapas se recrean otros juegos o sagas populares como son El Señor de los Anillos, Matrix…
- Mapas tipo “Defensa”: el argumento de este tipo de mapas es el asedio de una posición concreta.
- Mapas de diplomacia: estos mapas tienen su base en algún lugar geográfico real, aunque pueden contener elementos de ficción. El argumento es la

administración de una nación: en especial, de sus alianzas, pactos y guerras. Se puede conquistar territorio neutral o enemigo, pero el gran atractivo es la dinámica social: no se puede tomar una decisión sin recibir repercusiones. Por ejemplo, si se decide atacar a un jugador, los aliados del mismo le deberán ayudar, mientras que otras naciones tratarán de conseguir ventajas económicas a expensas de la guerra. Un ejemplo muy conocido es "Blood & Iron", ambientado en la Europa de los años 70.

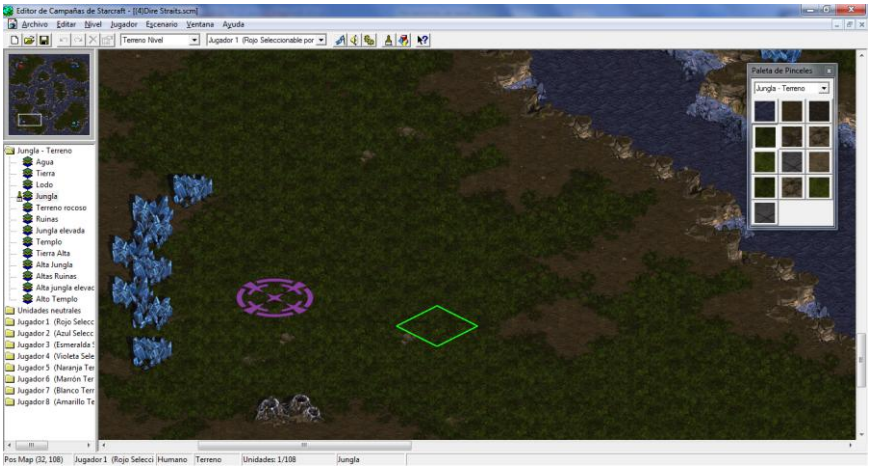

Figura 52. Editor de mapas oficial de StarCraft

Editores no oficiales[r146]: modo no jugable. Estos editores son los más utilizados por usuarios avanzados del juego, ya que permiten crear mapas, unidades, edificios… que el editor oficial no soporta. También permiten modificar elementos internos del juego: variables, nombres de archivos o de personajes, gráficos, sonidos… Algunos nombres destacados son Starforge, SCXE y SCMDraft.

Modo observador y repeticiones de partidas: modos no jugables. Estos modos consisten en el visionado de partidas sin formar parte de ellas. El modo observador considera al cliente local como un jugador más, aunque inactivo. De esta forma se puede observar en directo la acción. Las repeticiones de partidas, por otro lado, permiten visionar la partida en diferido: su manejo es similar al de una película de vídeo. Ambos modos son muy populares, dada la utilidad que tienen a la hora de aprender nuevas estrategias y estudiar el comportamiento de futuros contrincantes.

Recursos y economía

En este apartado se describen los recursos que pueden ser recogidos y/o utilizados por los jugadores como moneda de cambio durante el juego. Así mismo se describen los tipos de terreno propios de los diferentes mapas del juego.

Recursos: sirven para crear unidades y edificios y para investigar nuevas tecnologías. Controlar los puntos de extracción y extraer el máximo posible es la base de la economía del juego.

- Mineral: este elemento se representa en todo el juego por un prisma de color azul brillante ligeramente translúcido. Se extrae de zonas característicamente azules situadas sobre la corteza terrestre. Es un recurso limitado, ya que las vetas existentes no se regeneran: una vez explotadas sólo queda tierra yerma.
- Gas vespeno: este elemento se representa en todo el juego por un barril marcado característicamente en verde. Su lugar de recolección son ciertos agujeros hacia el subsuelo planetario de los que salen volutas de humo verde. El gas no puede ser aprovechado directamente, por lo que es necesario construir una estructura sobre el depósito para posibilitar la extracción. Este recurso no se consume hasta el límite, pero el período de regeneración suele ser prohibitivo en condiciones de juego normales.
- Límite de unidades: este elemento, propio de cada raza, representa una forma de limitar el número de unidades que controla un bando en un momento determinado de la partida. Cada raza trata este recurso de forma específica (los Terran utilizan edificios Granja, los Zerg utilizan unidades Supermente y los Protoss utilizan edificios Pilón).

Tipos de terreno

La versión original del juego incluye cinco tipos de terreno en total que se dividen en cuatro de exterior y uno de interior. Estos terrenos son los que se observan a medida que se desarrolla el modo Campaña.

- Instalaciones: representan los laboratorios Terran. Predominan los colores grisáceos y los aspectos metálicos, y se pueden apreciar pantallas, puertas automáticas, escaleras…
- Jungla: representa al mundo original Protoss, de nombre Aiur. Predominan los colores verde y marrón, y se pueden apreciar árboles, templos de piedra, ríos, elevaciones sobre el terreno…
- Mundo ceniza: utilizado para ilustrar Char, el mundo original Zerg. Predominan el negro y el rojo, y se pueden apreciar elementos volcánicos, lava, árboles quemados…
- Plataforma espacial: representa las plataformas Terran en el espacio. Predomina el negro, el gris y los aspectos metálicos. Siempre tendrá un fondo estrellado. Se aprecian elementos como carreteras, antenas, construcciones sobre y bajo el nivel del suelo…
- Tierras áridas: representa mundos de escasa vegetación como son Mar Sara, Antiga Prime y Tarsonis. Predominan los colores arenosos (amarillo, marrón), aunque todavía se puede apreciar cierta cantidad de vida vegetal.

Raza Terran

La raza Terran, conocida popularmente como “los Terran”, representan una visión de la raza humana futurista y de ficción. Según la historia del juego, esta raza llegó al sector Koprulu relativamente tarde. Son los descendientes de una expedición de colonos enviada desde la Tierra, aunque perdieron el contacto siglos atrás. La tripulación original estaba formada por disidentes e insurrectos que el gobierno de la Tierra consideró prescindibles. Los supervivientes de la expedición establecieron tres colonias que, con el tiempo, se convertirían en la base para los bloques de poder más significativos del espacio Terran: la Confederación, la Asociación Kel-Morian y el Protectorado Umojan.

Durante el desarrollo del juego, la corrupta Confederación es tomada a la fuerza por el terrorista y revolucionario Arcturus Mengsk durante el caos de la invasión Zerg. Más tarde, el dominio Terran renace de sus cenizas, agrupando lo que queda de los tres bloques de poder: el gobernador de esta alianza es el autoproclamado Emperador Arcturus I.

Los Terran son una clara muestra de supervivencia. Desde su origen, los conflictos internos y la necesidad de adaptación a un mundo extraño les han marcado profundamente. Aunque no poseen la avanzada tecnología Protoss ni la destreza Zerg, sí cuentan con un variado conjunto de unidades. Desde la infantería básica como los soldados, hasta los cruceros de batalla, equipados con armamento pesado, las unidades Terran disponen de sólidas armaduras, abundante potencia de fuego y un gran número de unidades para resistir ante el enemigo. Los Terran destacan en situaciones defensivas donde dominan sus búnkeres y sus tanques de asedio.

Ventajas:

- El jugador puede construir edificios en cualquier lugar.
- Los edificios principales pueden migrar, volando hasta instalarse en otro sitio.
- La mayoría de unidades pueden atacar a distancia.
- La mayoría de unidades pueden atacar a objetivos de tierra y de aire.
- El jugador dispone de dos unidades con invisibilidad limitada (Fantasma y Espectro).
- La unidad Nave de la ciencia puede neutralizar fácilmente a las otras dos especies: el ataque Impacto EMP reduce a la mitad los puntos de impacto de los Protoss (elimina el escudo), mientras que la habilidad Irradiar termina matando a cualquier unidad Zerg que no sea capaz de defenderse o huir.
- La unidad Fantasma puede bloquear cualquier unidad mecánica.
- La unidad VCE puede arreglar completamente las unidades mecánicas.
- La unidad Marine tiene un ligero beneficio respecto a las otras unidades comparables: pueden atacar a distancia.

- El jugador dispone de armas de destrucción masiva (Misil nuclear y Arma Yamato).
- La unidad Tanque de Asedio y el edificio Búnker pueden ser combinados para presentar una defensa inexpugnable.
- El edificio anexo Estación ComSat puede revelar cualquier lugar del mapa, descubriendo incluso las unidades invisibles.
- El jugador puede emplear multitud de habilidades para neutralizar emboscadas de otras razas. Por ejemplo, un solo Impacto EMP de una Nave de la ciencia elimina los escudos Protoss cercanos y corta el suministro de energía de las unidades que estén en su rango de acción.
- El jugador puede mejorar las habilidades e investigar las tecnologías más rápido que las otras razas.

Desventajas:

- El jugador necesita mucho espacio para construir.
- La mayoría de los edificios requiere un edificio anexo para proporcionar toda su funcionalidad.
- La mayoría de unidades está en desventaja respecto a sus homólogas de otras razas por el hecho de ser mecánicas.
- Todos los edificios arden cuando están dañados, perdiendo progresivamente puntos de vida. Terminan destruyéndose a sí mismos si no son reparados.
- La unidad VCE suele quedarse atrapada con mayor frecuencia que sus homólogas de otras razas, perdiendo así la unidad y teniendo que crear una nueva.
- La unidad Tanque de asedio puede producir fuego amigo, dado que sus proyectiles producen una explosión bastante grande.

Figura 53. Escena de StarCraft (Raza Terran)

El ejército Terran está formado por las siguientes unidades:

- Unidades terrestres: Vehículo de Construcción Espacial (VCE), Marine, Murciélago de fuego, Fantasma, Tanque de asalto, Goliat, Buitre.
- Unidades aéreas: Espectro, Nave de evacuación, Crucero de batalla, Nave de la ciencia.
- Estructuras básicas: Centro de mando, Depósito de suministro, Barracas, Bahía de ingeniería, Academia, Búnker, Torreta de misiles.
- Estructuras avanzadas: Fábrica, Puerto estelar, Laboratorio de ciencia, Arsenal.
- Anexiones: Estación ComSat, Silo nuclear, Torre de control, Centro de operaciones secretas, Laboratorio de física.

Raza Zerg

Esta raza está formada por un conjunto de seres de aspecto extraterrestre con pensamiento de colmena. En todo momento se encuentran guiados por una criatura reina conocida como Supermente. La característica principal es que todo lo relativo a esta raza está vivo: unidades, edificios… incluso el propio suelo, que se conoce como biomateria.

Los Zerg son conocidos por su ferocidad y su aspecto grotesco. Se basan en el ataque de gran número de unidades ya que, al estar constituidos enteramente por materia biológica, sus unidades son más económicas y de rápida creación.

Esta raza tiene su origen en el primitivo mundo de Zerus. Fueron creados por los Xel'Naga, una raza ya olvidada. Inicialmente, sus individuos fueron extremadamente pequeños, con aspecto de oruga e incapaces de manipular físicamente su entorno. Aun así, los Zerg se adaptaron y sobrevivieron. Con el tiempo, desarrollaron la habilidad de "enterrarse" en la carne de las especies indígenas menos vulnerables. Y, cuando fueron capaces de controlar los procesos metabólicos y anatómicos de sus anfitriones, los Zerg utilizaron sus nuevos cuerpos para manipular sus entornos.

La química de los Zerg empezó a mutar y a adaptarse al volumen de nuevo material genético que estaba siendo procesado. Sin embargo, existía siempre un impulso innato de consumir solamente las especies evolutivamente más avanzadas. Los Zerg son selectivos en cuanto a qué especies consumir, asegurándose en cada etapa del desarrollo que ellos están siempre en la parte superior de la proverbial cadena alimenticia. Tras un sorprendente corto espacio de tiempo, las especies de Zerus evolucionaron hasta parecer una sola raza terriblemente voraz.

Tras innumerables victorias, los Zerg tuvieron conocimiento de otra raza, llamada Protoss. Reinaba entonces un ambiente de inquietud, pues poco se sabía de esta nueva raza: que fue creada también por los Xel'Naga, que era tan antigua como los

propios Zerg… y que ninguna especie conocida había conseguido derrotarlos en batalla.

La innata selección genética impulsó a los Zerg a buscar a los Protoss a lo largo y ancho de la galaxia, buscando el enfrentamiento que los haría infinitamente más sapientes. Cuando los encontró, los Protoss estaban trabados en combate con otra raza: los Terran. La batalla parecía no inclinarse hacia ninguno de los bandos… Por tanto, sabiendo que el conflicto venidero sería el mayor desafío de su existencia, los Zerg se ocultaron y se quedaron observando. En este punto comienza la historia descrita por StarCraft.

Debido al hecho de que todo elemento Zerg está vivo, el modo de juego es bastante peculiar. No se puede edificar en ningún sitio que no sea el suelo con biomateria, de color morado característico. La expansión de las colonias, por tanto, se produce por la expansión de suelo con biomateria. Inicialmente sólo se puede construir un Criadero, que irá extendiendo el suelo conquistado. Después se podrán añadir edificios, que extenderán a su vez el suelo apto para la construcción. Otras características destacables son que los edificios y unidades se regeneran en lugar de repararse, y que las unidades no se entrenan, evolucionan a partir de larvas.

Los Zerg son característicos por su ferocidad y su aspecto grotesco. Su táctica básica se centra en la utilización del factor numérico, atacando a la vez un gran número de unidades. Al estar formados enteramente por materia biológica, las unidades son más económicas y de rápida creación. En general, son una raza muy numerosa, rápida y versátil.

Ventajas:

- Las unidades básicas “Larva” evolucionan automáticamente y con gran velocidad, de manera que siempre hay alguna disponible para el jugador.
- Los costes de producción son menores, lo que les permite explotar una potencial ventaja numérica.
- La regeneración de las unidades les confiere otra ventaja táctica en combate: el factor tiempo.
- Las unidades Zergling y Atormentador, si bien no son las más poderosas, evolucionan de manera especialmente económica: dos unidades producidas por cada Larva invertida.
- Las unidades “Superamo” son voladoras y sirven para transportar otras unidades, para detectar unidades ocultas y para aumentar el límite de unidades controladas. Esto supone una considerable ventaja táctica y de tiempo.
- Las unidades son muy rápidas en sus desplazamientos sobre el mapa.
- La mayoría de las unidades terrestres pueden utilizar la habilidad Madriguera: se introducen bajo tierra y no pueden ser atacadas. Mientras tanto, pueden vigilar y regenerar sus puntos de vida. Las posibles utilidades son la realización de emboscadas o el espionaje.

Desventajas:

- Las unidades, cuando se encuentran aisladas, son comparativamente más débiles que sus contrapartidas de otras razas.
- Solo pueden construir edificios en el terreno especial, la biomateria. Es necesario construir un Criadero o un Extractor para aumentar la superficie conquistada.
- Para construir un edificio se debe sacrificar una unidad Zángano. Esto supone un desvío de recursos: o bien se construye un edificio, o bien se produce una unidad de ataque.
- La evolución de nuevas unidades atacantes es lenta: es necesario tener un suministro continuo de Larvas en sus respectivos Criaderos. La construcción de estos edificios supone un gran coste económico y favorece la dispersión de las unidades a lo largo del mapa.
- Los Atormentadores y los Terran Infestados, a pesar de ser dos grandes unidades de apoyo, tienen a su disposición la opción del suicidio. Cuando esto ocurre, el jugador pierde inesperadamente parte de su ejército, lo cual es una considerable desventaja.
- Los ataques, considerados individualmente, son menos dañinos que los de las otras razas.
- Las mutaciones y evoluciones de las diferentes unidades obliga, en algunos casos, a permanecer totalmente quieto y sin defensas. Esto hace del proceso de evolución un punto vulnerable que es aconsejable espaciar en el tiempo.

Figura 54. Escena de StarCraft (Raza Zerg)

El ejército Zerg está formado por las siguientes unidades:

- Unidades terrestres: Larva, Zángano, Zergling, Hidralisco, Corruptor, Terran Infestado, Cría, Ultralisco.
- Unidades aéreas: Superamo, Mutalisco, Atormentador, Guardián, Reina.
- Estructuras básicas: Criadero, Guarida, Colmena, Reserva de reproducción, Cámara de evolución, Extractor, Colonia de biomateria, Colonia hundida, Colonia de esporas, Guarida de hidralisco.
- Estructuras avanzadas: Nido de la reina, Espiral, Espiral mayor, Canal nydus, Montículo del corruptor, Caverna del ultralisco.

Raza Protoss

Los Protoss forman una de las tres razas existentes en StarCraft. Se encuentran en claro contraste con los adaptativos Terran y los salvajes Zerg, y destacan por su impasibilidad y conservadurismo. A lo largo del tiempo han madurado una tecnología muy avanzada, al tiempo que han conseguidos diversas habilidades mentales. Se consideran a sí mismos la raza más poderosa de la galaxia conocida. Sin embargo, si los Protoss tienen un punto débil, ése es su negativa a aceptar el cambio…

La raza es originaria del mundo de Aiur; fue creada por los Xel'Naga, al igual que la raza Zerg. Los Xel'Naga emplearon gran cantidad de tiempo guiando sutilmente los pasos de sus hijos, hasta que alcanzaron el éxito: los Protoss llegaron a un estado de sensibilidad y conciencia totales. De esta manera se convirtieron, gradualmente, en seres altamente intelectuales e introspectivos, consiguiendo niveles muy altos, no sólo en sus avances culturales, sino también en sus avances individuales y colectivos.

Con el tiempo, la primitiva sociedad Protoss fue tornándose cada vez más individualista. Lo que un día fue una sociedad sólida con un gobierno firme y centralizado se convirtió en multitud de pequeñas tribus que guerreaban entre sí por cualquier conflicto. Al observar este comportamiento, los Xel'Naga dieron su creación por fallida y partieron en busca de nuevas oportunidades de creación.

Nada de interés sucedería durante los siguientes millones de años, hasta que un día, por casualidad o a causa del destino, un individuo llamado Khas desenterró unos antiguos artefactos: los Cristales Khaydarin. Dichos elementos, olvidados por los Xel'Naga, permitieron canalizar las energías presentes en el planeta a través de Khas, permitiéndole el acceso a la unión psíquica primordial de su raza. Por primera vez en millones de años se tocó la fibra primitiva de los Protoss. Khas, inundado por las emociones que emanaban de cada uno de los miembros de su raza, supo que los Protoss no habían perdido su unión primitiva: simplemente habían olvidado cómo sintonizarse con ella.

A partir de entonces, la raza volvería a su antiguo esplendor tecnológico y social, aunque sin llegar a los extremos que causaron largos años de guerras civiles. Es en este punto donde comienza la historia narrada en StarCraft: con el trágico encuentro del mundo Protoss y la raza Terran…

El modo de juego de esta raza no está exento de peculiaridades: las zonas edificables están limitadas, las estructuras no se construyen sino que se invocan, y todos los elementos, sin excepción, disponen de un escudo protector regenerativo.

Ventajas:

- Las unidades son muy resistentes, ya que cuentan con salud y escudo psiónico. Tras agotar el escudo, la unidad perderá salud, pero normalmente el escudo no se llega a perder.
- Las unidades constructoras no están inutilizadas durante todo el proceso de construcción. Simplemente solicitan la invocación del edificio y retoman sus tareas previas.
- Todas sus unidades tienen la posibilidad de hacerse invisibles gracias a la unidad Árbitro, con la única excepción de los propios Árbitros. Esto supone una clara ventaja táctica. Los Observadores y los Templarios oscuros poseen la habilidad por sí mismos.
- Las habilidades de sus unidades tienen son muy dañinas. Una sola activación de las mismas puede causar estragos entre las filas enemigas.
- La construcción defensiva Cañón de fotones puede atacar tanto a tierra como a aire, fortaleciendo definitivamente una posición estratégica. También permite detectar unidades ocultas enemigas.

Desventajas:

- Las unidades Protoss son más costosas que sus contrapartidas Terran o Zerg.
- No es posible reparar los edificios.
- La construcción de edificios está limitada al área de efecto del Pilón más cercano.
- Si un Pilón es destruido, los edificios en su área de influencia quedan sin energía, siendo automáticamente inhabilitados. Esta inhabilitación, además, cancela la posible actividad del edificio, anulando el entrenamiento de nuevas unidades o la investigación de mejoras.
- Las unidades tienen un tiempo de entrenamiento mucho más lento que el de las otras razas.
- Algunas unidades utilizan parte de los recursos del jugador cada vez que atacan, lo que se traduce en una fuente más de gastos.
- Cada unidad consume dos o más unidades de población, alcanzando el límite con facilidad. La excepción a esta regla es la unidad Sonda, que solamente consume una unidad.

- El hecho de no poder curar a las unidades implica que éstas deben ser reemplazadas continuamente.
- Las unidades son débiles frente a unidades bien acompañadas, como Zerglings con Mutaliscos o Marines con Murciélagos. En ambos casos, aunque los ataques individuales son muy poderosos, la distribución y la cantidad de enemigos juega en contra. Por tanto, a igualdad de recursos invertidos, los Protoss suelen perder.
- Sus escudos pueden desaparecer instantáneamente con la habilidad "Impulso Electromagnético" de la Nave de la ciencia Terran.
- Las unidades aéreas son muy débiles contra unidades terrestres: hacen menos daño aire-tierra que aire-aire.

Figura 55. Escena de StarCraft (Raza Protoss)

El ejército Protoss está formado por las siguientes unidades:

- Unidades terrestres: Sonda, Fanático, Dragón, Alto templario, Arconte, Reaver.
- Unidades aéreas: Observador, Lanzadera, Explorador, Árbitro, Transporte.
- Estructuras básicas: Nexo, Pilón, Asimilador, Acceso, Fragua, Cañón de fotones, Núcleo cibernético, Batería de escudo.
- Estructuras avanzadas: Ciudadela de Adun, Puerto estelar, Instalación robótica, Archivos Templarios, Baliza de flota, Bahía de apoyo robótico, Observatorio, Tribunal del árbitro.

Trama argumental

La historia narrada comienza pocos días después del primero de los ataques. El gobierno Terran se encuentra en un estado de pánico, ya que está bajo el ataque de

los Protoss, la raza Zerg acaba de realizar su primera y hostil aparición, y Arcturus Mengsk y sus rebeldes están protagonizando un aumento significativo de la violencia.

La Confederación finalmente sucumbe ante la actividad rebelde. La táctica utilizada fue la activación de un señuelo para que los Zerg atacaran la capital Terran. En el caos reinante, Arcturus Mengsk abandona a su segunda de a bordo, Sarah Kerrigan, para que los Zerg den buena cuenta de ella.

Los Zerg capturan e infestan a Kerrigan, creando su mejor agente. Esta traición es causa de que otro comandante de Mengsk, Jim Raynor, desertara con armas y bagajes. Mengsk se autoproclama emperador del Dominio Terran con el nombre de Arcturus I.

En este punto, los Protoss atacan la colmena principal de los Zerg. Al frente de la operación se encuentra el comandante Tassadar, y el templario oscuro Zeratul es el segundo al mando. Este último, durante el asesinato del máximo dirigente de la colmena permitió inconscientemente que la Supermente Zerg conociera el acceso a la ubicación de la nación de los Protoss.

Los Zerg se movieron rápidamente para invadir la capital Protoss, en un esfuerzo para completar cuanto antes su búsqueda de la perfección. Mientras tanto, Tassadar regresa a territorio Protoss, donde es perseguido por dejar obrar libremente a Zeratul.

Finalmente, con ayuda del Terran Raynor y del Protoss Fénix, Tassadar es capaz de lanzar un desesperado ataque sobre la Supermente Zerg, sacrificándose a sí mismo para matar a la criatura.

2.3.3. Actualidad

Esta sección se dedica a ofrecer una imagen actual del mundo en relación al videojuego StarCraft. Para cumplir este cometido se abandona el estilo narrativo en favor de la enumeración de hechos notables.

El nuevo estilo ha sido seleccionado por ser capaz de ofrecer una imagen realista, independiente y, sobre todo, relativamente atemporal. Sin embargo, es imposible mantener rigurosamente actualizado un apartado como el actual, por lo que se espera que el lector interesado investigue con mayor profundidad aquellos hechos que puedan experimentar un mayor cambio a lo largo del tiempo.

- El juego protagoniza tres entradas en el Guinness World Records[r147]: es el "juego de estrategia para ordenador más vendido", es causa del "salario más grande en competición de videojuegos profesional" y, con más de 120000 aficionados congregados, tuvo la "mayor audiencia protagonizada por una competición de videojuegos".
- StarCraft ha ocupado el temario entero en una asignatura[r148] de la Universidad de Berkeley (primavera de 2009).

- El videojuego tiene una tremenda popularidad en Corea del Sur:
    - Los jugadores pueden adquirir el estatus de jugador profesional, que requiere una licencia emitida por el colegio profesional asociado.
    - Los jugadores son celebridades conocidas por todo el país.
    - Existen tres canales dedicados a retransmitir competiciones de videojuegos, siendo StarCraft el que mayor tiempo ocupa en pantalla.
    - Como resultado de lo anterior, los jugadores profesionales perciben cifras escandalosamente altas en concepto de salario.
    - Otra consecuencia de lo expuesto es que las trampas tienen legislación específica. En 2010 terminó un proceso en el que cuatro jugadores recibieron penas de cárcel de entre uno y tres años, y multas de entre uno y tres millones de dólares.
- Desde el año 2000 se han publicado multitud de libros ambientados en el juego, sea como precuela, secuela, aventura paralela o aventura interconectada. Algunos de estos títulos son "Uprising", del desarrollador Micky Neilson, "Liberty's Crusade" de Jeff Grubb o "Queen of Blades" de Aaron Rosenberg.
- Hay una gran diversidad de figuras de acción y coleccionables. Las principales compañías que trabajan este aspecto con Blizzard Entertainment son Academy Hobby Model Kits y Fantasy Flight Games. Esto daría lugar, irónicamente, a la creación de un juego de mesa: StarCraft Adventures.
- El juego tiene una serie de elementos adicionales:
    - Expansiones: en esta categoría encaja únicamente StarCraft: Brood Wars, que continúa la historia narrada en el punto donde concluyó en la primera entrega.
    - Campañas oficiales: conforman realidades previas, paralelas o futuras a la narración del juego ("Enslavers", "Precursors", "Enslavers 2", "Deception", "Resurrection IV").
    - Campañas no oficiales: similares a las oficiales, pero desarrolladas por la comunidad y avaladas por Blizzard Entertainment ("Insurrection", "Retribution").
    - Campañas de la comunidad: existe una innumerable cantidad de campañas que por un motivo u otro no han recibido el carácter oficial. Estas historias han sido enteramente confeccionadas por la comunidad de usuarios mediante programas de edición de mapas, de gráficos…

2.4. XNA Game Studio

Esta sección se dedica al estudio comparativo del framework elegido para desarrollar el trabajo necesario para completar este Proyecto de Fin de Carrera. El objetivo es ofrecer información suficiente al lector para comprender por qué se ha

elegido XNA como base del videojuego creado, así como sus ventajas, inconvenientes y alternativas.

2.3.1. Estudio de alternativas

Uno de los primeros pasos al considerar la realización de un proyecto es la elección de las herramientas de trabajo. Estas utilidades deben ayudar a realizar la serie de tareas que componen el proceso de realización del proyecto: diseño, construcción, prueba, implementación, documentación…

Estas herramientas se denominan marcos de trabajo o frameworks. En el caso que nos ocupa, el término común es "motor de videojuego" o "game engine". Estas creaciones ayudan, en mayor o menor medida, a llevar el proyecto a buen término. Algunas de las funciones que suelen incluir están relacionadas con gráficos ("motor gráfico") en dos o tres dimensiones, aspectos físicos ("motor de colisiones"), de sonido ("motor de audio"), de inteligencia artificial, de redes, de administración de datos…

De la misma manera, algunos motores de videojuegos están pensados para sistemas concretos, mientras que otras tienen un carácter más generalista. Estos y otros aspectos fueron sopesados a la hora de seleccionar XNA como base del proyecto. Este apartado se dedica a recopilar la información consultada sobre los distintos motores y frameworks que fueron considerados como posible base tecnológica del proyecto.

- **Blender 3D**[r149]**:** es una aplicación gratuita (GNU) concebida para diseño 2D/3D y animación gráfica. Sus principales características incluyen diseño, animación, iluminación… pero también es posible realizar simulaciones de fluidos, detección de colisiones o simulaciones físicas. En las últimas versiones publicadas se ha aprovechado la potencia descrita para crear aplicaciones interactivas en dos y tres dimensiones, varias de ellas videojuegos de temáticadiversa. Blender se puede utilizar bajo Linux, Mac OS o Windows. El diseño se realiza a partir de su interfaz gráfica, mientras que el lenguaje de programación Python soporta la gestión lógica.

- **Crystal Space**[r150]**:** en este caso, el sistema es un completo framework de carácter general diseñado para desarrollar aplicaciones 3D, aunque se referencia aquí por su utilización como motor de videojuegos. Las ventajas que tiene este sistema son su portabilidad (Linux, Mac OS X, Windows), su compatibilidad (OpenGL, SDL, X11, SVGALib) y que está programado en lenguaje C++.

- **DIV2 Game Studio**[r151]**:** es un lenguaje de programación similar a Pascal y a C orientado a la creación de videojuegos bajo MS-DOS. Además, el sistema DIV2 incluye un editor de gráficos y otro de sonido. El conjunto completo está

simplificado para no requerir amplios conocimientos de programación. Aunque este sistema está abandonado han surgido diversas continuaciones que mantienen la base ofrecida (Proyecto Fénix, Gemix Studio, BennuGD) y la amplían hacia sistemas modernos como GP2X o Nintendo Wii.

- **Gamebryo**[r152]**:** sistema perteneciente a la compañía Emergent Game Technologies orientado a la gestión de gráficos 3D. Está escrito en C++. Tiene la gran ventaja de soportar todos los sistemas modernos, sean de sobremesa (Linux, Mac OS, Windows) o de consola (Gamecube, Wii, PS2, PS3, XBOX, XBOX360). Sin embargo, tiene el inconveniente de ser un sistema profesional con costosas licencias de uso.

- **Game Maker**[r153]**:** es un completo entorno de trabajo orientado al desarrollo rápido de aplicaciones (en este caso, videojuegos). Está basado en una modificación del lenguaje Delphi y no requiere gran conocimiento de programación ni de sistemas para crear los primeros juegos. El sistema tiene dos versiones: una gratuita, más versátil, y otra de pago, que aporta muchas funciones específicas.

- **Game Studio**[r154]: sistema perteneciente a Conitec Systems, y también conocido como 3D Game Studio o 3DGS. Es un sistema de desarrollo de videojuegos compuesto por un motor gráfico ("A8"), un editor de modelos 2D/3D, un editor de terreno, un editor de niveles y un sistema de depuración. El motor gráfico es especialmente potente, ya que incluye toda la computación gráfica conocida (sombras dinámicas y estáticas, iluminación exterior e interior, sombreado, texturas…), así como manejo automático de sonido, red e interfaz (teclado, ratón…). El sistema está dirigido a varios tipos de usuario y niveles de habilidad: desde un artista conceptual poco tecnológico hasta el programador de más bajo nivel. Por tanto, ofrece una ingente cantidad de opciones. Para el desarrollo de este proyecto únicamente se tendrá en cuenta que ofrece dos lenguajes de programación: C++ y Lite-C (C-Script). Existen diversas versiones: cada una de ellas supone más funcionalidad y coste que la anterior.

- **GtkRadiant**[r155]**:** es un software desarrollado por id Software y Loki Software, utilizado para crear mapas para videojuegos tipo FPS. Es la herramienta utilizada para el diseño de niveles de Quake III Arena[r156]. Su desarrollo inicial fue para Windows bajo el nombre Q3Radiant. Al adaptarse al sistema GTK+ se hizo compatible con Linux y Mac OS, al tiempo que se hizo independiente del videojuego concreto. El software es parcialmente libre: el código fuente se puede descargar del repositorio de id Software, pero el núcleo original es propietario y su uso en videojuegos comerciales sin el permiso explícito de la compañía no está permitido. La herramienta no ofrece soporte para lenguajes de programación: se limita a la creación de mapas.

- **Havok:** de nombre completo “Havok Game Dynamics SDK”, este software es un completo motor físico diseñado para otorgar realismo a las creaciones de ocio electrónico. En él se calculan en tiempo real las interacciones entre objetos y personajes del juego, sean de iluminación, de colisión, de gravedad… De cara al desarrollador ofrece directamente un conjunto de funciones: no es un entorno de desarrollo ni un conjunto de herramientas. Entre estas funciones destacan aquellas relativas a gravedad, masa, velocidad… El sistema tiene como requisito DirectX y OpenGL, ya que las utiliza como base para realizar sus cálculos. Por tanto, Havok destaca por su potencia y la de creaciones construidas sobre él[r157], pero tiene la desventaja de la complejidad: es muy denso para usuarios no profesionales.

- **LWJGL**[r158]**:** Lightweight Java Game Library es una librería de funciones Java dirigida a todo tipo y clase de programador: desde aficionados hasta profesionales. Proporciona una abstracción del paquete JNI (teclado, ratón…) y ofrece funcionalidad de OpenGL (gráficos) y de OpenAL (sonido), resultando en una cómoda manera de trabajar. El objetivo no es realizar desarrollos con rapidez, sino crear la posibilidad de desarrollar juegos en el lenguaje Java. LWJGL está disponible bajo licencia BSD, por lo que es de libre distribución.

- **M.U.G.E.N.**[r159]**:** es un sistema de creación de videojuegos en dos dimensiones creado y mantenido por Elecbyte. Tiene un gran reconocimiento entre desarrolladores y aficionados, lo cual es una ventaja. Además, el sistema es capaz de importar gran variedad de datos de otras creaciones, lo cual es otra ventaja. Sin embargo, este sistema tiene una gran desventaja: está diseñado para crear juegos de género lucha uno contra uno; cualquier otro tipo de creaciones es difícil de implementar.

- **OGRE 3D**[r160]**:** es un motor gráfico tridimensional de propósito general, aunque las clases en las que se basa permiten unir multitud de conceptos con aquellos utilizados en el desarrollo de videojuegos en tres dimensiones. Según los propios creadores y la comunidad de usuarios que los apoya, OGRE destaca por su buen diseño, flexibilidad, documentación y ejemplos. Está escrito en C++, y su intención es la de situarse por encima de librerías gráficas más básicas, ocultando la dificultad existente y aportando valor útil al desarrollador.

- **Source**[r161]**:** es el sistema de creación de videojuegos de Valve Software. Incluye todos los aspectos necesarios: gestión gráfica, física, de sonido, de red… También soporta lógica de juego personalizada en lenguaje C++. Es compatible con las plataformas PC (Windows, Mac OS), XBOX, XBOX 360 y PlayStation 3. Como ventaja cabe citar el excelente diseño modular que lo hace gradualmente utilizable y/o actualizable. Como desventaja es necesario destacar el coste de diversas licencias, ya que es un software propietario.

- **Unreal Engine**[r162]**:** es un completísimo conjunto de funciones creado y mantenido por Epic Games. Originalmente se utilizó en el juego Unreal, hecho que sirvió como prueba de funcionamiento. Posteriormente sería utilizado en toda la saga Unreal, así como en una enorme cantidad de creaciones de empresas teóricamente competidoras[r163]. El lenguaje elegido para esta creación es C++. Actualmente se comercializan la versión 3 para todos los usuarios y la versión 4 bajo licencia comercial. Es compatible con Linux, Mac OS, Windows, la mayoría de consolas (Dreamcast, Gamecube, Wii, XBOX, XBOX 360, PSP, PS2, PS3) y la mayoría de dispositivos portátiles (iOS, Android).

- **XNA**[r164]**:** es un conjunto de funciones creado y mantenido por Microsoft. Su concepción inicial fue dar la posibilidad de crear juegos para XBOX 360 a desarrolladores aficionados. Siendo un producto de Microsoft no es raro que exista una documentación razonablemente buena y siempre actualizada. Los videojuegos resultantes son compatibles con un amplio abanico de plataformas: Windows XP, Windows Vista, Windows 7, XBOX 360, Windows Phone y Zune. Finalmente, la comunidad de usuarios ha creado muchísimo contenido, por lo hay gran cantidad de contenido de ejemplo, manuales, tutoriales… El lenguaje de programación oficialmente soportado es C#, aunque es posible trabajar con cualquier lenguaje que esté basado en .NET. La licencia de uso no tiene coste alguno; sin embargo, la licencia comercial implica ceder parte de los beneficios a Microsoft.

Todo lo anterior son solamente algunas herramientas que es posible encontrar en la red y que permiten diseñar y desarrollar videojuegos. Como se ha visto, muchas tienen el inconveniente de ser de pago o tener licencias restrictivas, sobre todo en los casos más avanzados y potentes. En otros casos, la complejidad del sistema a abordar puede restar atractivo al proyecto, o resultar demasiado simple para lo que se quiere acometer. De la misma manera muchas de estas herramientas se apoyan en elementos extraños: librerías externas, lenguajes de programación poco conocidos por el alumno… Esto obliga a manejar varias librerías, herramientas y lenguajes para conseguir un resultado óptimo. Por último resaltar que la mayoría proporciona una mínima documentación y escasos ejemplos, lo que complica el desarrollo.

Casi todos los problemas quedan solventados en mayor o menor medida en XNA, API utilizado para el desarrollo de este proyecto. A continuación se detalla en qué consiste XNA y se justifica la elección de XNA en su versión 3.1.

2.3.2. Elección de XNA

La lista de herramientas es increíblemente extensa. En el apartado anterior se detallan únicamente las más relevantes, pero se ha observado que existen varias decenas, casi cientos de ellas. Como se puede observar, los productos más populares

son los sistemas gráficos, alrededor de los cuales suele agruparse un conjunto de utilidades: motor físico, motor de colisiones, editor de terrenos, niveles o recursos gráficos en general…

En general, la mayoría de las opciones revisadas ofrecen una funcionalidad generalista y diversa, pero tienen unos inconvenientes comunes:

- Software propietario: algunos sistemas son software propietario, por lo que es necesario adquirir licencias de uso y/o comerciales.

- Derechos sobre el producto: las licencias son, en ocasiones, muy restrictivas, sobre todo en los entornos más novedosos, avanzados o potentes. En estos casos, el sacrificio es mayor, ya que al coste derivado de las licencias se le añade parte de los derechos sobre el videojuego creado.

- Complejidad: en algunos casos, los sistemas descritos son muy complejos, excediendo tanto lo que es razonable para un Proyecto de Fin de Carrera como lo que es lógico emplear para llevar a cabo un videojuego de estrategia en tiempo real en dos dimensiones.

- Simplicidad: en otros casos, la complejidad destaca por su ausencia. El inconveniente es que los sistemas de creación proporcionan muy poco control sobre el desarrollo, por lo que resultan demasiado simples para el proyecto actual.

- Código desconocido: en ocasiones, las herramientas de creación de videojuegos se basan en lenguajes de programación o librerías que son desconocidas para el alumno. Mientras que esto no representa un inconveniente serio, sí desaconseja el uso de estas herramientas a favor de otras más conocidas.

- Contenido de ayuda: por último, cabe destacar la escasa documentación de algunos motores, o la relativa ausencia de contenido de ejemplo o de ayuda.

Estas cuestiones encuentran un equilibrio adecuado en Microsoft XNA. En primer lugar, es posible utilizar el sistema para desarrollar videojuegos no comerciales. El único coste derivado de su utilización es la parte de los beneficios que se debe ceder a Microsoft al comercializar las creaciones realizadas.

Por otro lado, la propuesta no es excesivamente compleja ni simple. Es muy específica en cuanto a que alienta el desarrollo de juegos en dos dimensiones, lo cual es tanto una ventaja como una desventaja: se hace más sencillo realizar el proyecto pero es difícil crear código que exceda el diseño base de XNA.

El sistema base de XNA, Microsoft .NET, hace posible desarrollar código en C#. Tanto el lenguaje como el estándar han sido trabajados a lo largo de la carrera, por lo que no suponen dificultad ninguna. De manera adicional, se cuenta con una amplísima

base de recursos, bien en forma de documentación, de ejemplos, de tutoriales, de componentes…

Finalmente, el código producido es compatible con los sistemas Windows XP, Windows Vista, Windows 7, XBOX 360, Zune y Windows Phone. Esta selección de sistemas es amplia, aunque tiene como desventaja la exclusividad de una única familia de productos.

2.3.3. XNA versión 3.1

En apartados anteriores se compararon diversos motores de videojuegos, eligiéndose justificadamente Microsoft XNA. En este apartado se realiza una presentación formal en profundidad de este sistema, detallando la selección de su versión 3.1 como base del presente PFC.

El nombre del producto es, sarcásticamente, un acrónimo ("XNA is Not Acronymed", que en castellano significa "XNA no es acrónimo"). Actualmente el producto está totalmente integrado en Visual Studio, utilizándose como una extensión de las librerías proporcionadas por .NET. Está compuesto por un conjunto de funciones, un programa de edición de sonido y la documentación asociada a ambos elementos.

Inicialmente fue diseñado como una capa superior de Microsoft DirectX con la intención de acercar este sistema a los desarrolladores. Con el tiempo y la evolución del mercado las funcionalidades implementadas han crecido. En el proceso también se ha separado de DirectX. Actualmente se fundamenta sobre .NET, que a su vez utiliza parte de DirectX y de otras librerías para cumplir las funcionalidades que ofrece.

Figura 56. El motor de videojuegos XNA y su entorno

Hoy en día, XNA es una herramienta que facilita parte del proceso software de creación de un videojuego: desarrollo, implementación, pruebas, implantación y mantenimiento. Este cometido se cumple mediante un amplio conjunto de funciones de alto nivel que ocultan la interacción con el hardware. Algunos aspectos simplificados de esta manera son la gestión de memoria dinámica, la carga de recursos desde fichero o el dibujado en pantalla mediante la tarjeta gráfica.

Otra consecuencia derivada del uso de .NET es la posibilidad de desplegar el código en multitud de plataformas: Windows XP, Windows Vista, Windows 7, XBOX 360, Zune y Windows Phone (ver Figura 56). Será necesario realizar cierta cantidad de modificaciones, pero serán cuestiones mínimas. Por ejemplo, será necesario cambiar el ratón de Windows por el mando de juego de XBOX 360.

Para concluir la introducción a XNA es necesario destacar que se puede codificar en Visual C#, Visual C++ o Visual Basic, ya que estas opciones están soportadas por el compilador de .NET. Se ha seleccionado C# para realizar este PFC por haber sido trabajado extensivamente a lo largo de la carrera.

A continuación comienza el análisis técnico de XNA. En la Figura 57 se muestra el diagrama interno del sistema, donde se pueden observar las capas que componen el modelo del motor de videojuegos.

Figura 57. Modelo interno de XNA

XNA está estructurado en cuatro capas: de más bajo nivel a más alto, son Plataforma, Núcleo del framework, Framework extendido y Juegos. Cada capa tiene diversos elementos aislados: algunos son instalados con el motor de videojuegos, otros son creados por el usuario individual y otros son incorporados desde la red mediante colaboración con otros usuarios. A continuación se describe cada capa en detalle.

- **Plataforma:** es la capa de más bajo nivel. Su cometido es la interacción con los recursos físicos del sistema. Se apoya en el hardware de la máquina y en el sistema operativo para proporcionar funcionalidad a las capas superiores. En esta capa tienen cabida Direct3D (acceso a tarjeta gráfica), XACT (operaciones con archivos de sonido), XINPUT (obtención de información de entrada de usuario), XCONTENT (manejo de recursos gráficos)…

- **Núcleo del framework:** esta capa tiene como base la capa Plataforma y ofrece funcionalidad diversa a las capas superiores. Esta capa es utilizada de manera automática por XNA en la capa Framework extendido, pero también es ampliamente consultada por el desarrollador. Se proporcionan funciones básicas orientadas a la creación de código propio. Las áreas agrupadas en esta capa son:

    - **Gráficos**: en esta parte se agrupan las funciones necesarias para interactuar con el subsistema gráfico. Se trabaja con operaciones de Direct3D versión 9, aunque XNA permite también definir y aplicar efectos propios.

    - **Audio**: este apartado recoge las funciones destinadas a operar con recursos de sonido. En este caso, XNA trabaja con DirectAudio para ofrecer el soporte necesario.

- **Entradas**: en esta sección tienen cabida las funciones necesarias para consultar las posibles entradas de usuario. XNA mantiene actualizado en todo momento un vínculo con DirectInput que permite acceder a cualquier información de teclado, ratón o mando de juego.

- **Matemáticas:** esta parte amplía las funciones matemáticas aportadas por .NET con aquellas más comúnmente utilizadas en el desarrollo de videojuegos (cifras en general, vectores, matrices, operaciones matemáticas simples, interpolaciones, cálculos estadísticos, cálculos de superficie, cálculos de volumen, operaciones sobre conjuntos…). Un aspecto a destacar es que se define un conjunto de valores constantes (PI, E…), favoreciendo la independencia de la precisión que pueda ofrecer la arquitectura subyacente.

- **Almacenamiento:** esta abstracción de XNA proporciona acceso uniforme a la memoria secundaria del sistema. La ventaja que aporta es la transparencia de interacción con el medio y la independencia de la plataforma.

- **Framework extendido:** esta capa se sitúa por encima de Plataforma y de Núcleo extendido. Su cometido es facilitar las labores de desarrollo mediante la automatización de ciertas tareas.

  - **Modelo de aplicación:** las aplicaciones XNA tienen un flujo de llamadas predefinido y constante, lo que favorece la claridad y ayuda a enfocar los esfuerzos en partes más concretas del código.

  - **Administrador de contenido:** evita la problemática relacionada con la interacción con memoria principal y secundaria. Este subsistema se ocupa de la carga y descarga de los recursos que tengan un tipo conocido y soportado. La gestión que realiza es muy útil: si los recursos no se utilizan más, se descargan; si se utilizan mucho, se bloquean en memoria; si se intenta cargar dos veces el mismo recurso, se devuelve una referencia al recurso que ya estaba en memoria… Este subsistema es ampliable mediante la definición de tipos propios de usuario.

- **Juegos:** es la capa más alta de XNA. Esta capa se basa en las funcionalidades ofrecidas por todas las demás. Aquí se encuentra el código y el contenido creado por el usuario: sus propias clases, sus propios datos… En esta sección tienen cabida también elementos como los kits de inicio (juegos proporcionados por Microsoft para aprender a usar el sistema, para tomar ideas…) o los componentes (código creado por otros usuarios que realiza alguna función genérica, como la transición entre pantallas o la creación de un teclado virtual).

Capítulo 3

Fase de análisis

Este capítulo recoge el análisis previo a las etapas de diseño e implementación.

El primer apartado, **3.1. Alcance del sistema**, recoge los aspectos generales que debe incluir el software resultante de este proyecto.

El segundo apartado, **3.2. Requisitos de usuario**, plasma el catálogo inicial de requisitos. En él se especifica lo que el sistema debe hacer, así como las limitaciones que debe observar.

El tercer apartado, **3.3. Requisitos software**, explora en profundidad los requisitos iniciales, detallando aspectos del desarrollo que cubrirán las funcionalidades deseadas.

El cuarto apartado, **3.4. Casos de uso**, se ofrece la colección de casos de uso descubiertos a partir de los requisitos definidos previamente.

Por último, el quinto apartado, **3.5. Diagramas de secuencia y de actividad del sistema**, recoge el análisis de interacciones entre los elementos internos del sistema.

3.1. Alcance del sistema

Esta sección parte de la base ofrecida en el apartado 1.2, **Objetivos,** para desarrollar en mayor profundidad las características del cliente de StarCraft que se quiere desarrollar. Dicho programa deberá presentar, al menos, las siguientes funcionalidades:

- Existirá un sistema de menús para acceder a las diferentes funcionalidades implementadas. Este menú distinguirá, por lo menos, las opciones “jugar” y “configuración”.
- Se deberá visualizar el mapa de juego a pantalla completa.
- Se deberá permitir el desplazamiento del punto de vista sobre el mapa a voluntad del jugador.
- Se deberá visualizar una miniatura del mapa global para reforzar la visión de conjunto que tiene el jugador.
- El jugador tendrá acceso a recursos que servirán para producir unidades y edificios. Estos elementos serán recogidos a partir de algún elemento sobre el mapa.
- Se deberá poder interactuar con cuatro unidades diferentes: al menos una de aire, al menos una de ataque a distancia, al menos una de ataque cuerpo a cuerpo y al menos una de tipo recolección/construcción.
- Se deberá poder interactuar con tres edificios diferentes: uno de tipo base central, uno de producción de unidades y uno de recolección de minerales.
- Los posibles comandos de las unidades serán: mover, parar, atacar, seguir, construir y recolectar. Cada comando deberá llevar a cabo la acción indicada (mover desplazará la unidad a otro punto del mapa, atacar entablará batalla con otra unidad o edificio, seguir mantendrá a la unidad cerca de otra unidad…).
- Los posibles comandos de los edificios son relativos a la producción de las unidades descritas (producir unidad 1, producir unidad 2…).
- Se deberá presentar en pantalla interfaz suficiente para que el usuario pueda acceder a las diferentes funcionalidades.
- El jugador podrá configurar el volumen de la reproducción de efectos de sonido y de música.
- El jugador podrá seleccionar el tamaño de ventana que utiliza el juego.
- Existirán tres bandos: el del jugador (amistoso), uno neutral y otro enemigo.

- El jugador podrá elegir el color que distingue al bando propio del bando enemigo.

Siempre que se respete todo lo anterior, se tendrá libertad para modelar los distintos aspectos del producto de la manera que el alumno juzgue más conveniente. Algunos criterios recomendados son crear una experiencia más dinámica, favorecer la simplicidad o unificar diferentes aspectos (configurables, de interfaz…). La única restricción sobre esto es el trabajo sobre mapas. El juego deberá permitir la definición de mapas de manera independiente del código. En este sentido se acordó la utilización de ficheros de texto. Estos archivos harán referencia a un conjunto de imágenes predefinido que será incluido en el programa en tiempo de compilación.

3.2. Requisitos de usuario

En esta sección se recogen y clasifican los requisitos de usuario que delimitan el sistema. Estos requisitos han sido extraídos a partir de diversas reuniones con el tutor y se han visto reforzados con el conocimiento obtenido durante la elaboración del capítulo 2, **Estado de la cuestión**.

Los requisitos extraídos plasman tanto lo que el sistema debe realizar como la manera en la que debe realizarlo. Siguiendo este criterio, el conjunto se divide en dos grandes bloques:

- Requisitos de capacidad: son aquellos que denotan una funcionalidad que el sistema debe proporcionar o una cualidad que debe mostrar. Son requisitos que hacen referencia a la cuestión que se desea resolver.

- Requisitos de restricción: los elementos de esta categoría indican de qué manera debe comportarse el sistema: cómo debe resolver los problemas tratados, cómo debe interactuar con el usuario… En general, estos requisitos definen las restricciones que se aplican al sistema sin menoscabo de la funcionalidad solicitada.

Al objeto de realizar una clasificación homogénea y uniforme de los requisitos de usuario se ha definido una plantilla. A continuación se presenta dicha construcción junto a la definición de cada uno de sus elementos.

RUT-NN	Título		
Prioridad		**Fuente**	
Verificabilidad		**Claridad**	
Necesidad		**Estabilidad**	
Descripción			

Tabla 1: Plantilla de definición de Requisitos de Usuario

Donde:

- RU: siglas utilizadas para abreviar "Requisito de Usuario".
- T: Tipo de requisito de usuario. Existen dos posibilidades: C para requisitos de capacidad y R para requisitos de restricción
- NN: identificador numérico del requisito.
- Prioridad: indica la importancia relativa que tiene llevar a cabo el requisito. Los posibles valores son Alta, Media y Baja.
- Fuente: indica el origen del requisito.
- Verificabilidad: indica la facilidad con que se puede verificar el cumplimiento del requisito. Los posibles valores son Alta, Media y Baja.
- Claridad: indica el grado de comprensión de lo que el requisito expresa. Los posibles valores son Alta, Media y Baja.
- Necesidad: indica el interés de los participantes en que el requisito sea cumplido. Los posibles valores son Esencial, Deseable y Opcional.
- Estabilidad: indica el grado de variabilidad del requisito a lo largo del proyecto. Los posibles valores son Alta, Media y Baja.
- Descripción: describe la funcionalidad que se desea concretar mediante la inclusión del requisito.

3.2.1. Requisitos de capacidad

Este apartado se ha dedicado a recoger la colección de requisitos de usuario de capacidad que dirigen el presente proyecto.

RUC-01	Inicio de la aplicación		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación se iniciará a través de un archivo ejecutable.		

Tabla 2: Requisito de Capacidad RUC-01

RUC-02	Menús de la aplicación		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación tendrá un menú principal, una pantalla de opciones y una pantalla de juego.		

Tabla 3: Requisito de Capacidad RUC-02

RUC-03	Estructura del menú principal		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La pantalla principal permitirá jugar, configurar opciones y salir del juego.		

Tabla 4: Requisito de Capacidad RUC-03

RUC-04	Estructura del menú de opciones		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La pantalla de opciones permitirá configurar las opciones disponibles.		

Tabla 5: Requisito de Capacidad RUC-04

RUC-05	Opciones por defecto		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación permitirá activar un conjunto de opciones por defecto		

Tabla 6: Requisito de Capacidad RUC-05

RUC-06	Guardar opciones		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación permitirá guardar las opciones de manera permanente.		

Tabla 7: Requisito de Capacidad RUC-06

RUC-07	Estructura del menú de juego		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La pantalla de juego permitirá jugar una partida.		

Tabla 8: Requisito de Capacidad RUC-07

RUC-08	Mapa de juego		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación orientará al usuario durante la partida mediante un mapa.		

Tabla 9: Requisito de Capacidad RUC-08

RUC-09	Punto de vista sobre el mapa		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación permitirá que el jugador cambie el punto de vista que tiene sobre el mapa.		

Tabla 10: Requisito de Capacidad RUC-09

RUC-10	Construcción de un bando		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación permitirá que cada bando construya sus propias unidades y edificios.		

Tabla 11: Requisito de Capacidad RUC-10

RUC-11	Destrucción de otros bandos		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación permitirá que cada bando destruya unidades y edificios de otros bandos.		

Tabla 12: Requisito de Capacidad RUC-11

RUC-12	Selección de unidades y edificios		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación permitirá que el jugador seleccione las unidades y los edificios.		

Tabla 13: Requisito de Capacidad RUC-12

RUC-13	Pausa de una partida		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación permitirá que el jugador detenga temporalmente una partida para reanudarla después.		

Tabla 14: Requisito de Capacidad RUC-13

RUC-14	Recursos de una partida		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación pondrá a disposición del jugador los recursos de gas, cristal y población.		

Tabla 15: Requisito de Capacidad RUC-14

RUC-15	Existencia de hilo musical		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación reproducirá un hilo musical.		

Tabla 16: Requisito de Capacidad RUC-15

RUC-16	Existencia de efectos de sonido		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación reproducirá efectos de sonido.		

Tabla 17: Requisito de Capacidad RUC-16

RUC-17	Entrada de usuario		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación utilizará como entradas de usuario el ratón y el teclado.		

Tabla 18: Requisito de Capacidad RUC-17

3.2.2. Requisitos de restricción

En esta sección se recogen los requisitos de usuario de restricción que dan forma a la aplicación. Este conjunto de datos debe ser interpretado de manera complementaria a los requisitos de usuario de capacidad.

RUR-01	Bandos		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación implementará tres bandos: amigo, neutral y enemigo.		

Tabla 19: Requisito de Restricción RUR-01

RUR-02	Composición de una partida		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Una partida estará compuesta por un mapa, el bando amigo y uno o varios bandos más, cada uno con al menos una unidad constructora.		

Tabla 20: Requisito de Restricción RUR-02

RUR-03	Composición de un mapa		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Un mapa estará compuesto por casillas atravesables por unidades de tierra y casillas no atravesables por unidades de tierra.		

Tabla 21: Requisito de Restricción RUR-03

RUR-04	Existencia de mapa y mapa en miniatura		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación deberá mostrar el punto de vista sobre el mapa a pantalla completa y una visión global a escala del mapa entero.		

Tabla 22: Requisito de Restricción RUR-04

RUR-05	Tipos de unidades		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Deberá haber al menos una unidad aérea, al menos una de ataque a distancia, al menos una de ataque cuerpo a cuerpo y al menos una capaz de recolectar recursos y construir edificios.		

Tabla 23: Requisito de Restricción RUR-05

RUR-06	Tipos de edificios		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Deberá haber al menos un edificio de tipo base central, uno de producción de unidades y uno de recolección de minerales.		

Tabla 24: Requisito de Restricción RUR-06

RUR-07	Proceso de recolección		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La recolección de recursos será realizada por una unidad recolectora, y será llevada a cabo mediante viajes entre el edificio tipo base central y el edificio de recolección de minerales.		

Tabla 25: Requisito de Restricción RUR-07

RUR-08	Órdenes de las unidades		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Las órdenes que pueden recibir las unidades son: mover, parar, atacar, seguir, construir y recolectar.		

Tabla 26: Requisito de Restricción RUR-08

RUR-09	Órdenes de los edificios		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Las órdenes que pueden recibir los edificios son: construir una unidad concreta.		

Tabla 27: Requisito de Restricción RUR-09

RUR-10	Selección de unidades propias		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El jugador podrá seleccionar sus unidades y edificios para darles órdenes o consultar la información asociada.		

Tabla 28: Requisito de Restricción RUR-10

RUR-11	Selección de unidades ajenas		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El jugador podrá interactuar con unidades y edificios de otros bandos para simular la existencia de otra aplicación cliente.		

Tabla 29: Requisito de Restricción RUR-11

RUR-12	Construcción de unidades		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Las unidades de un bando serán construidas en edificios de su mismo bando.		

Tabla 30: Requisito de Restricción RUR-12

RUR-13	Construcción de edificios		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Las unidades de un bando serán capaces de construir edificios de su mismo bando.		

Tabla 31: Requisito de Restricción RUR-13

RUR-14	Destrucción de otros bandos		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Las unidades de un bando serán capaces de destruir unidades y edificios de otros bandos.		

Tabla 32: Requisito de Restricción RUR-14

RUR-15	Recursos constructivos		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Los recursos que servirán para construir unidades o edificios serán el cristal y el gas.		

Tabla 33: Requisito de Restricción RUR-15

RUR-16	Recurso limitante		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El recurso que limitará el número de unidades que es capaz de controlar un jugador será la población.		

Tabla 34: Requisito de Restricción RUR-16

RUR-17	Uso de recursos local		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El uso de recursos se implementará para el bando amigo ya que la aplicación es solamente la parte cliente.		

Tabla 35: Requisito de Restricción RUR-17

RUR-18	Idioma de la aplicación		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El idioma de la aplicación será el inglés.		

Tabla 36: Requisito de Restricción RUR-18

RUR-19	Opciones configurables		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Las opciones configurables serán volumen de audio, volumen de efectos de sonido, color del bando propio, color del bando enemigo, resolución de pantalla y velocidad de desplazamiento del punto de vista sobre el mapa.		

Tabla 37: Requisito de Restricción RUR-19

RUR-20	Variación del hilo musical		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El hilo musical variará dependiendo de la pantalla en la que se encuentre el usuario.		

Tabla 38: Requisito de Restricción RUR-20

RUR-21	Volumen del hilo musical		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El volumen del hilo musical podrá ser configurado en todo momento.		

Tabla 39: Requisito de Restricción RUR-21

RUR-22	Variación de los efectos de sonido		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Los efectos de sonido mostrarán la reacción de la aplicación ante las entradas de usuario (selección de menú, orden de unidad...).		

Tabla 40: Requisito de Restricción RUR-22

RUR-23	Volumen de los efectos de sonido		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El volumen de los efectos de sonido podrá ser configurado en todo momento.		

Tabla 41: Requisito de Restricción RUR-23

RUR-24	Interacción con unidades y edificios		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación permitirá acceder a las órdenes e información de unidades y edificios directamente desde la pantalla de juego sin pantallas intermedias.		

Tabla 42: Requisito de Restricción RUR-24

RUR-25	Acción por defecto		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación reconocerá una entrada asociada a la acción por defecto de una unidad o edificio.		

Tabla 43: Requisito de Restricción RUR-25

RUR-26	Diseño de interfaz		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Baja	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación dispondrá los elementos de interfaz bloqueando el mínimo espacio de pantalla necesario siguiendo un diseño sencillo y, en la medida de lo posible, vistoso.		

Tabla 44: Requisito de Restricción RUR-26

RUR-27	Manual de usuario		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Existirá un manual de usuario.		

Tabla 45: Requisito de Restricción RUR-27

RUR-28	Sistema objetivo		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación será desplegada en Windows XP, Windows Vista o Windows 7.		

Tabla 46: Requisito de Restricción RUR-28

RUR-29	Portabilidad		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Baja	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación deberá minimizar el código exclusivo de PC para facilitar su portabilidad.		

Tabla 47: Requisito de Restricción RUR-29

RUR-30	Tiempo de carga		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación deberá tardar 10 segundos como máximo en cargar el contenido de una pantalla cualquiera.		

Tabla 48: Requisito de Restricción RUR-30

RUR-31	Formato de archivos		
Prioridad	Alta	**Fuente**	Entrevistas
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El formato de ficheros utilizados deberá ser BMP para imágenes, MP3 para música, WAV para efectos de sonido, AVI para vídeos y XML para texto.		

Tabla 49: Requisito de Restricción RUR-31

3.3. Requisitos de software

En esta sección se identifican, clasifican y catalogan los distintos requisitos de software que dirigen el desarrollo del proyecto. Estas necesidades han sido detectadas a través de los requisitos de usuario, ya expuestos, y mediante los casos de uso de la aplicación, recogidos en la siguiente sección del documento.

Los requisitos de software deben recoger tanto lo que el sistema es capaz de hacer como la forma en que lo hace. También deben mostrar las propiedades o valores concretos que, a grandes rasgos, definen el programa. En base a esto se tienen dos grandes grupos:

- Requisitos funcionales: especifican la actividad que tiene que realizar el sistema.
- Requisitos no funcionales: concretan la manera en que se debe llevar a cabo el cometido diseñado.

A continuación se presenta la plantilla de definición de requisitos de usuario junto a la definición de cada uno de sus elementos.

RST-NN	Título		
Prioridad		**Fuente**	
Verificabilidad		**Claridad**	
Necesidad		**Estabilidad**	
Descripción			

Tabla 50: Plantilla de definición de Requisitos de Software

Donde:

- RS: siglas utilizadas para abreviar “Requisito de Software”.
- T: Tipo de requisito de usuario. Existen diversas posibilidades:
  - F: denota un requisito funcional.
  - I: denota un requisito no funcional relativo a la interfaz del usuario con la aplicación.
  - D: denota un requisito no funcional relativo a la documentación (ayuda, manual de usuario…).
  - P: denota un requisito no funcional relativo a la portabilidad de la aplicación hacia otros sistemas diferentes al de despliegue.

    - O: denota un requisito no funcional relativo al entorno de ejecución en el que debe funcionar el sistema.
    - R: denota un requisito no funcional relativo al rendimiento del sistema.
    - S: denota un requisito no funcional relativo a la seguridad ante fallos (recuperación del sistema frente a errores propios). Este tipo de requisito se enumera, aunque no se han encontrado necesidades de este tipo.
- NN: identificador numérico del requisito.
- Prioridad: indica la importancia relativa que tiene llevar a cabo el requisito. Los posibles valores son Alta, Media y Baja.
- Fuente: indica el origen del requisito.
- Verificabilidad: indica la facilidad con que se puede verificar el cumplimiento del requisito. Los posibles valores son Alta, Media y Baja.
- Claridad: indica el grado de comprensión de lo que el requisito expresa. Los posibles valores son Alta, Media y Baja.
- Necesidad: indica el interés de los participantes en que el requisito sea cumplido. Los posibles valores son Esencial, Deseable y Opcional.
- Estabilidad: indica el grado de variabilidad del requisito a lo largo del proyecto. Los posibles valores son Alta, Media y Baja.
- Descripción: describe la funcionalidad que se desea concretar mediante la inclusión del requisito.

3.3.1. Requisitos funcionales

Este apartado recoge los requisitos de software funcionales que se han detectado. Se encuentran agrupados por la funcionalidad a la que hacen referencia.

RSF-01	Inicio de la aplicación con ejecutable		
Prioridad	Alta	**Fuente**	RUC-01
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación se iniciará mediante la ejecución del fichero .EXE generado al compilar el código fuente.		

Tabla 51: Requisito de Restricción RSF-01

RSF-02	Mostrar pantalla inicial		
Prioridad	Alta	**Fuente**	RUC-02
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Una vez iniciada la aplicación, se mostrará la Pantalla de Menú Principal.		

Tabla 52: Requisito de Restricción RSF-02

RSF-03	Opciones del menú principal		
Prioridad	Alta	**Fuente**	RUC-03
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La Pantalla de Menú Principal permitirá acceder a la Pantalla de Juego, a la Pantalla de Configuración y Salir.		

Tabla 53: Requisito de Restricción RSF-03

RSF-04	Función de la pantalla de configuración		
Prioridad	Alta	**Fuente**	RUC-04
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La Pantalla de Configuración permitirá seleccionar los valores asociados a las opciones disponibles.		

Tabla 54: Requisito de Restricción RSF-04

RSF-05	Guardar cambios, cancelar cambios y poner valores por defecto		
Prioridad	Alta	**Fuente**	RUC-05, RUC-06
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La Pantalla de Configuración tendrá una opción explícita para guardar los cambios, para cancelar los cambios y para volver a los valores por defecto.		

Tabla 55: Requisito de Restricción RSF-05

RSF-06	Opciones disponibles		
Prioridad	Alta	**Fuente**	RUR-19
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Las opciones disponibles serán: color del bando amigo, color del bando enemigo, resolución de pantalla, velocidad de desplazamiento del mapa, volumen de la música, existencia de hilo musical, volumen de los efectos de sonido, existencia de efectos de sonido.		

Tabla 56: Requisito de Restricción RSF-06

RSF-07	Guardar cambios en XML		
Prioridad	Alta	**Fuente**	RUC-06
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación guardará los cambios de manera permanente en el formato XML.		

Tabla 57: Requisito de Restricción RSF-07

RSF-08	Valores por defecto		
Prioridad	Alta	**Fuente**	RUC-05
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Los valores por defecto serán: color del bando amigo azul, color del bando enemigo rojo, resolución de pantalla 800x600, velocidad de desplazamiento del mapa 2, volumen de la música 5, música activada, volumen de los efectos de sonido 5, efectos de sonido activados.		

Tabla 58: Requisito de Restricción RSF-08

RSF-09	Opción Salir		
Prioridad	Alta	**Fuente**	RUC-03
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La opción Salir del Menú Principal terminará la ejecución del programa sin esperas de ningún tipo.		

Tabla 59: Requisito de Restricción RSF-09

RSF-10	Juego con Pausa		
Prioridad	Alta	**Fuente**	RUC-13
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La Pantalla de Juego podrá dar paso a una Pantalla de Pausa para detener momentáneamente la partida.		

Tabla 60: Requisito de Restricción RSF-10

RSF-11	Contenido de la pantalla de juego		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La Pantalla de Juego presentará una barra de control, el mapa, las unidades, los edificios, los recursos, el volumen de la música y el volumen de los efectos de sonido.		

Tabla 61: Requisito de Restricción RSF-11

RSF-12	Contenido de la barra de control		
Prioridad	Alta	**Fuente**	RUR-24
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La barra de control mostrará un mapa en miniatura, la información asociada a la unidad seleccionada y las órdenes que se puedan dar.		

Tabla 62: Requisito de Restricción RSF-12

RSF-13	Contenido del mapa en miniatura		
Prioridad	Alta	**Fuente**	RUR-04
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El mapa en miniatura tendrá toda la información del mapa global a escala.		

Tabla 63: Requisito de Restricción RSF-13

RSF-14	Información adicional ilustrada		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La información mostrada estará ilustrada con una miniatura de la unidad o del edificio.		

Tabla 64: Requisito de Restricción RSF-14

RSF-15	Órdenes ilustradas		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Las órdenes estarán representadas por ilustraciones características.		

Tabla 65: Requisito de Restricción RSF-15

RSF-16	Órdenes con sonido asociado		
Prioridad	Alta	**Fuente**	RUC-16
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se reproducirá un sonido característico al dar una orden.		

Tabla 66: Requisito de Restricción RSF-16

RSF-17	Contenido del mapa		
Prioridad	Alta	**Fuente**	RUC-08
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El mapa será un conjunto de casillas de dimensiones arbitrarias (p.ej.: 40x40 casillas, 50x25...).		

Tabla 67: Requisito de Restricción RSF-17

RSF-18	Casillas del mapa		
Prioridad	Alta	**Fuente**	RUR-03
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El mapa tendrá casillas bloqueantes y casillas pasables.		

Tabla 68: Requisito de Restricción RSF-18

RSF-19	Navegación por el mapa		
Prioridad	Alta	**Fuente**	RUC-09
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El punto de vista sobre el mapa podrá ser cambiado por el jugador de forma directa y gradual.		

Tabla 69: Requisito de Restricción RSF-19

RSF-20	Selección de unidades		
Prioridad	Alta	**Fuente**	RUC-12
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se podrá seleccionar y deseleccionar una unidad o un grupo de ellas.		

Tabla 70: Requisito de Restricción RSF-20

RSF-21	Dar órdenes a unidades		
Prioridad	Alta	**Fuente**	RUR-10, RUR-11
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El usuario podrá dar las órdenes pertinentes a las unidades.		

Tabla 71: Requisito de Restricción RSF-21

RSF-22	Destrucción de otros bandos		
Prioridad	Alta	**Fuente**	RUR-01
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Las unidades podrán destruir otras unidades o edificios de distinto bando.		

Tabla 72: Requisito de Restricción RSF-22

RSF-23	Comando mover		
Prioridad	Alta	**Fuente**	RUR-08
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Una unidad podrá recibir el comando mover, teniendo entonces que situarse, si es posible, en la posición indicada.		

Tabla 73: Requisito de Restricción RSF-23

RSF-24	Comando seguir		
Prioridad	Alta	**Fuente**	RUR-08
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Una unidad podrá recibir el comando seguir a una unidad del mismo bando, teniendo entonces que situarse lo más cerca posible de la unidad seguida en todo momento.		

Tabla 74: Requisito de Restricción RSF-24

RSF-25	Comando atacar		
Prioridad	Alta	**Fuente**	RUC-11, RUR-14
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Una unidad podrá recibir el comando atacar, teniendo entonces que acercarse a su objetivo y comenzar a disparar.		

Tabla 75: Requisito de Restricción RSF-25

RSF-26	Comando construir		
Prioridad	Alta	**Fuente**	RUC-10, RUR-13
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Una unidad constructora podrá recibir el comando construir, teniendo entonces que comenzar la construcción del edificio señalado.		

Tabla 76: Requisito de Restricción RSF-26

RSF-27	Comando parar		
Prioridad	Alta	**Fuente**	RUR-08
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Una unidad podrá recibir el comando parar, teniendo entonces que detener cualquier actividad que esté realizando.		

Tabla 77: Requisito de Restricción RSF-27

RSF-28	Comando recolectar		
Prioridad	Alta	**Fuente**	RUR-08
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Una unidad recolectora podrá recibir el comando recolectar, teniendo que realizar a partir de entonces viajes periódicos entre un edificio base y un edificio recolector de su mismo bando.		

Tabla 78: Requisito de Restricción RSF-28

RSF-29	Tipos de unidad existentes		
Prioridad	Alta	**Fuente**	RUR-05
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Habrá al menos una unidad aérea, al menos una de ataque a distancia, al menos una de ataque cuerpo a cuerpo y al menos una capaz de recolectar recursos y construir edificios.		

Tabla 79: Requisito de Restricción RSF-29

RSF-30	Tipos de edificio existentes		
Prioridad	Alta	**Fuente**	RUR-06
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Habrá al menos un edificio de tipo base central, uno de producción de unidades y uno de recolección de minerales.		

Tabla 80: Requisito de Restricción RSF-30

RSF-31	Orden por defecto		
Prioridad	Alta	**Fuente**	RUR-25
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación reconocerá una orden por defecto por cada unidad.		

Tabla 81: Requisito de Restricción RSF-31

RSF-32	Selección de edificios		
Prioridad	Alta	**Fuente**	RUC-12
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se podrá seleccionar y deseleccionar un edificio.		

Tabla 82: Requisito de Restricción RSF-32

RSF-33	Órdenes a edificios		
Prioridad	Alta	**Fuente**	RUR-10, RUR-11
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El usuario podrá dar las órdenes pertinentes a los edificios.		

Tabla 83: Requisito de Restricción RSF-33

RSF-34	Comando producir unidad		
Prioridad	Alta	**Fuente**	RUR-09, RUR-12
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Un edificio podrá recibir la orden de producir una unidad, teniendo que comenzar si es posible la producción de dicha unidad.		

Tabla 84: Requisito de Restricción RSF-34

RSF-35	Recursos existentes		
Prioridad	Alta	**Fuente**	RUC-14
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Los recursos existentes serán gas, cristal y población.		

Tabla 85: Requisito de Restricción RSF-35

RSF-36	Recursos actuales en pantalla de juego		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se mostrarán en la Pantalla de Juego los recursos actuales en todo momento.		

Tabla 86: Requisito de Restricción RSF-36

RSF-37	Recursos máximos en pantalla de juego		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se mostrarán en la Pantalla de Juego los recursos máximos acumulables en todo momento.		

Tabla 87: Requisito de Restricción RSF-37

RSF-38	Recursos para unidades y edificios		
Prioridad	Alta	**Fuente**	RUR-15, RUR-17
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se impondrá un coste en gas y/o en cristal a las unidades y edificios construidos por el jugador.		

Tabla 88: Requisito de Restricción RSF-38

RSF-39	Población para unidades		
Prioridad	Alta	**Fuente**	RUR-16, RUR-17
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se limitará el número de unidades construidas por el jugador mediante el recurso población.		

Tabla 89: Requisito de Restricción RSF-39

RSF-40	Proceso de recolección		
Prioridad	Alta	**Fuente**	RUR-07
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Los recursos de gas y cristal aumentarán en proporción a la actividad de las unidades recolectoras.		

Tabla 90: Requisito de Restricción RSF-40

RSF-41	Hilo musical		
Prioridad	Alta	**Fuente**	RUC-15, RUR-20
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Existirá un hilo musical dependiente de la pantalla en la que se encuentre el usuario.		

Tabla 91: Requisito de Restricción RSF-41

RSF-42	Volumen de música global		
Prioridad	Alta	**Fuente**	RUR-21
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El volumen de la música podrá ser configurado en todo momento.		

Tabla 92: Requisito de Restricción RSF-42

RSF-43	Silenciar música		
Prioridad	Alta	**Fuente**	RUR-21
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La música podrá ser silenciada en todo momento.		

Tabla 93: Requisito de Restricción RSF-43

RSF-44	Volumen de efectos de sonido global		
Prioridad	Alta	**Fuente**	RUR-23
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El volumen de los efectos de sonido podrá ser configurado en todo momento.		

Tabla 94: Requisito de Restricción RSF-44

RSF-45	Silenciar efectos de sonido		
Prioridad	Alta	**Fuente**	RUR-23
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Los efectos de sonido podrán ser silenciados en todo momento.		

Tabla 95: Requisito de Restricción RSF-45

RSF-46	Combates ilustrados		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Será necesario ilustrar los combates en pantalla (disparos realizados, disparos recibidos y muerte).		

Tabla 96: Requisito de Restricción RSF-46

RSF-47	Combates con sonido		
Prioridad	Alta	**Fuente**	RUC-16, RUR-22
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Será necesario dotar de sonido a los combates (disparo realizado y muerte).		

Tabla 97: Requisito de Restricción RSF-47

RSF-48	Comienzo de una partida		
Prioridad	Alta	**Fuente**	RUR-02
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Al comenzar una partida, el jugador y al menos otro bando tendrán por lo menos una unidad constructora y recursos suficientes para empezar a jugar.		

Tabla 98: Requisito de Restricción RSF-48

RSF-49	Control de teclado y ratón		
Prioridad	Alta	**Fuente**	RUC-17
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación debe ser capaz de controlar el teclado y el ratón para realizar las operaciones mencionadas (seleccionar, deseleccionar, dar orden, elegir una entrada de menú, navegar por el mapa).		

Tabla 99: Requisito de Restricción RSF-49

3.3.2. Requisitos no funcionales

Esta sección se dedica a mostrar la colección de requisitos de software no funcionales, agrupados por el tipo concreto: interfaz, documentación, operación… Estos datos son complementarios a los requisitos de software previamente expuestos.

RSI-01	Claridad y estilo en menú principal		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La Pantalla de Menú Principal mostrará con claridad y estilo propio el nombre del juego.		

Tabla 100: Requisito de Restricción RSI-01

RSI-02	Color del bando amigo		
Prioridad	Alta	**Fuente**	RUR-01
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se podrá cambiar el color del bando amigo eligiendo entre los valores disponibles: azul, rojo, morado.		

Tabla 101: Requisito de Restricción RSI-02

RSI-03	Color del bando enemigo		
Prioridad	Alta	**Fuente**	RUR-01
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se podrá cambiar el color del bando enemigo eligiendo entre los valores disponibles: azul, rojo, morado.		

Tabla 102: Requisito de Restricción RSI-03

RSI-04	Volumen de música configurable		
Prioridad	Alta	**Fuente**	RUR-19
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se podrá elegir el volumen de la música con valores crecientes de 1 a 10.		

Tabla 103: Requisito de Restricción RSI-04

RSI-05	Presencia de música configurable		
Prioridad	Alta	**Fuente**	RUR-19
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se podrá elegir la presencia o ausencia de la música.		

Tabla 104: Requisito de Restricción RSI-05

RSI-06	Volumen de efectos de sonido configurable		
Prioridad	Alta	**Fuente**	RUR-19
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se podrá elegir el volumen de los efectos de sonido con valores crecientes de 1 a 10.		

Tabla 105: Requisito de Restricción RSI-06

RSI-07	Presencia de efectos de sonido configurable		
Prioridad	Alta	**Fuente**	RUR-19
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se podrá elegir la presencia o ausencia de los efectos de sonido.		

Tabla 106: Requisito de Restricción RSI-07

RSI-08	Resolución de pantalla configurable		
Prioridad	Alta	**Fuente**	RUR-19
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se podrá elegir la resolución de pantalla utilizada entre un conjunto de opciones soportadas por el sistema.		

Tabla 107: Requisito de Restricción RSI-08

RSI-09	Velocidad de desplazamiento del mapa configurable		
Prioridad	Alta	**Fuente**	RUR-19
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se podrá elegir la velocidad de desplazamiento del mapa con valores crecientes de 1 a 10.		

Tabla 108: Requisito de Restricción RSI-09

RSI-10	Concepto de pantalla de juego		
Prioridad	Alta	**Fuente**	RUC-07
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Entrar en la Pantalla de Juego será equivalente a empezar una partida		

Tabla 109: Requisito de Restricción RSI-10

RSI-11	Características del mapa en miniatura		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El mapa en miniatura mostrará claramente el punto de vista que el usuario tiene sobre el mapa global.		

Tabla 110: Requisito de Restricción RSI-11

RSI-12	Navegación a través del mapa en miniatura		
Prioridad	Alta	**Fuente**	RUC-08
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El mapa en miniatura permitirá cambiar el punto de vista sobre el mapa en cualquier momento.		

Tabla 111: Requisito de Restricción RSI-12

RSI-13	Contenido de la información adicional		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La información mostrada será el mayor conjunto posible a partir de los siguientes conceptos: ataque, velocidad, coste de cristal, coste de gas, coste de tiempo, vida.		

Tabla 112: Requisito de Restricción RSI-13

RSI-14	Mapa en función de archivo base		
Prioridad	Alta	**Fuente**	RUC-08
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El mapa estará definido en función del archivo que define el conjunto de casillas posibles.		

Tabla 113: Requisito de Restricción RSI-14

RSI-15	Unidades y edificios sobre el mapa		
Prioridad	Alta	**Fuente**	RUR-02
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Las unidades y los edificios se mostrarán sobre el mapa.		

Tabla 114: Requisito de Restricción RSI-15

RSI-16	Selección ilustrada de unidad		
Prioridad	Alta	**Fuente**	RUR-01
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La selección de una unidad estará ilustrada según el bando de la unidad y tendrá asociado un sonido.		

Tabla 115: Requisito de Restricción RSI-16

RSI-17	Deselección ilustrada de unidad		
Prioridad	Alta	**Fuente**	RUR-01
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La deselección de una unidad estará ilustrada según el bando de la unidad.		

Tabla 116: Requisito de Restricción RSI-17

RSI-18	Selección ilustrada de edificio		
Prioridad	Alta	**Fuente**	RUR-01
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La selección de un edificio estará ilustrada según el bando de la unidad y tendrá asociado un sonido.		

Tabla 117: Requisito de Restricción RSI-18

RSI-19	Deselección ilustrada de edificio		
Prioridad	Alta	**Fuente**	RUR-01
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La deselección de un edificio estará ilustrada según el bando de la unidad.		

Tabla 118: Requisito de Restricción RSI-19

RSI-20	Sonidos en respuesta a órdenes		
Prioridad	Alta	**Fuente**	RUC-16
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Los sonidos asociados a las órdenes serán diferentes para órdenes posibles y órdenes imposibles.		

Tabla 119: Requisito de Restricción RSI-20

RSI-21	Aplicación en inglés		
Prioridad	Alta	**Fuente**	RUR-18
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación estará escrita totalmente en inglés: tanto el código como los aspectos de interfaz.		

Tabla 120: Requisito de Restricción RSI-21

RSI-22	Criterios de diseño de interfaz		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La interfaz seguirá los principios de diseño observados en los requisitos de usuario: sencillez, que ocupe poco espacio en pantalla y que sea fácil de aprender a utilizar.		

Tabla 121: Requisito de Restricción RSI-22

RSI-23	Formato de archivo de ilustración		
Prioridad	Alta	**Fuente**	RUR-31
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Los archivos de ilustración utilizados estarán almacenados en formato PNG.		

Tabla 122: Requisito de Restricción RSI-23

RSI-24	Formato de archivo de efecto de sonido		
Prioridad	Alta	**Fuente**	RUR-31
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Los archivos de sonido utilizados estarán almacenados en formato WAV.		

Tabla 123: Requisito de Restricción RSI-24

RSI-25	Formato de archivo de mapa		
Prioridad	Alta	**Fuente**	RUR-31
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El mapa será leído desde un fichero XML.		

Tabla 124: Requisito de Restricción RSI-25

RSI-26	Formato de archivo de música		
Prioridad	Alta	**Fuente**	RUR-31
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Los archivos de música utilizados estarán almacenados en formato MP3.		

Tabla 125: Requisito de Restricción RSI-26

RSI-27	Formato de conjunto de casillas		
Prioridad	Alta	**Fuente**	RUR-31
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El conjunto de casillas posibles para definir el mapa estará almacenado en un fichero BMP.		

Tabla 126: Requisito de Restricción RSI-27

RSR-28	Respuesta ante el usuario		
Prioridad	Alta	**Fuente**	RUR-30
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La información mostrada se actualizará en tiempo real.		

Tabla 127: Requisito de Restricción RSR-28

RSP-29	Desarrollo portable		
Prioridad	Alta	**Fuente**	RUR-29
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se utilizarán Visual Studio .NET 2008 (C#) y XNA 3.1 como herramientas principales de desarrollo.		

Tabla 128: Requisito de Restricción RSP-29

RSP-30	Límite de espacio en disco duro		
Prioridad	Alta	**Fuente**	RUR-29
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	El sistema listo para despliegue no superará los 200 megabytes de espacio en disco duro.		

Tabla 129: Requisito de Restricción RSP-30

RSO-31	Equipo de desarrollo		
Prioridad	Alta	**Fuente**	RUR-28
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Para desarrollar el juego será necesario un equipo con Windows XP Service Pack 3 o superior, DirectX 10 o superior, XNA Game Studio 3.1 o superior, Visual Studio .NET 2008, 1GB (Windows XP) o 2GB (Windows Vista, Windows 7) de RAM, 100MB de espacio libre en disco duro, CPU de un núcleo a 1.5GHz o superior, ratón, teclado, altavoces (opcional).		

Tabla 130: Requisito de Restricción RSO-31

RSO-32	Equipo de ejecución		
Prioridad	Alta	**Fuente**	RUR-28
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Para ejecutar del juego será necesario un equipo con Windows XP Service Pack 3 o superior, DirectX 10 o superior, XNA Game Studio 3.1 o superior, 1GB (Windows XP) o 2GB (Windows Vista, Windows 7) de RAM, 100MB de espacio libre en disco duro, CPU de un núcleo a 1.5GHz o superior, ratón, teclado, altavoces (opcional).		

Tabla 131: Requisito de Restricción RSO-32

RSO-33	Control de tiempos de carga		
Prioridad	Alta	**Fuente**	RUR-30
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	La aplicación controlará mediante una Pantalla de Carga que no se exceda el tiempo estipulado para los instantes de carga de datos.		

Tabla 132: Requisito de Restricción RSO-33

RSO-34	Comportamiento frente a errores de definición de mapa		
Prioridad	Alta	**Fuente**	RUR-26
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Un fichero de mapa no válido no producirá comportamiento anómalo por parte de la aplicación (pantallas de error, salida inmediata...).		

Tabla 133: Requisito de Restricción RSO-34

RSD-35	Manual de usuario		
Prioridad	Alta	**Fuente**	RUR-27
Verificabilidad	Alta	**Claridad**	Alta
Necesidad	Esencial	**Estabilidad**	Alta
Descripción	Se creará un manual de usuario, adjuntándolo la documentación del proyecto.		

Tabla 134: Requisito de Restricción RSD-35

3.4. Casos de uso

Este apartado recoge el trabajo realizado acerca de los Casos de Uso. En primer lugar se ofrece la reflexión realizada acerca de la herramienta: para qué sirve, por qué es necesaria... Después se especifica individualmente cada uno de los Casos de Uso descubiertos en el sistema.

3.4.1. Diagrama de casos de uso

Los Casos de Uso son un instrumento que ayuda a mejorar la comprensión y especificación de la actividad que realiza la aplicación. Tienen su base en los requisitos de usuario y software, pero permiten conseguir un punto de vista diferente. En lugar de detallar qué se debe hacer o cómo, los Casos de Uso definen el proceso necesario para conseguir lo que detallan los requisitos. En este sentido, los Casos de Uso tienen una componente dinámica radicalmente diferente y muy útil. Por tanto, el modelado de Casos de Uso es el instrumento adecuado para identificar más funcionalidades del videojuego.

Los Casos de Uso están diseñados para especificar las respuestas del sistema a la interacción de agentes externos. Dada la naturaleza de un videojuego y las funcionalidades que posee, se ha detectado un único actor: el Jugador. En algunos Casos parecerá posible que el actor sea otro (el Sistema, la Aplicación, una Unidad...), pero no se trata de acciones autónomas: siempre son reacciones frente al comportamiento del usuario. A modo de ejemplo se propone el siguiente: un edificio está produciendo una unidad. Cuando el proceso termina, la nueva unidad se desplaza hasta su posición inicial. Este movimiento es iniciado indirectamente por el Jugador, ya que la unidad está programada para situarse en un lugar del mapa al que el usuario pueda acceder.

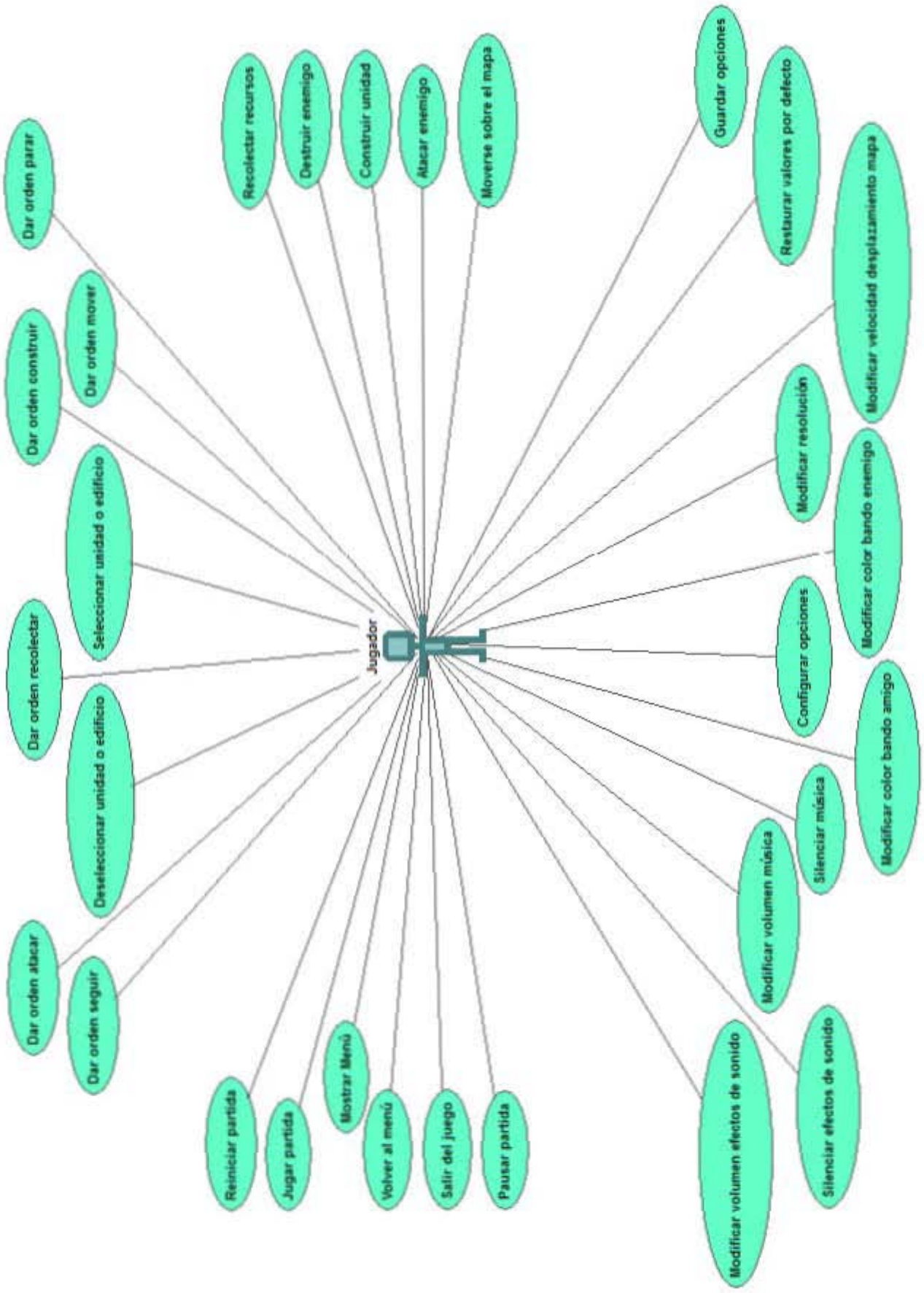

Figura 58. Diagrama de Casos de Uso

3.4.2. Especificación textual de los casos de uso

En esta sección se recoge el catálogo completo de Casos de Uso descubiertos para el presente proyecto. A continuación se presenta el formato utilizado para describir cada Caso individual.

CU-NN	Título
Actores	
Objetivo	
Precondiciones	
Postcondiciones	
Escenario básico	

Tabla 135: Plantilla de definición de Caso de Uso

Donde:

- CU: siglas que abrevian la expresión "Caso de Uso".
- NN: número secuencial que distingue cada Caso de Uso individual.
- Título: nombre corto que resume el propósito del Caso de Uso.
- Actores: entidades que intervienen directamente en la correcta realización del Caso de Uso.
- Objetivo: acción que se desea conseguir mediante la aplicación del Caso de Uso.
- Precondiciones: condiciones necesarias para comenzar el Caso de Uso.
- Postcondiciones: condiciones nuevas del sistema tras la aplicación del Caso de Uso.
- Escenario básico: breve descripción de los pasos necesarios para llevar a cabo el Caso de Uso.

A continuación se presenta el catálogo de Casos de Uso definidos en el desarrollo de la aplicación que ocupa el presente proyecto. El orden seguido para presentarlos es, según la Figura 58, el sentido horario.

CU-01	Dar orden seguir
Actores	Jugador
Objetivo	Que una unidad permanezca próxima a otra de su mismo bando.
Precondiciones	Haber seleccionado la unidad que debe seguir a otra.
Postcondiciones	La unidad que ha recibido la orden de seguir a otra permanece próxima a esta otra unidad.
Escenario básico	1. El jugador seleccionará una unidad. 2. El jugador activará la acción por defecto sobre otra unidad de su mismo bando.

Tabla 136: Caso de Uso CU-01

CU-02	Dar orden atacar
Actores	Jugador
Objetivo	Que una unidad entable combate con otra unidad o edificio.
Precondiciones	Haber seleccionado la unidad que debe atacar a otra.
Postcondiciones	La unidad que ha recibido la orden de atacar a otra dispara periódicamente a esta otra unidad.
Escenario básico	1. El jugador seleccionará una unidad. 2. El jugador activará la acción por defecto sobre una unidad de otro bando.

Tabla 137: Caso de Uso CU-02

CU-03	Deseleccionar unidad o edificio
Actores	Jugador
Objetivo	Que una unidad o edificio deje de estar seleccionado/a.
Precondiciones	Haber seleccionado la unidad o edificio.
Postcondiciones	La unidad o edificio no está seleccionado/a.
Escenario básico	1. El jugador seleccionará un espacio vacío del mapa u otra unidad.

Tabla 138: Caso de Uso CU-03

CU-04	Dar orden recolectar
Actores	Jugador
Objetivo	Que una unidad comience el proceso de recolección.
Precondiciones	Haber seleccionado unidad de tipo recolección.
Postcondiciones	La unidad realiza viajes periódicos entre un edificio de tipo base y un edificio de tipo recolección, ambos de su mismo bando, aumentando los recursos disponibles a cada viaje completo.
Escenario básico	1. El jugador activará la opción por defecto sobre el edificio tipo recolección que desee utilizar.

Tabla 139: Caso de Uso CU-04

CU-05	Seleccionar unidad o edificio
Actores	Jugador
Objetivo	Que una unidad o edificio pase a estar seleccionado/a.
Precondiciones	Que la unidad no esté seleccionada.
Postcondiciones	La unidad ilustrará el hecho de que está seleccionada.
Escenario básico	1. El jugador hará clic sobre la unidad deseada.

Tabla 140: Caso de Uso CU-05

CU-06	Dar orden construir
Actores	Jugador
Objetivo	Que una unidad construya un edificio.
Precondiciones	Haber seleccionado una unidad de tipo constructor.
Postcondiciones	La unidad, si es posible, comenzará a construir el edificio.
Escenario básico	1. El jugador activará la opción deseada de la barra de control.

Tabla 141: Caso de Uso CU-06

CU-07	Dar orden mover
Actores	Jugador
Objetivo	Que la unidad se desplace a un punto del mapa.
Precondiciones	Haber seleccionado una unidad.
Postcondiciones	La unidad, si es posible, comenzará a avanzar hacia la posición indicada.
Escenario básico	1. El jugador activará la opción por defecto de la uniad en un lugar vacío del mapa.

Tabla 142: Caso de Uso CU-07

CU-08	Dar orden parar
Actores	Jugador
Objetivo	Que la unidad detenga toda actividad.
Precondiciones	Haber seleccionado la unidad.
Postcondiciones	La unidad está quieta sobre el mapa.
Escenario básico	1. El jugador activará la opción deseada de la barra de control.

Tabla 143: Caso de Uso CU-08

CU-09	Recolectar recursos
Actores	Jugador
Objetivo	Que los recursos del jugador aumenten.
Precondiciones	La unidad recolectora debe haber pasado por un edificio de tipo recolección.
Postcondiciones	Los recursos han aumentado en una cantidad leve.
Escenario básico	1. La unidad entrará en el edificio tipo base. 2. La unidad descargará los recursos que transporte.

Tabla 144: Caso de Uso CU-09

CU-10	Destruir enemigo
Actores	Jugador
Objetivo	Que una unidad o un edificio enemigo deje de existir.
Precondiciones	Haber dado la orden de ataque contra el objetivo.
Postcondiciones	La unidad objetivo ha dejado de existir.
Escenario básico	1. La unidad se acercará a su objetivo. 2. La unidad atacará periódicamente a su objetivo.

Tabla 145: Caso de Uso CU-10

CU-11	Construir unidad
Actores	Jugador
Objetivo	Que el jugador disponga de una unidad más.
Precondiciones	Haber ordenado al edificio que construya una unidad.
Postcondiciones	La unidad aparece junto al edificio.
Escenario básico	1. El edificio comenzará la producción gastando recursos. 2. El edificio quedará inutilizado. 3. La construcción de la unidad avanzará a medida que pase el tiempo.

Tabla 146: Caso de Uso CU-11

CU-12	Atacar enemigo
Actores	Jugador
Objetivo	Que una unidad realice un disparo sobre un enemigo.
Precondiciones	Haber ordenado a la unidad que ataque a otra unidad o edificio enemigo.
Postcondiciones	El objetivo ha perdido parte de sus puntos de vida.
Escenario básico	1. La unidad realizará un disparo a corta distancia de su objetivo.

Tabla 147: Caso de Uso CU-12

CU-13	Moverse sobre el mapa
Actores	Jugador
Objetivo	Que una unidad cambie su posición sobre el mapa.
Precondiciones	Haber dado una orden que implique desplazarse hasta el lugar de realización de la misma.
Postcondiciones	La unidad ha cambiado su posición en el mapa, llegando así a su destino.
Escenario básico	1. La unidad considerará la distancia a recorrer. 2. La unidad calculará el camino más corto. 3. La unidad avanzará a su velocidad por las diferentes etapas del camino.

Tabla 148: Caso de Uso CU-13

CU-14	Guardar opciones
Actores	Jugador
Objetivo	Que las opciones estén vigentes entre ejecuciones.
Precondiciones	Haber entrado en el menú de configuración de opciones.
Postcondiciones	Las opciones se mantienen iguales en próximas ejecuciones.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú.

Tabla 149: Caso de Uso CU-14

CU-15	Restaurar valores por defecto
Actores	Jugador
Objetivo	Que las opciones tomen los valores de fábrica.
Precondiciones	Haber entrado en el menú de configuración de opciones.
Postcondiciones	Las opciones tienen el valor por defecto.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú.

Tabla 150: Caso de Uso CU-15

CU-16	Modificar velocidad desplazamiento mapa
Actores	Jugador
Objetivo	Que el mapa se desplace a la velocidad adecuada.
Precondiciones	Haber entrado en el menú de configuración de opciones.
Postcondiciones	El valor de la opción de desplazamiento de mapa es el deseado.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú.

Tabla 151: Caso de Uso CU-16

CU-17	Modificar resolución
Actores	Jugador
Objetivo	Que la aplicación ocupe un tamaño concreto en pantalla.
Precondiciones	Haber entrado en el menú de configuración de opciones.
Postcondiciones	El valor de la opción de resolución de pantalla es el deseado.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú.

Tabla 152: Caso de Uso CU-17

CU-18	Modificar color bando enemigo
Actores	Jugador
Objetivo	Que el bando enemigo muestre el color adecuado.
Precondiciones	Haber entrado en el menú de configuración de opciones.
Postcondiciones	El valor de la opción de color de bando enemigo es el deseado.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú.

Tabla 153: Caso de Uso CU-18

CU-19	Configurar opciones
Actores	Jugador
Objetivo	Mostrar el menú de configuración de opciones.
Precondiciones	Haber entrado en el menú principal.
Postcondiciones	La aplicación muestra el conjunto de opciones configurables, sus valores actuales y las operaciones de guardar, restaurar y cancelar.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú.

Tabla 154: Caso de Uso CU-19

CU-20	Modificar color bando amigo
Actores	Jugador
Objetivo	Que el bando amigo muestre el color adecuado.
Precondiciones	Haber entrado en el menú de configuración de opciones.
Postcondiciones	El valor de la opción de color de bando amigo es el deseado.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú.

Tabla 155: Caso de Uso CU-20

CU-21	Silenciar música
Actores	Jugador
Objetivo	Que la música no se oiga.
Precondiciones	La música debe estar oyéndose.
Postcondiciones	La música no se oye.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú de opciones o activa un método directo desde teclado.

Tabla 156: Caso de Uso CU-21

CU-22	Modificar volumen música
Actores	Jugador
Objetivo	Cambiar el volumen de la música.
Precondiciones	La música debe estar oyéndose.
Postcondiciones	El volumen de la música tiene el valor adecuado.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú de opciones o activa un método directo desde teclado.

Tabla 157: Caso de Uso CU-22

CU-23	Silenciar efectos de sonido
Actores	Jugador
Objetivo	Que los efectos de sonido no se oigan.
Precondiciones	Los efectos de sonido deben estar oyéndose.
Postcondiciones	Los efectos de sonido no se oyen.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú de opciones o activa un método directo desde teclado.

Tabla 158: Caso de Uso CU-23

CU-24	Modificar volumen efectos de sonido
Actores	Jugador
Objetivo	Cambiar el volumen de los efectos de sonido.
Precondiciones	Los efectos de sonido deben estar oyéndose.
Postcondiciones	El volumen de los efectos de sonido tiene el valor adecuado.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú de opciones o activa un método directo desde teclado.

Tabla 159: Caso de Uso CU-24

CU-25	Pausar partida
Actores	Jugador
Objetivo	Detener momentáneamente una partida
Precondiciones	Haber empezado a jugar una partida.
Postcondiciones	La partida se encuentra detenida.
Escenario básico	1. El jugador activa la opción adecuada mediante un método directo desde teclado.

Tabla 160: Caso de Uso CU-25

CU-26	Salir del juego
Actores	Jugador
Objetivo	Terminar la ejecución de la aplicación.
Precondiciones	Encontrarse en el menú principal.
Postcondiciones	La aplicación se habrá cerrado.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú.

Tabla 161: Caso de Uso CU-26

CU-27	Volver al menú
Actores	Jugador
Objetivo	Retroceder al menú principal.
Precondiciones	Estar en la pantalla de juego o en la pantalla de configuración de opciones.
Postcondiciones	La aplicación muestra el menú principal.
Escenario básico	1. El jugador activa el menú de pausa y selecciona la opción adecuada (pantalla de juego) o simplemente selecciona la opción adecuada (pantalla de configuración de opciones).

Tabla 162: Caso de Uso CU-27

CU-28	Mostrar menú
Actores	Jugador
Objetivo	Mostrar el menú principal.
Precondiciones	La aplicación debe estar cerrada.
Postcondiciones	La aplicación muestra el menú principal.
Escenario básico	1. El jugador activa la ejecución del fichero que contiene el juego.

Tabla 163: Caso de Uso CU-28

CU-29	Jugar partida
Actores	Jugador
Objetivo	Disputar una batalla entre varios bandos.
Precondiciones	Estar en el menú principal.
Postcondiciones	Se estará en la pantalla de juego.
Escenario básico	1. El jugador selecciona la opción adecuada dentro del menú.

Tabla 164: Caso de Uso CU-29

CU-30	Reiniciar partida
Actores	Jugador
Objetivo	Volver a jugar una partida.
Precondiciones	Encontrarse actualmente en la pantalla de juego.
Postcondiciones	Encontrarse en la pantalla de juego con la disposición inicial de unidades, enemigos y bandos.
Escenario básico	1. El jugador accede al menú de pausa. 2. El jugador accede al menú principal. 3. El jugador accede a la opción de jugar.

Tabla 165: Caso de Uso CU-30

3.5. Diagrama de actividad

En el apartado anterior se exploraron los Casos de Uso. Esta herramienta, utilizada para representar la interacción entre usuario y sistema, contribuye a clarificar las funcionalidades que se van a implementar. En este mismo sentido se ha decidido crear el diagrama de actividad del sistema.

Los diagramas de actividad describen las posibles secuencias de actividades en el sistema centrándose en el estado en que se encuentra la aplicación. Estos diagramas constan, por tanto, tendrán un punto de inicio, un punto de fin, un conjunto de estados y un conjunto de transiciones entre estados.

Las transiciones son la forma de avanzar entre estados y pueden implicar un procesamiento, un cambio de estado o ambos elementos. Definen los sucesos que pueden acontecer en un determinado estado y que influyen en el comportamiento y evolución del sistema. A continuación se muestra el diagrama actividad del sistema.

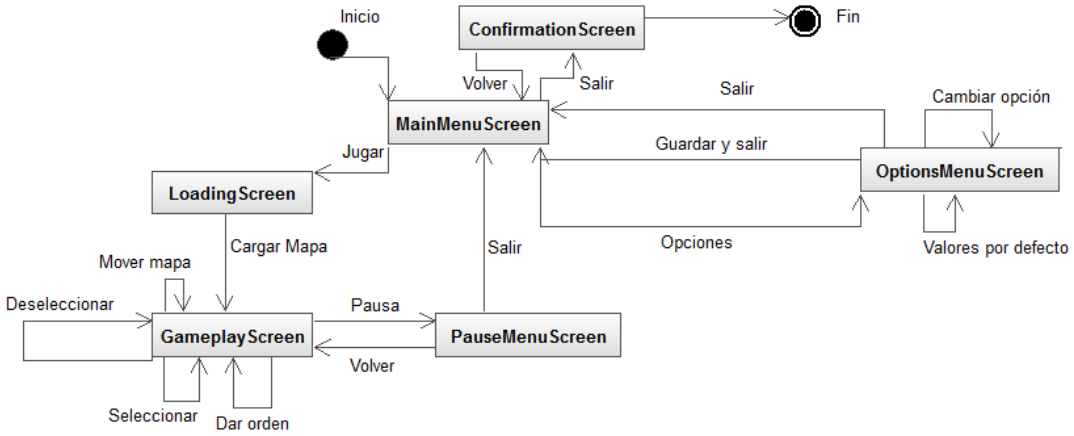

Figura 59. Diagrama de actividad del sistema

3.6. Diagramas de secuencia

Las herramientas anteriormente presentadas han permitido conseguir una clara percepción de los rasgos del sistema: concretamente, de cómo el usuario puede interactuar con el sistema, y qué debe realizar el sistema para el usuario. Esta sección se dedica a la exploración de una herramienta más: los diagramas de secuencia.

Los diagramas de secuencia permiten representar la comunicación entre diferentes objetos del sistema para llevar a cabo el Caso de Uso solicitado. Se estudia esta herramienta por dos motivos:

- Muestran detalles de implementación del escenario: clases utilizadas para implementar el sistema y mensajes intercambiados entre los objetos.
- Sirven para conducir el final de la fase de análisis hacia el comienzo de la fase de diseño.

Por último, antes de comenzar la presentación de diagramas de secuencia elaborados es necesario destacar que únicamente se recogen aquí aquellos que realmente aporten contenido de valor, eliminando aquellos que más similares entre sí.

Jugar

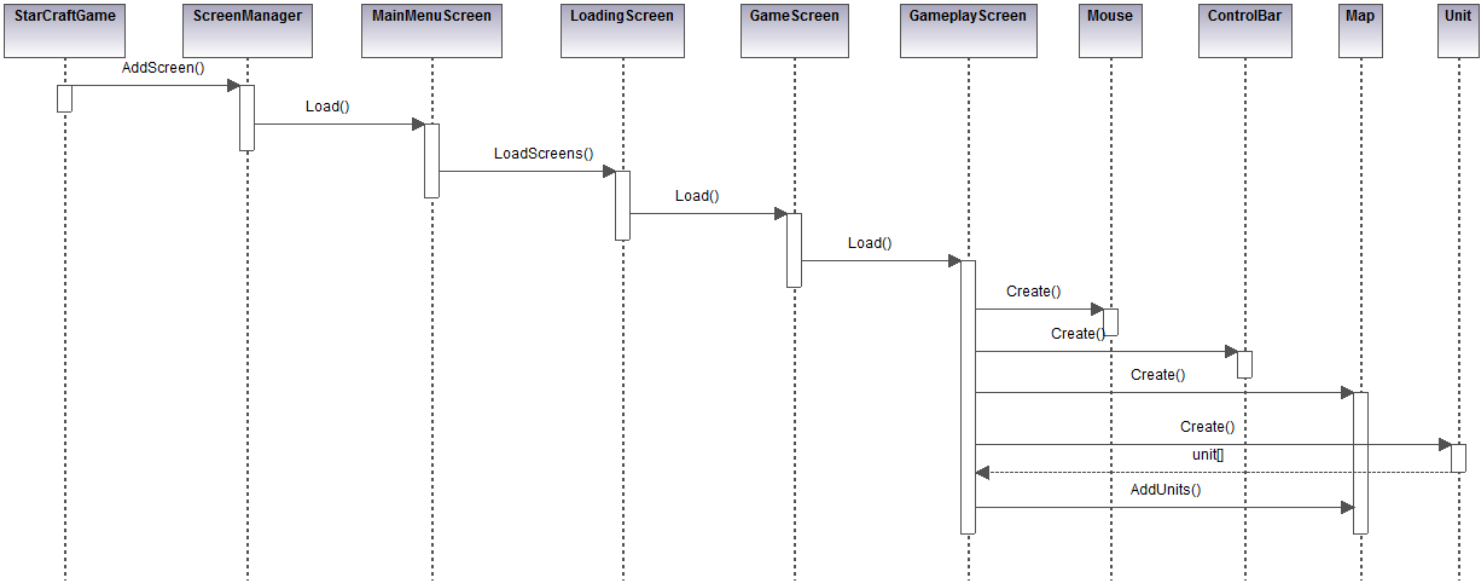

Figura 60. Diagrama de secuencia "Jugar"

En la Figura 60 se aprecia el diagrama de secuencia relativo a la operación "Jugar". Esta funcionalidad consiste en que el jugador lanza el juego y comienza una nueva partida. En el diagrama se observa el papel de la clase ScreenManager, cuya función es gestionar las pantallas de la aplicación. La primera pantalla será MainMenuScreen.

Al seleccionar la opción de Jugar en el menú, el resto del diagrama de secuencia entra en acción. En primer lugar, la pantalla de carga LoadingScreen orienta al usuario sobre la actividad que está realizando la aplicación. Después, la clase GameScreen carga los recursos asociados a la partida que se va a jugar y GameplayScreen los prepara, transformándolos en instancias concretas, para dar lugar al ratón, la barra de control, el mapa y las unidades.

Actualizar y dibujar una imagen en pantalla

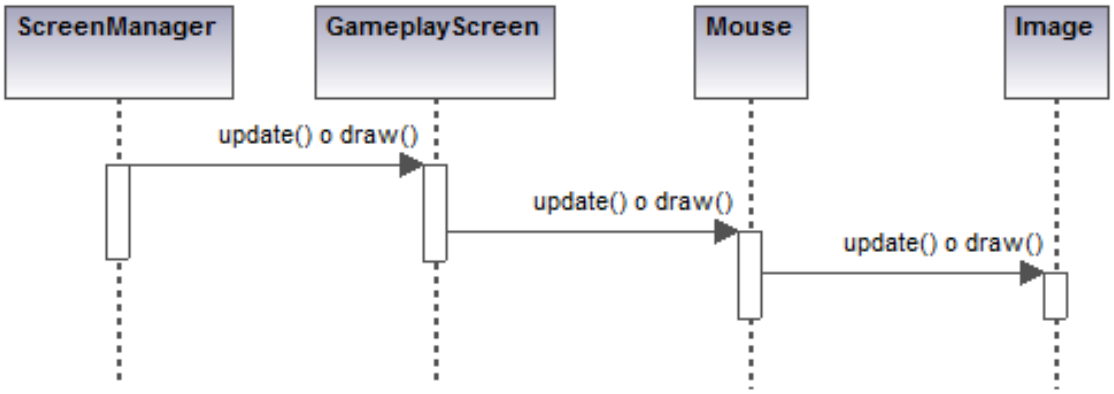

Figura 61. Diagrama de secuencia "Actualizar y dibujar"

En la Figura 61 se aprecia la secuencia necesaria para actualizar y para dibujar una imagen en pantalla. En primer lugar, el framework XNA seguirá la cadena de llamadas update()-draw()-update()… de manera indefinida. Dado que solamente se le presentará la existencia de la clase ScreenManager, XNA llamará a los métodos de dicha clase. Entonces, ScreenManager derivará la petición a la pantalla activa: en este caso, GameplayScreen. Esta pantalla pedirá a sus elementos que cumplan la petición, para lo cual éstos accederán a la clase Image, de más bajo nivel, solicitando la operación necesaria. Image, finalmente, calculará los datos que le sean necesarios y actualizará o dibujará según sea la orden pedida.

Dar una orden a una unidad o edificio

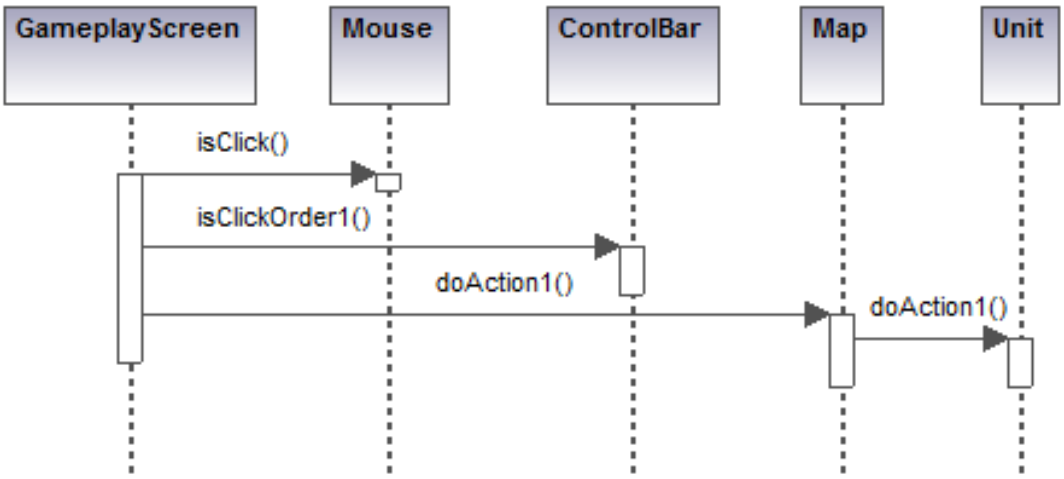

Figura 62. Diagrama de secuencia "Dar orden"

En la Figura 62 se detalla la cadena de llamadas necesaria para dar una orden a una unidad o un edificio. En primer lugar, la clase GameplayScreen comprobará si el usuario ha pulsado un botón del ratón. Este dato será provisto automáticamente por el framework XNA. Después, si se ha pulsado un botón, GameplayScreen preguntará a ControlBar si esa pulsación está dentro de su superficie de pantalla. En el diagrama, la pulsación ha recaído sobre la barra de control (isClickOrder1), por lo que GameplayScreen informa de ello a Map (doAction1). Map, a su vez, traslada la orden a la unidad seleccionada.

Este diagrama sería similar para todas las órdenes, sean para unidades o para edificios. También sería válido para las pulsaciones recibidas en el mapa en miniatura.

Acción por defecto “Mover”

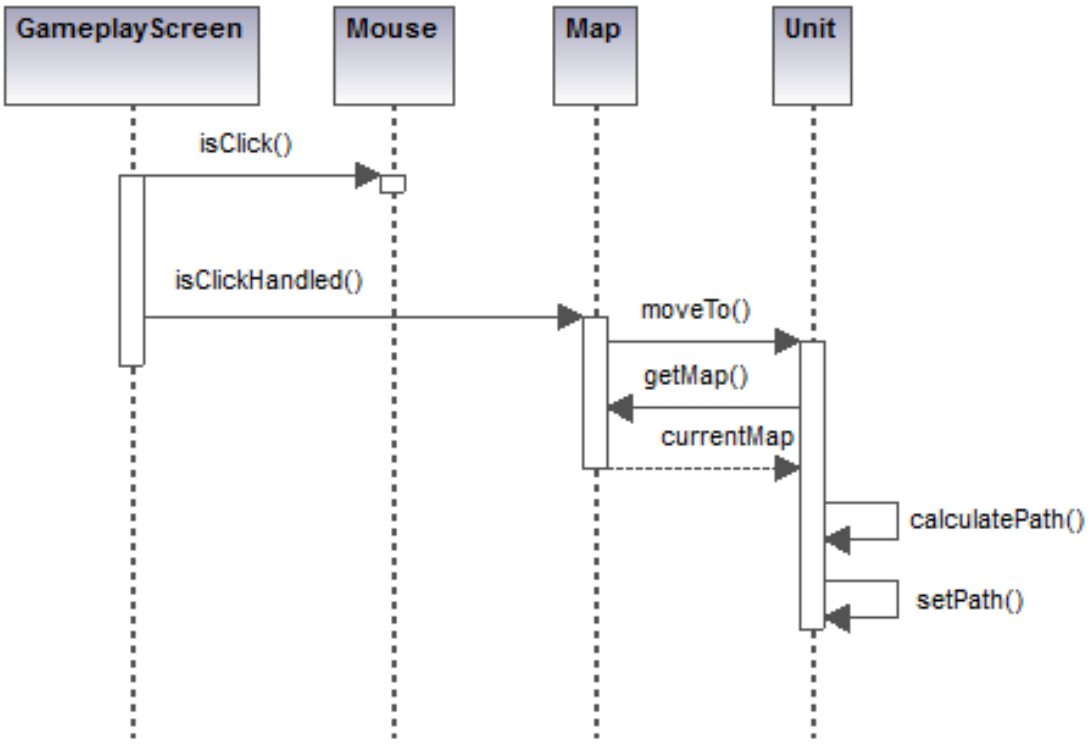

Figura 63. Diagrama de secuencia “Acción por defecto Mover”

En la Figura 63 se observa la cadena de llamadas necesaria para responder a una acción por defecto solicitada por el usuario. En primer lugar, GameplayScreen determinará si la pulsación se ha realizado sobre el mapa (isClickHandled). De ser así, si hay una unidad seleccionada, se entenderá que se quiere desplazar la unidad a la posición indicada (moveTo). Por tanto, Unit copiará los datos actuales del mapa, que cambia en tiempo de ejecución (getMap) y calculará el camino a seguir (calculatePath). Este camino será utilizado en cada llamada a update() para que la unidad actualice su posición en función del camino, el tiempo y la velocidad.

Capítulo 4

Fase de diseño

Este capítulo recoge el trabajo realizado durante la etapa de diseño del presente proyecto.

El primer apartado, **4.1. Diseño de interfaz**, recoge el material utilizado como base para la posterior implementación de la interfaz de usuario de la aplicación.

El segundo apartado, **4.2. Diagrama de clases**, muestra el diseño global del programa: las clases que lo componen y sus relaciones.

Por último, el tercer apartado, **4.3. Definición de clases**, sirve para concretar los aspectos clave de cada clase del sistema. Esto aumenta el nivel de detalle, introduciendo así la posterior fase de implementación.

4.1. Diseño de interfaz

Esta sección recoge el diseño realizado sobre los elementos de interfaz de usuario de la aplicación. Esta actividad es especialmente relevante por dos motivos: primero, porque la cantidad de requisitos relacionados con la interfaz es significativa; y, segundo, porque en un videojuego la experiencia de usuario es un aspecto clave.

A continuación se presentan los bocetos realizados en las pantallas que será necesario crear. Cada boceto tiene asociado un texto explicativo, detallando así los aspectos que no es posible recoger en una imagen esquemática (transiciones, interacciones, colores a utilizar, imagen de fondo…).

Menú Principal

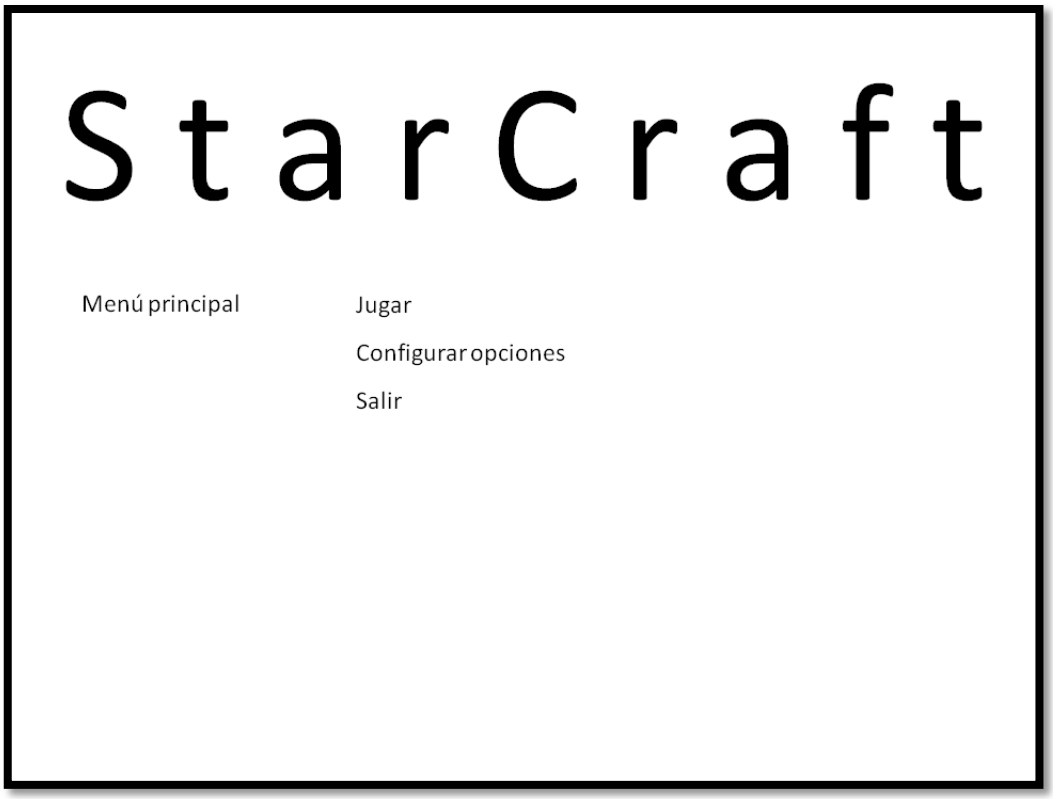

Figura 64. Diseño de Pantalla "Menú Principal"

En la Figura 64 se ofrece el diseño realizado para la primera pantalla del programa, llamada "Menú Principal". En ella se puede apreciar el título de la aplicación, "StarCraft", que deberá estar convenientemente resaltado y estilizado. La pantalla deberá indicar al usuario en qué menú se encuentra, sea mediante la etiqueta "Menú principal" o algún otro medio.

La pantalla tendrá una imagen de fondo en todo momento. Esta imagen será acorde con la temática futurista y espacial del juego original.

Las opciones de navegación disponibles aparecerán listadas en el “cuerpo” del menú. En la Figura 64 aparecen a la derecha de la etiqueta de navegación, aunque pueden estar en otra posición siempre que formen un único cuerpo de opciones y sea visualmente agradable.

Menú de Configuración de Opciones

Figura 65. Diseño de Pantalla “Menú de Configuración de Opciones”

En la Figura 65 se ofrece el diseño realizado para la pantalla de opciones, llamada “Menú de Configuración de Opciones”. El diseño de este Menú y el del Menú Principal se presentan similares en este documento: la idea subyacente es que deben estar unificados.

Por otro lado, este Menú tiene libertad para estructurar las opciones y sus valores: puede variar el orden de las mismas o su nomenclatura siempre y cuando mantenga la unidad de un cuerpo y sea similar al Menú Principal. Las opciones generalistas (Salir, Guardar y salir…) deben formar un único cuerpo que ha de aparecer separado de las opciones concretas (Opción 1, Opción 2…).

Pantalla de Carga

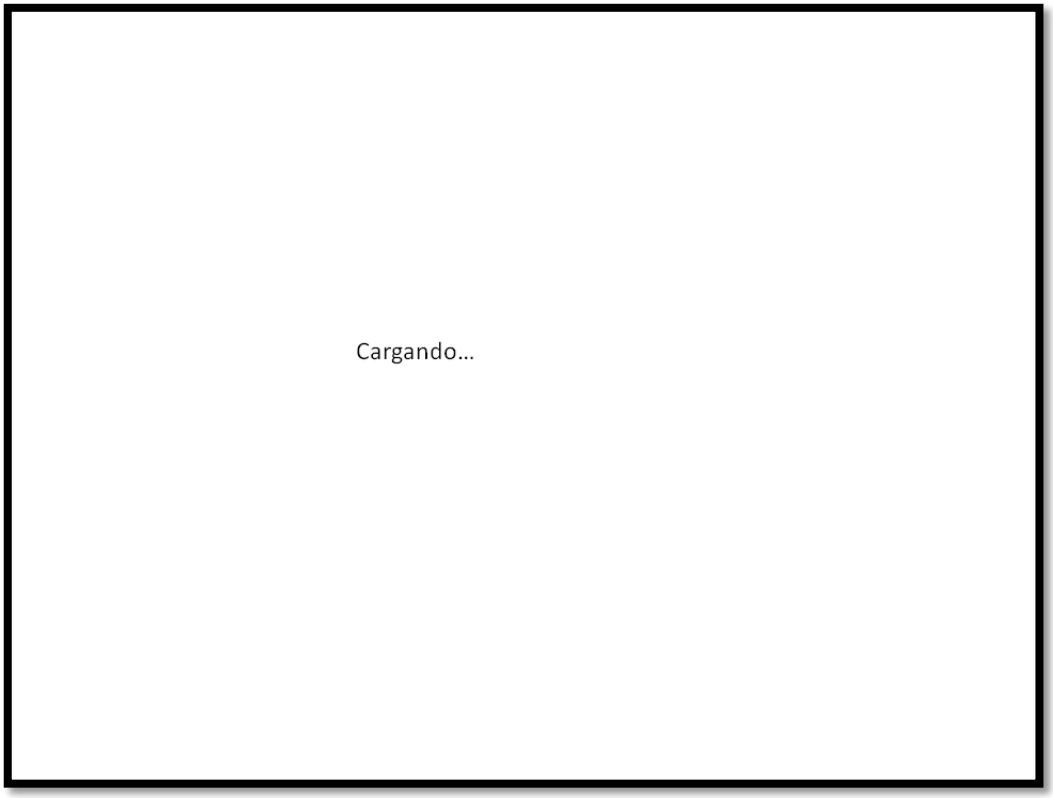

Figura 66. Diseño de Pantalla "Pantalla de Carga"

En la Figura 66 se ofrece el diseño realizado sobre la "Pantalla de Carga". Este elemento se encargará de regular el proceso de carga de recursos teniendo en cuenta las restricciones de tiempo descubiertas durante el análisis.

Esta Pantalla tendrá un carácter minimalista: ofrecerá una etiqueta informativa sobre el proceso que está teniendo lugar. Opcionalmente podrá incluirse una orientación de tiempo: porcentaje completado, porcentaje restante, tiempo restante, una animación… El fondo de la pantalla debe ser negro, ya que esta pantalla es la única que se presentará mientras dure la carga de datos.

Pantalla de Juego

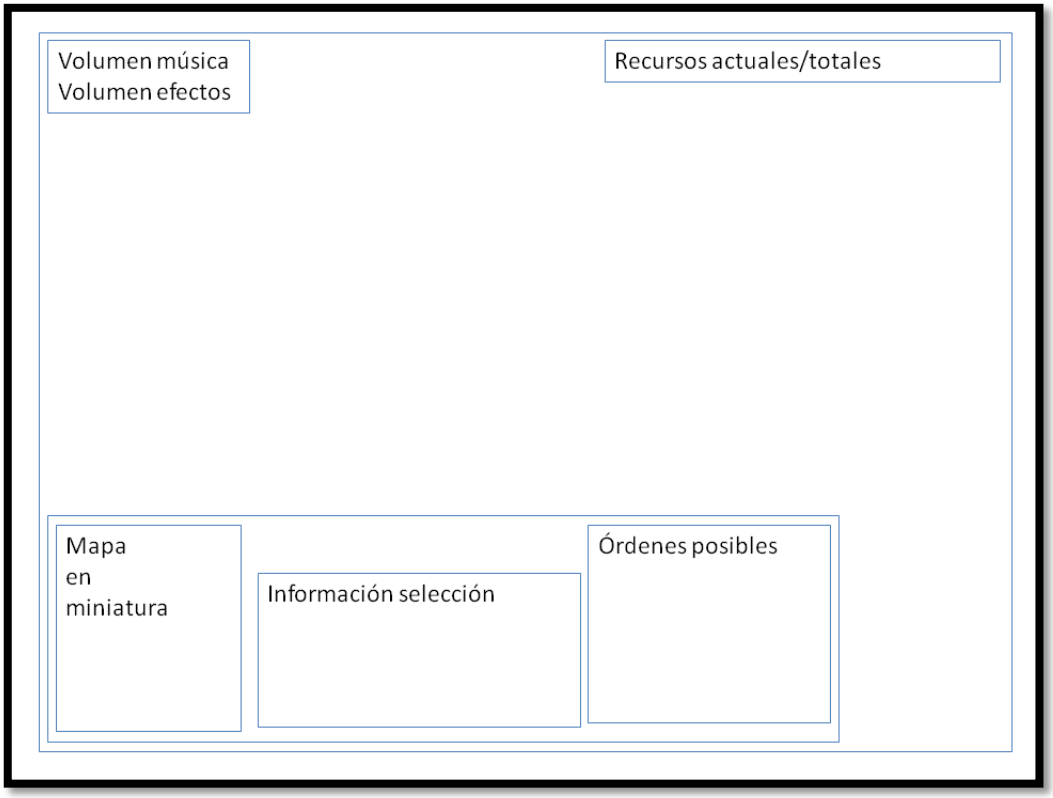

Figura 67. Diseño de Pantalla "Pantalla de Juego"

En la Figura 67 se ofrece el diseño realizado sobre la "Pantalla de Juego". Esta pantalla tiene como función la orientación visual del jugador a lo largo de una partida de juego.

En primer lugar, se desea destacar la presencia del mapa. Este elemento debe ocupar todo el espacio posible de pantalla, por lo que se permite el uso de transparencias según se vea adecuado. En el diagrama, el mapa es el recuadro azul más grande. Los demás elementos se sitúan encima del mapa.

En la esquina superior izquierda se mostrará el estado de la música y de los efectos de sonido: volumen actual y máximo o, si están silenciados, una etiqueta que indique este hecho. Estos valores podrán cambiar en cualquier momento, ya que el jugador podrá regularlos durante la partida.

En la esquina superior derecha se mostrarán los recursos de que dispone el jugador: gas, cristal y población. Estos recursos estarán ilustrados con iconos característicos. Se mostrará la cantidad actual y la cantidad máxima de cada uno. Estos datos variarán a lo largo de la partida.

En la parte inferior se muestra el elemento que agrupa el mapa en miniatura, la información sobre la selección realizada y las órdenes que puede dar el jugador. Este elemento centralizador tendrá un diseño similar a la misma estructura del juego

original y, en adelante, se hablará de ella como “Barra de Control”. Este elemento estará siempre unido a la esquina inferior izquierda de la pantalla.

- Mapa en miniatura: el mapa en miniatura representará con toda la fidelidad que sea posible el mapa real. Se mostrará un indicador del punto de vista actual del jugador.
- Información sobre la selección: si se selecciona una unidad o un edificio, esta zona mostrará la información adicional sobre el mismo. Esta información estará ilustrada con una miniatura de la unidad o edificio en cuestión.
- Órdenes posibles: cuando se haya seleccionado una unidad o edificio que pueda recibir órdenes, dichos elementos aparecerán en esta sección. Se presentará un icono característico y, opcionalmente, el nombre de la orden (“Atacar”, “Parar”…).

Pantalla de Pausa

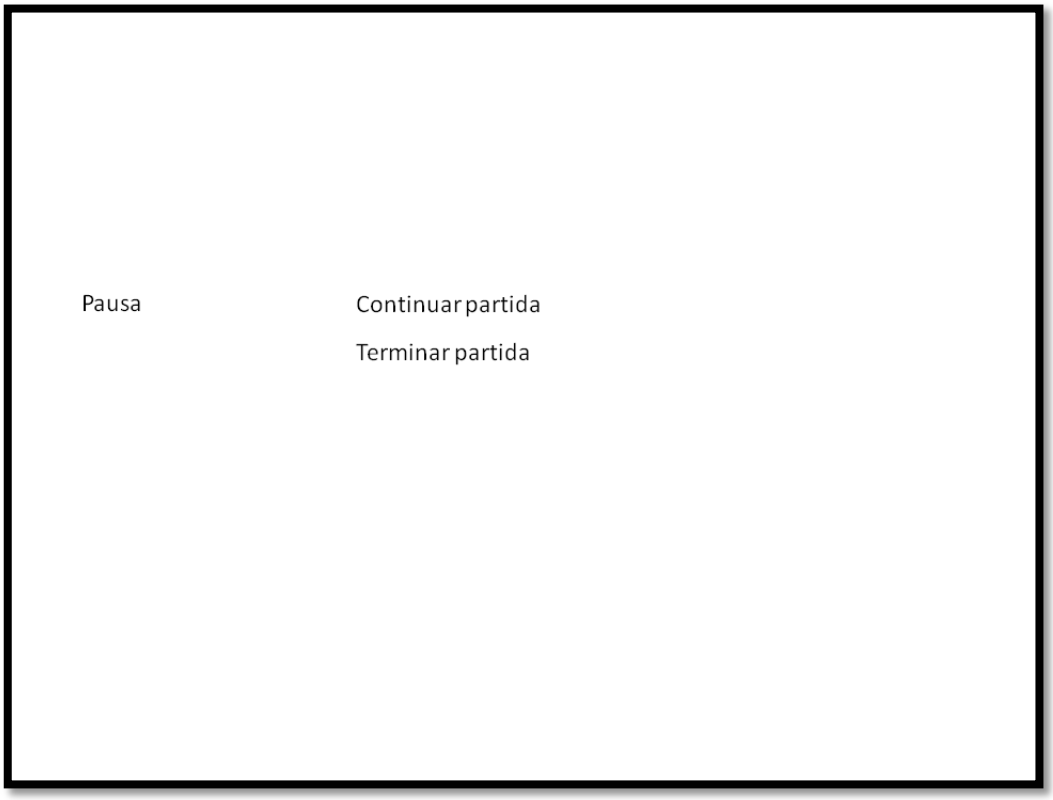

Figura 68. Diseño de Pantalla “Menú de Pausa”

En la Figura 68 se ofrece el diseño realizado sobre el “Menú de Pausa”. Este Menú será accesible desde la Pantalla de Juego, y servirá para detener o finalizar la partida en curso.

El diseño de este Menú respetará los diseños realizados en otros Menús: esto es, tendrá una apariencia similar. Este Menú será transparente, por lo que el fondo será la

propia Pantalla de Juego. Opcionalmente se podrá detener la música o los efectos de sonido.

Diálogo de Confirmación

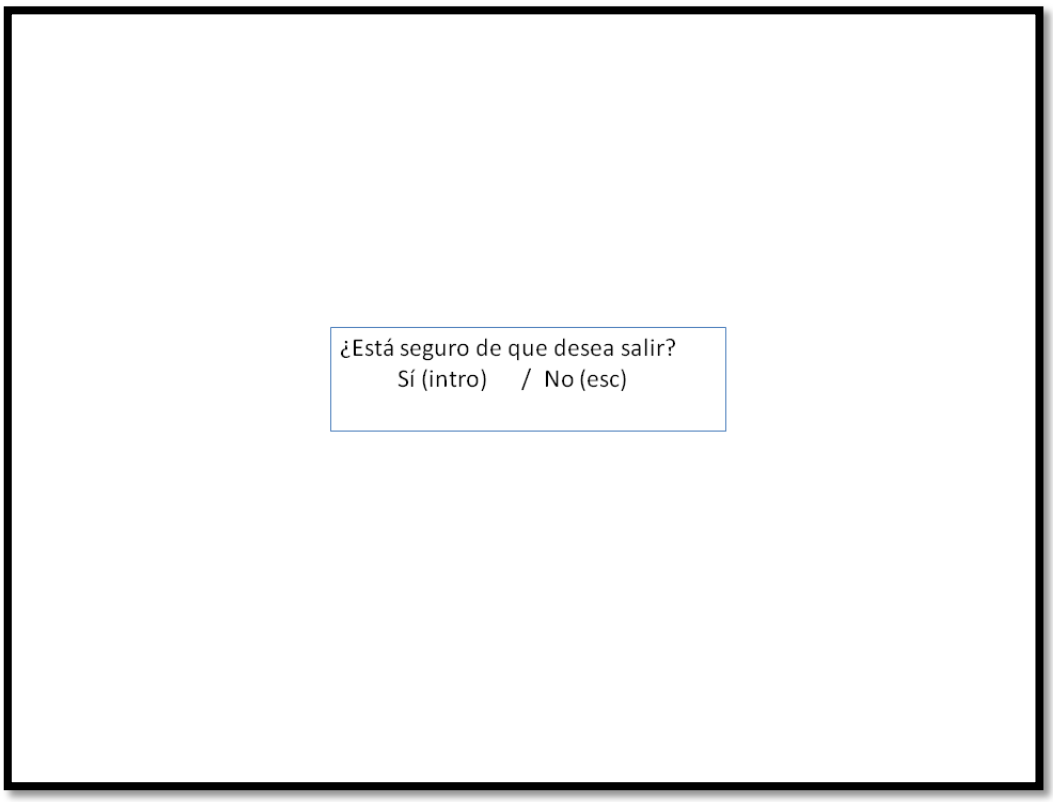

Figura 69. Diseño de Pantalla "Diálogo de Confirmación"

En la Figura 69 se ofrece el diseño realizado para el "Diálogo de Confirmación de Salida" de la aplicación, aunque todos los cuadros de diálogo implementados deberán ser similares.

Esta pantalla será transparente: tendrá de fondo la pantalla que haya propiciado la existencia del cuadro de diálogo. Se detallará la pregunta que se desea que el usuario responda así como las posibilidades de respuesta que éste tiene. Como se indica en el diagrama, no será necesaria la existencia de botones, pudiendo detallar las posibles respuestas mediante texto o algún recurso gráfico.

4.2. Diagrama de clases

Esta sección recoge el diagrama de clases diseñado para implementar la funcionalidad descrita en los requisitos de usuario. Este diagrama tiene su origen en el análisis del sistema realizado anteriormente (requisitos de usuario, de software, casos de uso, diagramas de actividad y de secuencia), aunque también se encuentra influenciado por el diseño de la interfaz de usuario.

En el diagrama se describen las clases que componen el sistema y sus relaciones a nivel global. En apartados posteriores se detallan los atributos, las operaciones y las restricciones de visibilidad más importantes.

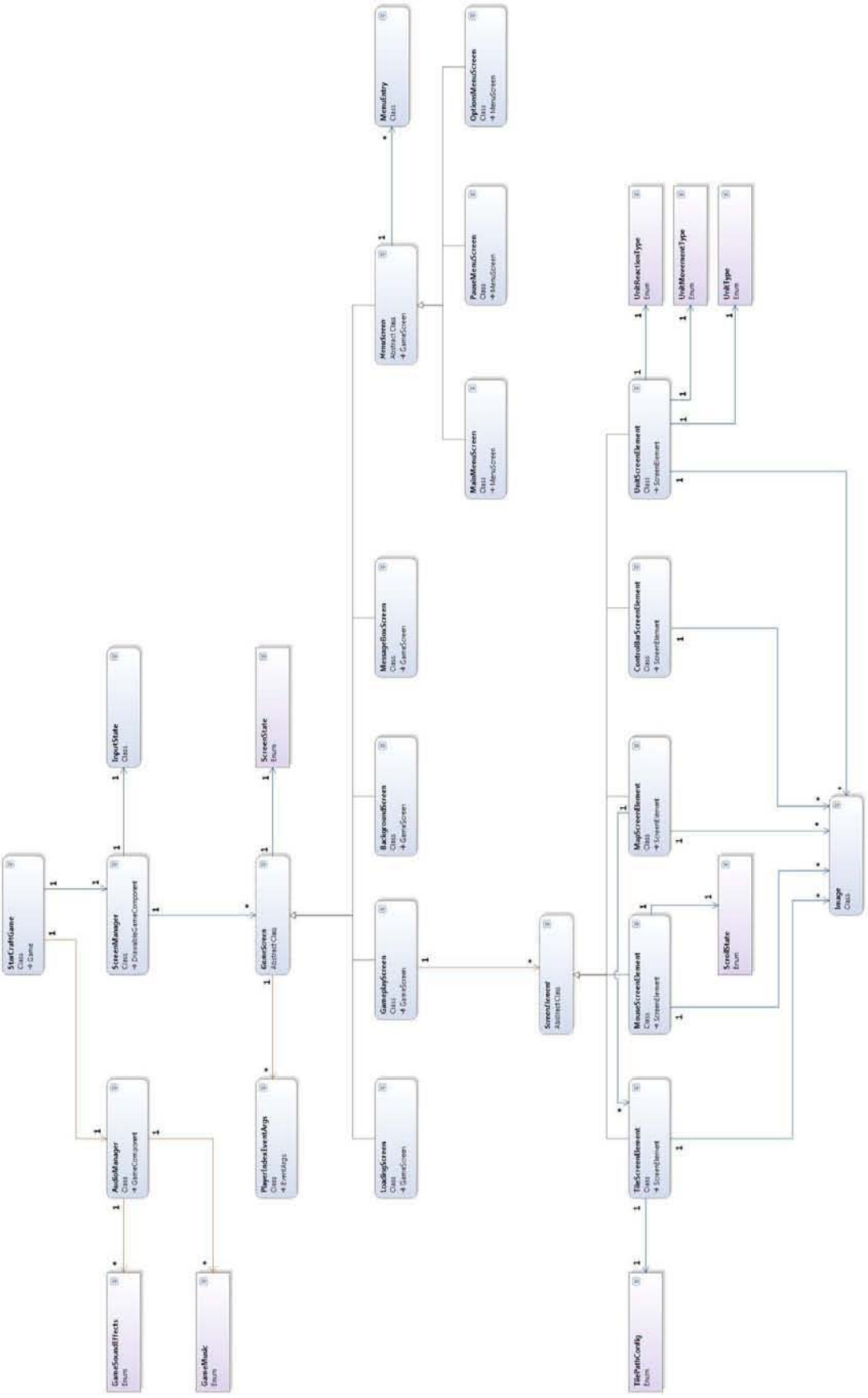

Figura 70. Diagrama de Clases del Sistema

4.3. Definición de clases

Partiendo del diagrama de clases mostrado en la Figura 70, se tiene el diseño de un sistema de tamaño medio compuesto por unas veinte clases. Para abordar el trabajo se explorarán las clases en sus debidos apartados, destacando el propósito de las funciones o variables más importantes.

A lo largo de la sección se utilizará el símbolo "*" para indicar cualquier nombre que encaje con el patrón. Por ejemplo: "Nombre*" puede referirse a "NombreCorto", "NombreLargo" y "Nombre". También es necesario destacar que se hará referencia a todas las clases, pero que no se describirán individualmente por dos motivos:

- En primer lugar, el diseño resulta en ocasiones muy similar (MainMenuScreen, MenuScreen, OptionsMenuScreen…)
- En segundo lugar, el proyecto está basado en el framework XNA (Sprite, GraphicsDevice…) y en componentes de apoyo de Microsoft (gestión de pantallas, InputState…). En este sentido, simplemente se comentará el uso que se da a dichos objetos o los cambios que han sido necesarios respecto a las librerías originales.

4.3.1. Clase StarCraftGame

Contiene el punto de inicio de la aplicación, cuya función es crear una instancia de esta misma clase y unirla a la cadena de operaciones del framework XNA.

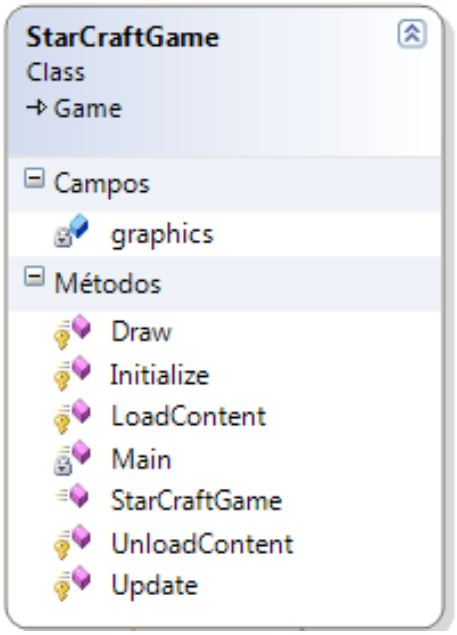

Figura 71. Clase StarCraftGame

Como elementos operacionales más importantes cabe destacar:

- Esta clase hereda directamente de la clase Game. Por este motivo es capaz de unirse a la cadena de operaciones de XNA: está obligada a implementar los métodos Initialize, LoadContent, UnloadContent, Update y Draw.
- Esta clase inicializa el componente AudioManager, que se ocupará de la gestión de sonido de la aplicación.
- Esta clase inicializa el componente ScreenManager, que gestionará el estado de la aplicación. Este componente mantendrá el control sobre el flujo de ejecución, y será actualizado y pintado continuamente.
- El campo graphics almacena la instancia de GraphicsDeviceManager que permite controlar el dispositivo gráfico.

En cuanto a los métodos de la clase:

- Constructor: método que reserva memoria para la instancia y asigna valores a los campos que deban tener algún valor inicial.
- Initialize: método que asigna los nombres de los recursos que utilizará la clase. También inicializará las subclases que sean necesarias (AudioManager, ScreenManager).
- LoadContent: método que realiza la carga de recursos a memoria. Se encarga de llamar al componente ContentManager de XNA, recibiendo en respuesta los recursos necesarios en un formato utilizable.
- UnloadContent: indica a ContentManager que algún recurso ya no es necesario para que éste se encargue de liberar espacio en memoria principal. Este método también es llamado antes de destruir la instancia.
- HandleInput: método que recibe el estado actualizado de las entradas de usuario. Es el punto más indicado para que sean procesadas, aunque no es obligatorio hacerlo. StarCraftGame no muestra este método, ya que no redefine la implementación que tiene su superclase.
- Update: método que actualiza los campos de la instancia en base a la situación actual (entrada de usuario, tiempo…). Es llamado por el framework continuamente.
- Draw: método que dibuja en pantalla lo que la instancia deba pintar. Es llamado por el framework después de llamar al método Update.

Las llamadas a los métodos detallados, dado que vienen impuestos por XNA, son iguales en todas las clases del proyecto. En adelante se dará por supuesto el conocimiento de la cadena de llamadas de XNA, por lo que se explicará directamente el cometido de cada método concreto.

4.3.2. Clases ScreenManager, InputState y GameScreen

Uno de los primeros asuntos a resolver para el diseño de un videojuego es, quizá, la gestión de las diferentes escenas que dan vida al proyecto. Dichas pantallas pueden mostrar datos muy diversos, pero cualquier videojuego puede ser entendido en términos de "pantallas": no en vano es de público dominio el hablar de una partida haciendo referencia a la pantalla en la que se encuentra el usuario.

Por tanto, para la correcta marcha del proyecto este asunto tuvo que ser resuelto lo antes posible. Vista la complejidad de la gestión de estados, actualización y dibujado se escogió un componente de Microsoft especialmente preparado para el efecto descrito anteriormente: el llamado Game State Management[r165].

Esta implementación proporciona un completo y versátil gestor de pantallas: controla la entrada de datos a la pantalla activa y facilita las transiciones entre cada pantalla. Al mismo tiempo permite que cada una de ellas tenga su propia lógica y contenido independiente.

La función de la clase ScreenManager es administrar la presentación, superposición y actualización de las pantallas, así como determinar cuál de ellas es la pantalla activa para proporcionarle la entrada de usuario.

En el paquete se incluye de manera independiente la clase InputState. Esta clase permite abstraer las operaciones de entrada de usuario ya que proporciona llamadas de alto nivel. Por ejemplo, se puede tener una función que determine si el usuario cancela una operación: en PC, la función comprobará una tecla del teclado, mientras que en XBOX 360 comprobará un botón del mando de la consola. Esta clase será la entrada utilizada por todas las pantallas de la aplicación.

A continuación se proporciona un diagrama de clases detallando el paquete utilizado.

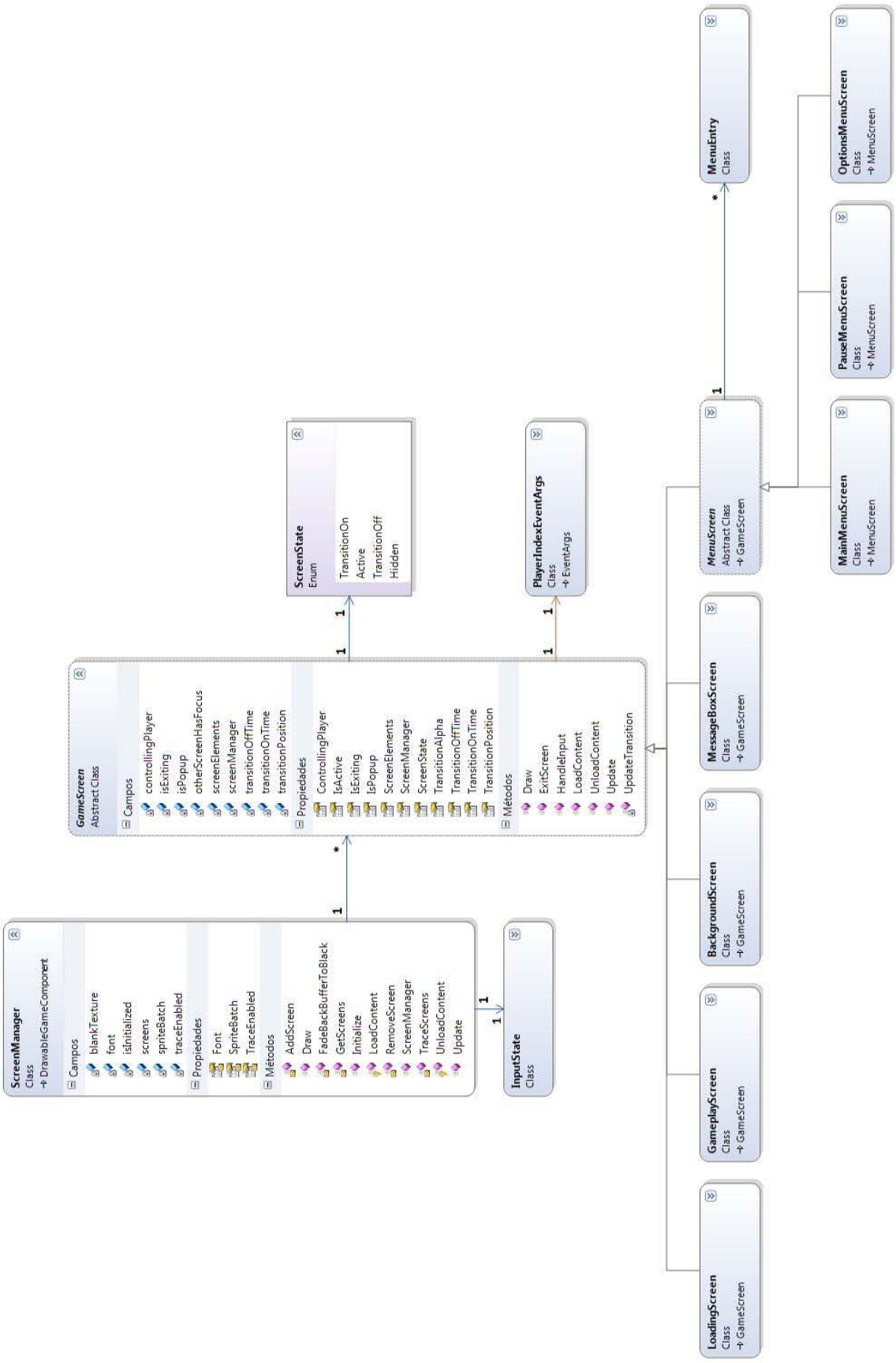

Figura 72. Clases ScreenManager, InputState y GameScreen

4.3.3. Clases Screen

De manera complementaria a las clases de gestión, el componente Game State Management implementa un conjunto de pantallas de uso común. Esto sirve al mismo tiempo de ejemplo de funcionalidad del componente y de demostración de uso del framework.

Parte de estas pantallas se reaprovecha para diseñar el videojuego que ocupa este proyecto. Como se expone a continuación, cada una de las pantallas contiene unas funcionalidades muy concretas.

- BackgroundScreen**:** esta clase está pensada para permanecer por debajo de todas las demás. Por tanto, es óptima su utilización como fondo del resto de pantallas.
- GameplayScreen: esta pantalla deberá estar unida a la opción de "Comenzar una nueva partida". Contendrá la lógica y los datos de juego, tanto de actualización como de dibujado.
- LoadingScreen: pantalla diseñada para controlar el proceso de carga de otras pantallas. Es capaz de mostrar un mensaje por pantalla informando sobre la actividad que se está realizando.
- MenuScreen: pantalla de menú que generaliza este sistema. Es una colección de instancias de MenuEntry, cada una configurada para realizar una acción cuando sean activadas.
    - MainMenuScreen: pantalla que contiene la funcionalidad del Menú Principal de la aplicación. Las opciones que tendrá serán las de iniciar una nueva partida, configurar las opciones disponibles o terminar la aplicación.
    - OptionsMenuScreen: pantalla que contiene las opciones disponibles y sus valores actuales. Esta pantalla era muy similar a MainMenuScreen, pero se ha modificado para ser el punto de acceso global a las opciones del sistema. También es capaz de leer y escribir los ajustes a memoria permanente.
    - PauseMenuScreen: pantalla de pausa del juego. Su funcionalidad es muy similar a la de MainMenuScreen. Sin embargo, esta pantalla está diseñada para ser invocada desde GameplayScreen, por lo que oscurece la pantalla gradualmente en lugar de utilizar una imagen de fondo.
- MessageBoxScreen: pantalla diseñada para mostrar un cuadro de diálogo al usuario, deteniendo el resto de operaciones y devolviendo el valor de aceptación o rechazo cuando se produzca.

A continuación se muestra el diagrama de clases que recoge los elementos comentados.

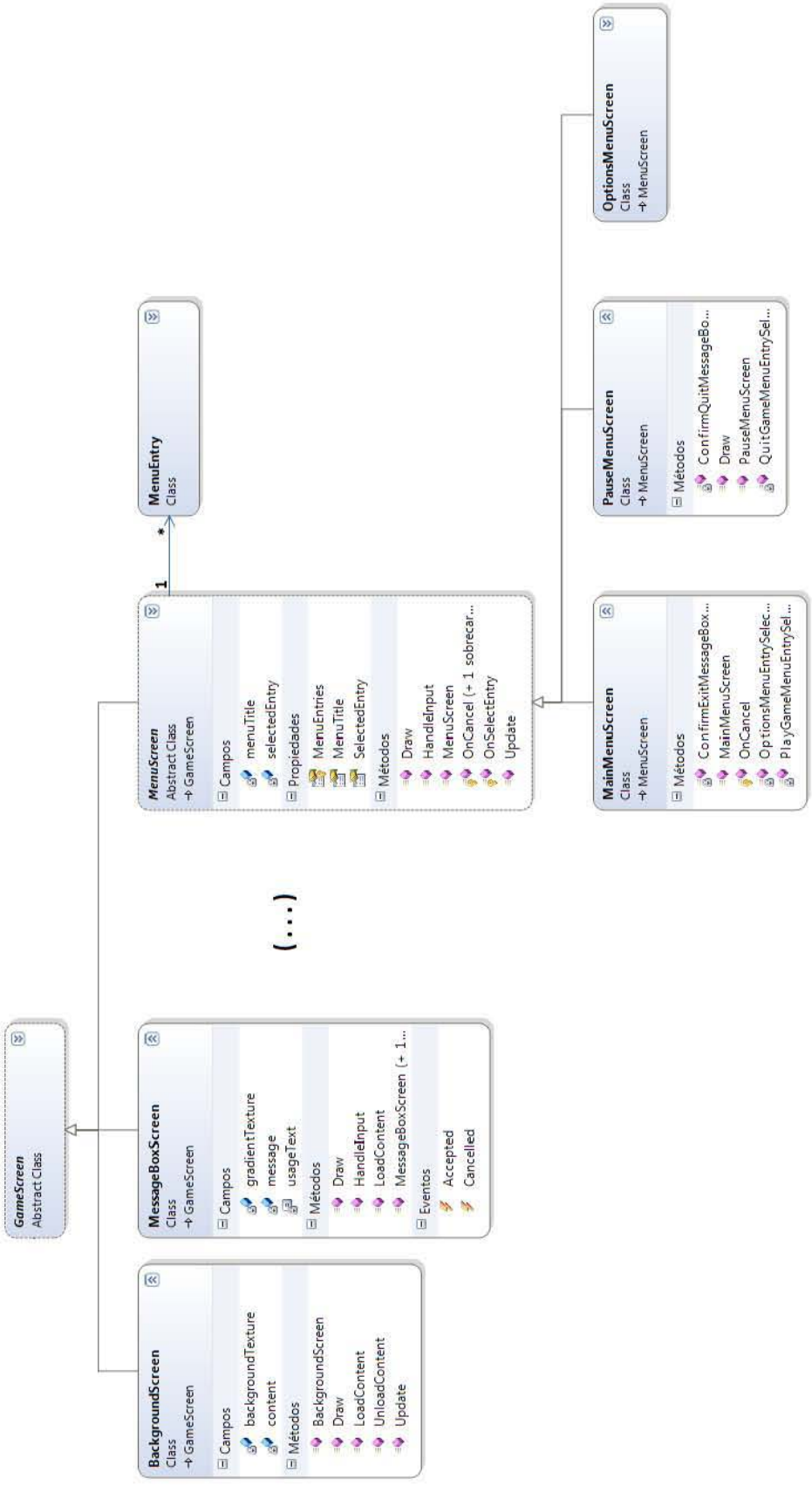

Figura 73. Detalle de las Clases Screen

4.3.4. Clase AudioManager

Esta clase gestiona los aspectos de sonido de la aplicación: música y efectos de sonido. Contiene la funcionalidad necesaria para cargar y descargar dichos recursos, reproducirlos, pararlos y cambiar el volumen de ambos por separado.

La clase AudioManager es creada por StarCraftGame y unida a su colección de componentes. La operación de esta clase se basa en las clases MediaPlayer y SoundEffect de XNA, así como en el hecho de ser una subclase de GameComponent. Estas herencias, en conjunto, conseguirán que el framework trate de actualizar las instancias de esta clase tantas veces como sea posible, consiguiendo la ilusión de un sonido continuo.

Esta clase dispone los métodos *VolumeUp, *VolumeDown, Mute* y Play* para que el resto de la aplicación pueda reproducir los sonidos que necesite. Estos sonidos son los que aparecen como Campos en el diagrama de clases siguiente, aunque se accederá a los datos siempre mediante los campos Enum.

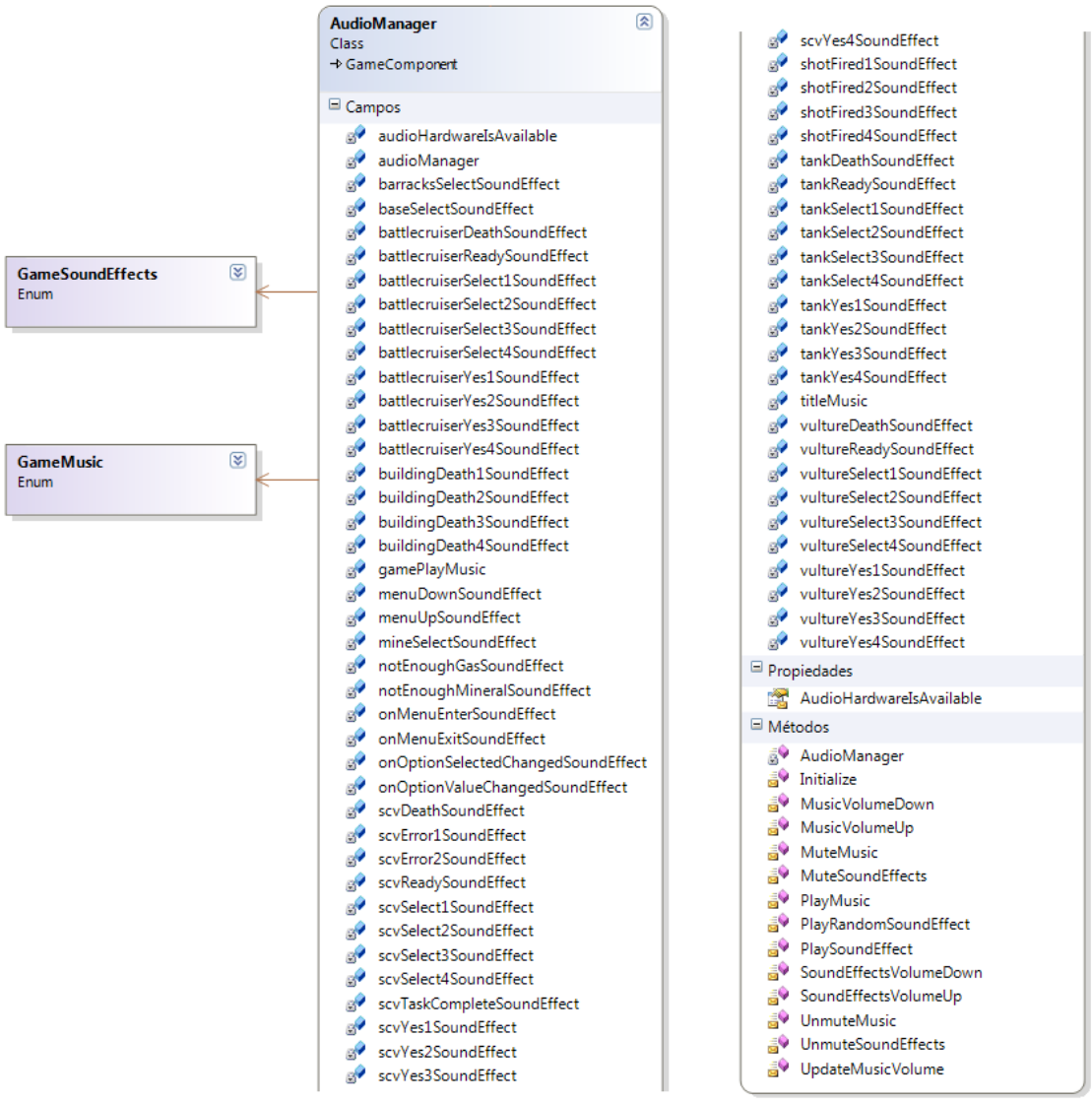

Figura 74. Clase AudioManager

4.3.5. Clase ScreenElement

Esta clase está diseñada como la generalización de todos los elementos con los que se puede interactuar a lo largo de una partida. Esta clase se sitúa por debajo de GameplayScreen, que tendrá referencias a diversas instancias, una por elemento constructivo de la partida.

La idea de crear esta clase es muy similar a la que motiva el paquete Game State Management. Mientras que el paquete mencionado gestiona qué pantalla se muestra y se ocupa de hacerle llegar la entrada de usuario, GameplayScreen controlará que todos los ScreenElement estén en pantalla al mismo tiempo, y se ocupará de hacer llegar la entrada de usuario al ScreenElement apropiado. Como se aprecia en la Figura 75, algunos ScreenElement son figuras básicas en el juego: la barra de control, el ratón…

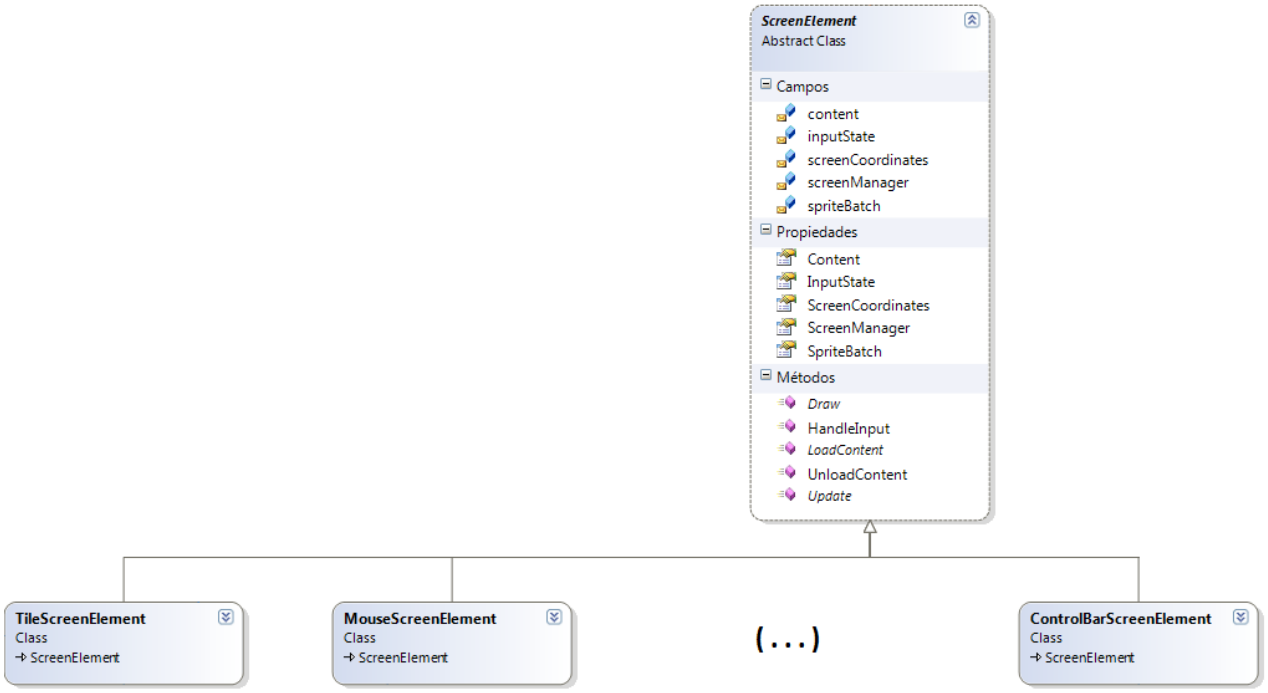

Figura 75. Clase ScreenElement

Como clase abstracta, ScreenElement define los elementos comunes a todas las subclases:

- Campos:
  - content: instancia de ContentManager utilizada para cargar los recursos en memoria en tiempo de ejecución.
  - screenCoordinates: rectángulo de pantalla del que se ocupa el ScreenElement, determinando dónde dibujarse y qué entradas de usuario deben ser atendidas.

- inputState: instancia de InputState que contiene la entrada de usuario actualizada. Esta instancia será consultada siempre que sea posible, ya que GameplayScreen habrá configurado los límites de acción de cada ScreenElement.
- screenManager: instancia de ScreenManager que contiene los datos sobre la ventana y el subsistema gráfico.
- spriteBatch: instancia de SpriteBatch que se utiliza para dibujar en pantalla el contenido visual del ScreenElement.

- Métodos:
  - HandleInput: este método es especialmente relevante, ya que debe adquirir la entrada de usuario actualizada, considerar las dimensiones de pantalla y calcular si/cómo debe reaccionar (por ejemplo, cómo debe ser el cursor del ratón).
  - El resto de métodos son los propios de un componente de XNA, por lo que no serán analizados en profundidad en esta clase.

4.3.6. Clase ControlBarScreenElement

Esta clase modela la barra de control, lugar donde se sitúan el mapa en miniatura, la información sobre la selección actual y las órdenes que es posible dar.

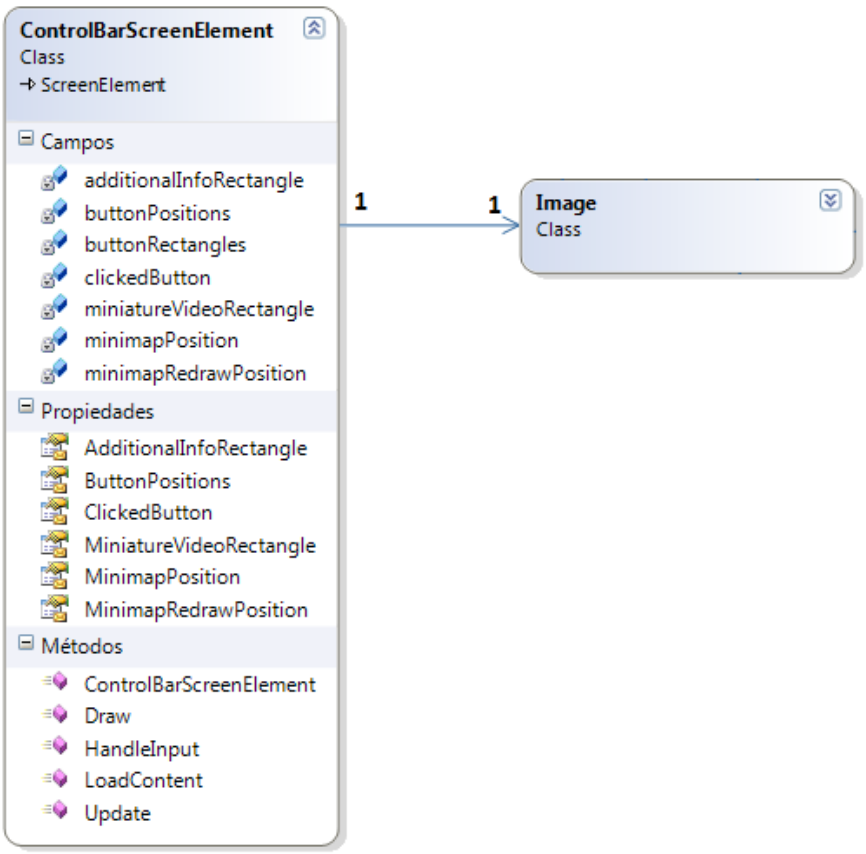

Figura 76. Clase ControlBarScreenElement

Esta clase contiene todos los campos de los que consta su superclase, ScreenElement, así como las siguientes variables y métodos:

- Campos:
    - additionalInfoRectangle, buttonRectangles, minimapPosition: son los rectángulos que definen la posición en pantalla de los elementos de la barra de control. Sirven para poder reaccionar de manera concreta a las entradas de usuario y para dibujar los elementos en el lugar apropiado.
    - clickedButton: si se determina que un botón ha sido pulsado, esta variable recogerá el resultado. Después de esto, será tarea de GameplayScreen el trasladar esa información de manera que se lleve a cabo la orden solicitada.
    - miniatureVideoRectangle: esta zona de pantalla es donde se dibuja un pequeño retrato de la unidad que se ha seleccionado. El nombre, "video", está tomado de la especificación del juego original, donde este elemento era tomado como un "enlace de vídeo" con la unidad.
    - controlBar: instancia de la clase Image que contiene los datos relativos a la imagen de fondo que da cuerpo a la barra de control.
- Métodos:
    - HandleInput: este método determina si se acaba de presionar el botón de acción principal sobre la zona del mapa o sobre alguna de las órdenes.

4.3.7. Clase MouseScreenElement

Esta clase modela y gestiona el puntero del ratón. Dada la orientación a imágenes de XNA, no es de extrañar que el ratón esté desactivado por defecto. Es más, tampoco es gestionado directamente por el framework. Si se quiere realizar un control exhaustivo sobre este elemento se tienen dos opciones: o bien se trabaja directamente con las librerías de .NET, o bien se aprovecha la gestión de entradas de XNA para crear un puntero de ratón.

En lo concerniente a este proyecto se ha optado por la última de las dos soluciones. La primera solución parecía muy engorrosa y, sobre todo, no se ajusta al modelo de trabajo con imágenes que se ha seguido durante todo el proyecto.

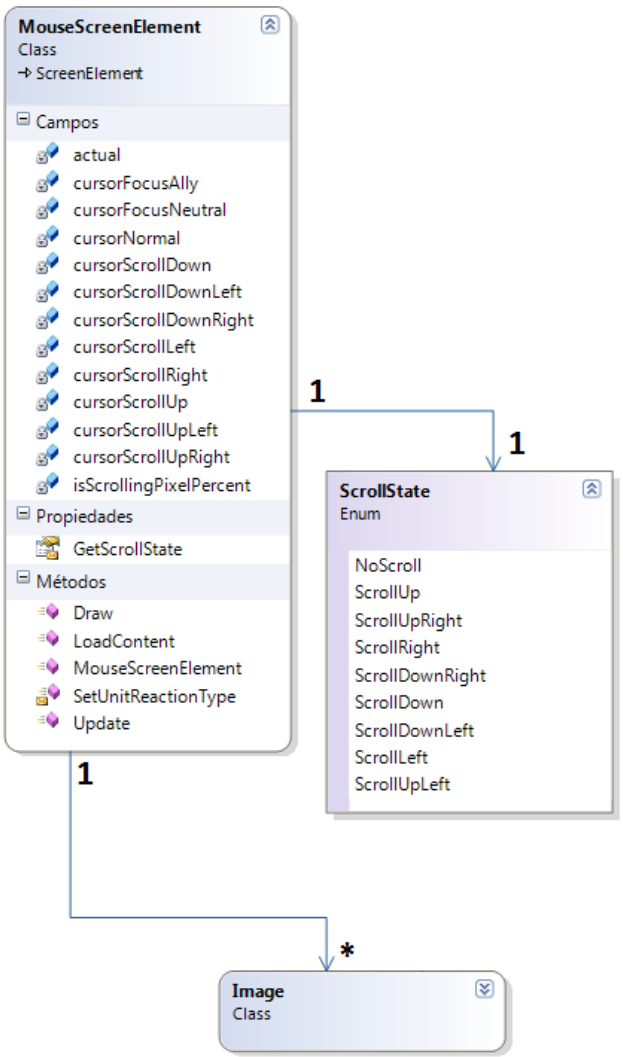

Figura 77. Clase MouseScreenElement

En definitiva, MouseScreenElement representa el puntero de ratón que se observa en la pantalla de juego en todo momento. Esta clase se ha diseñado para ser la

última en ser dibujada; de esta manera el puntero del ratón se ve siempre por encima de los demás objetos. Además, esta clase variará el puntero utilizado según sea el elemento que tenga “debajo”: el borde del mapa, una unidad… A continuación se describen los campos y métodos más representativos:

- Campos:
    - actual: mantiene la referencia a la clase Image que será dibujada en pantalla.
    - cursorScroll*: mantienen referencias a los cursores que indican que el mapa se está desplazando en uno de los posibles sentidos. Estos cursores serán dibujados cuando estén cerca de los bordes del mapa.
    - cursorFocus*: mantienen referencias a los cursores que resaltan el bando de la unidad que se encuentra “debajo” del puntero.
    - isScrollingPixelPercent: variable que indica si el puntero se encuentra tan cerca de un borde de pantalla como para desplazar el mapa.
    - GetScrollState: propiedad calculada que indica en qué dirección debe desplazarse el mapa en base a la posición del cursor en pantalla. Será responsabilidad de GameplayScreen transmitir este mensaje al mapa para producir el efecto de desplazamiento.
- Métodos:
    - SetUnitReactionType: método utilizado por GameplayScreen para indicar a esta clase que tiene una unidad de un bando concreto justo debajo.

4.3.8. Clase MapScreenElement

Esta clase modela el tablero de juego, también conocido como mapa. Este mapa estará compuesto por casillas y tendrá en su superficie las unidades y los edificios que estén en juego.

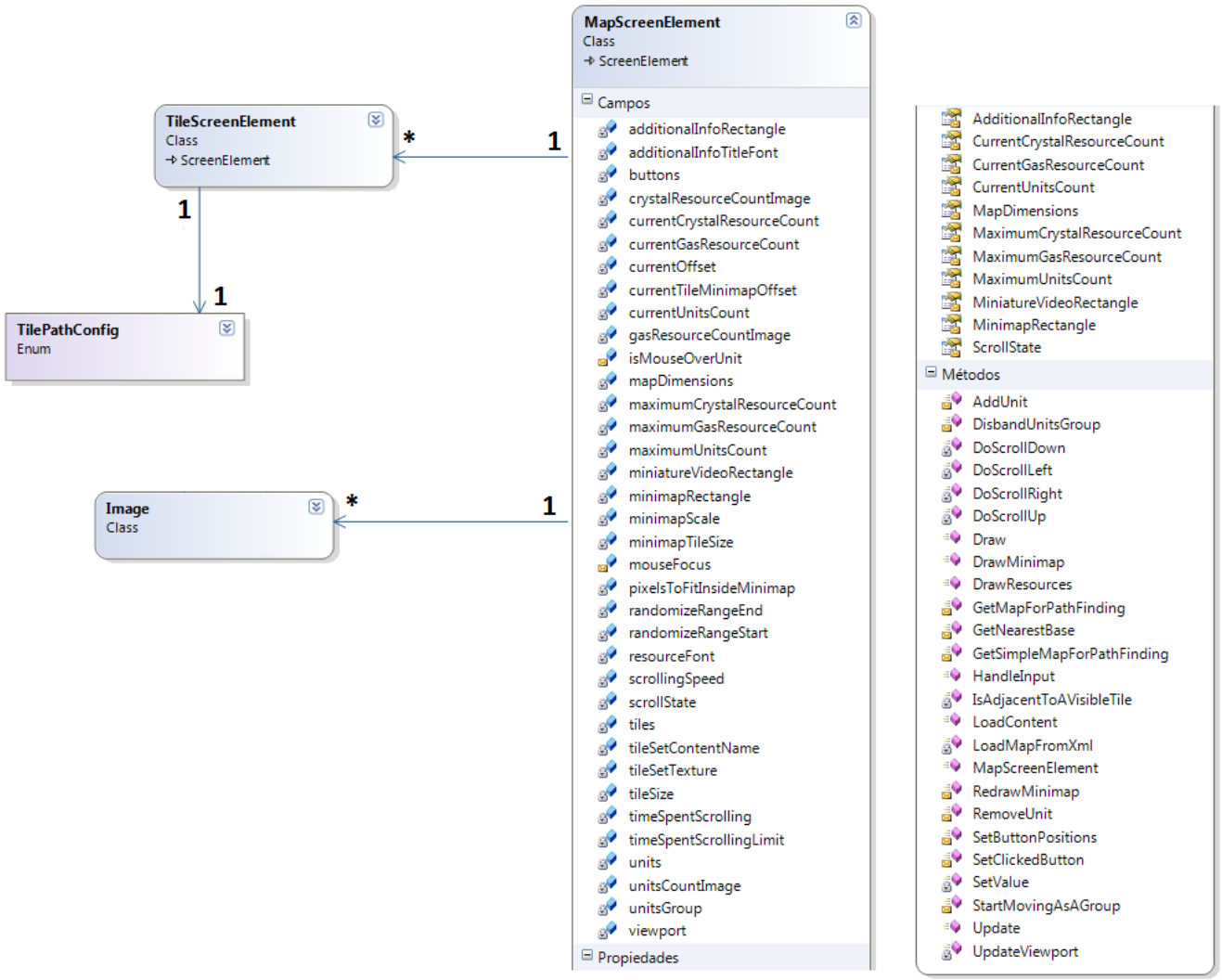

Figura 78. Clase MapScreenElement

Esta clase tiene su origen en la lectura del fichero que especifica el mapa. Después creará las casillas (instancias de TileScreenElement) y las configurará. Más tarde, creará las unidades y edificios iniciales.

Al terminar la inicialización, el jugador podrá observar el tablero de juego formado e interactuar con él. Esta clase llevará registro de las unidades, ya que su posición estará expresada en función de la casilla del mapa en la que se encuentren. De la misma manera, esta clase también llevará registro de los recursos del jugador, permitiendo o denegando las operaciones que se soliciten.

Esta clase también tiene la responsabilidad de desplazar el punto de vista sobre el mapa según sea necesario. Si el usuario interactúa con el mapa en miniatura, GameplayScreen trasladará esta operación a MapScreenElement, que deberá desplazar el punto de vista de igual manera. Si el usuario acerca el cursor del ratón a los bordes de la pantalla, MouseScreenElement avisará a GameplayScreen, que trasladará la petición de movimiento de mapa a esta clase. Y, finalmente, si el usuario activa la acción por defecto sobre un punto vacío del mapa, esta clase entenderá que debe centrar el mapa en ese punto, por lo que actuará en consecuencia.

A continuación se detallan los campos y métodos más significativos de la clase:

- Campos:
    - *ResourceCount: se disponen estos campos para controlar las cantidades actuales y máximas de recursos disponibles (gas, cristal y población).
    - mapDimensions: almacena las dimensiones del mapa (número entero de casillas de alto y número entero de casillas de ancho).
    - minimapRectangle, minimapScale y pixelsToFitInsideMinimap: campos utilizados para determinar la posición y dimensiones del mapa en miniatura. Serán utilizados cuando GameplayScreen decida que es necesario repintar dicho mapa en miniatura.
    - randomizeRange*: campos que indican las casillas a utilizar cuando el fichero de mapa no especifique un valor para una casilla concreta.
    - resourceFont: campo que almacena la referencia a la fuente utilizada para dibujar los recursos del jugador en pantalla.
    - scrollState y scrollingSpeed: campos que determinan el movimiento del punto de vista sobre el mapa y la velocidad de dicho desplazamiento.
    - timeSpentScrolling*: variables que moderan la velocidad de desplazamiento. En lugar de moverse X píxeles cada vez, el mapa se moverá a X unidades por segundo, redondeando al valor menor (normalmente 0 o 1 píxeles). De esta manera se evitará que el mapa aparezca cortado o con saltos bruscos.
    - tileSetContentName: almacena el nombre del fichero gráfico de casillas, que será utilizado para dibujar las casillas del mapa en pantalla.
    - units: almacena las instancias de UnitScreenElement que se encuentran activas actualmente.
    - viewport: almacena el punto de vista actual sobre el mapa.
- Métodos:
    - AddUnit: método que añade una unidad a la colección de unidades. Comprobará si es posible realizar esta operación en base a los recursos disponibles.
    - DoScroll*: métodos que determinan cuántos píxeles es necesario mover el punto de vista sobre el mapa. Los valores esperados son 0, 1 o, como mucho, 2, ya que se pretende un desplazamiento del mapa muy suave.

- Draw*: métodos de dibujado del mapa, del mapa en miniatura y de los recursos.
- GetMapForPathFinding y GetSimpleMapForPathFinding: métodos que obtienen una representación del mapa actual válida para la búsqueda de caminos. Para ello añaden las casillas bloqueantes, marcan como inválidas las casillas ocupadas por otras unidades…
- GetNearestBase: método que devuelve la posición de la base más cercana a la unidad que invoca este método. Será útil para la operación de recolección de recursos.
- IsAdjacentToAVisibleTile y RedrawMinimap: métodos utilizados para obtener qué casillas conforman los bordes del punto de vista actual. Estas casillas estarán resaltadas en el mapa en miniatura para orientar al jugador sobre la parte del mapa que está viendo.
- LoadMapFromXml: método que lee el fichero de especificación de mapa y crea las instancias de TileScreenElement necesarias. Si alguna casilla no estuviera especificada en el fichero, creará una instancia aleatoria dentro del rango indicado.
- RemoveUnit: método que extrae un UnitScreenElement de la lista units. Utilizado cuando una unidad o un edificio es destruido.
- SetButtonPositions y SetClickedButton: métodos de interacción con los botones de órdenes posibles. El primero de ellos configura los botones que deben ser mostrados actualmente, mientras que el segundo es llamado cuando uno de esos botones es activado por el jugador.
- UpdateViewport: método que determina si es necesario cambiar el punto de vista sobre el mapa en base a la entrada de usuario actual.

4.3.9. Clase TileScreenElement

Esta clase modela una casilla del mapa (recuérdese que el mapa estará compuesto por casillas, como un mosaico). Esta clase tiene, por tanto, la responsabilidad de gestionar todos los aspectos relacionados con una casilla individual del mapa.

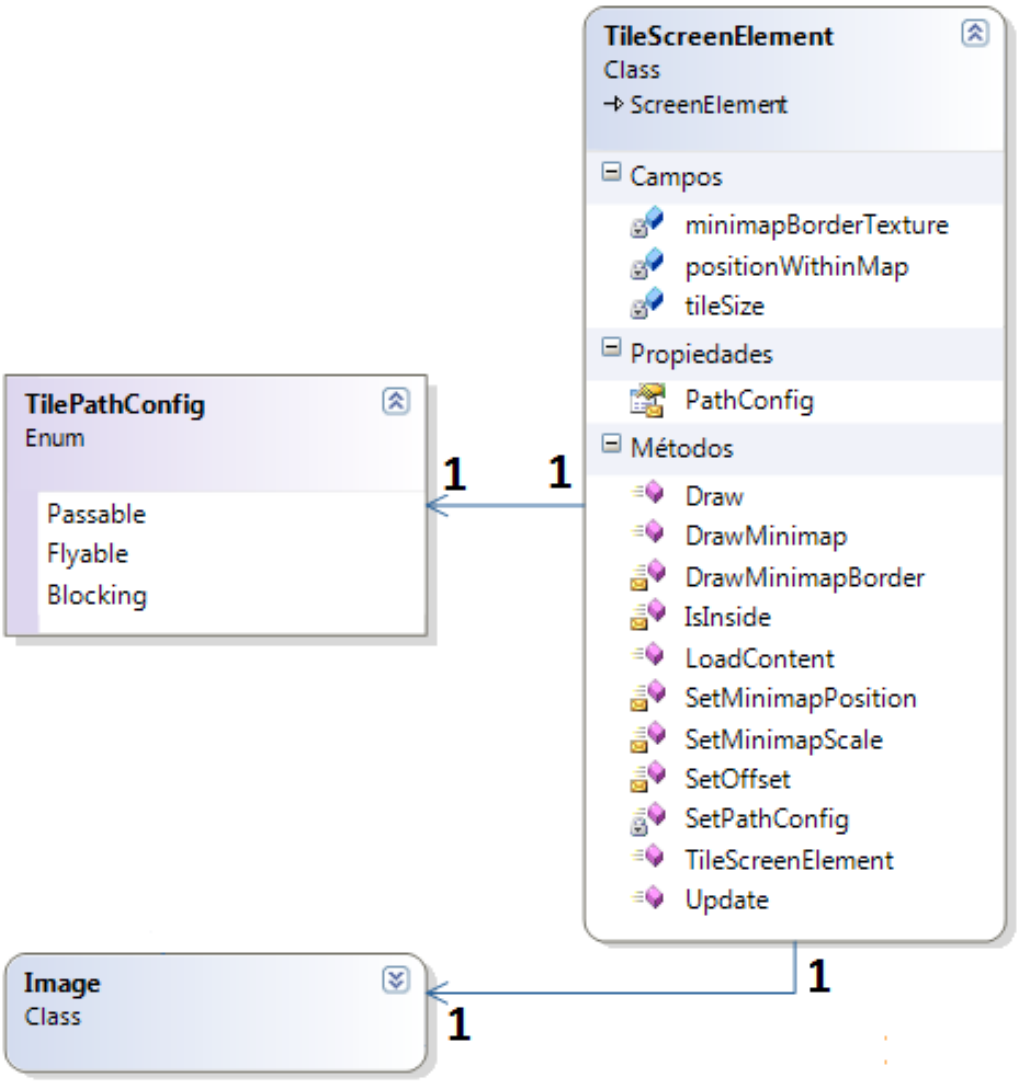

Figura 79. Clase TileScreenElement

- Actualización: si la casilla fuera animada, entonces deberá actualizar la imagen que muestra correctamente.
- Dibujado: la casilla debe dibujarse en la posición correcta de la pantalla. Esta posición dependerá del punto de vista actual sobre el mapa. También deberá dibujarse a sí misma en el mapa en miniatura.

- Aspectos de gestión:
    - Esta clase deberá dibujar una imagen diferente de sí misma si forma parte del borde de pantalla. De esta manera se resaltará el punto de vista actual en el mapa en miniatura.
    - Esta clase deberá informar a la instancia de MapScreenElement acerca de si se puede atravesar la casilla o no, ya que este dato es de vital importancia a la hora de calcular caminos sobre el mapa.

A continuación se describen los elementos más significativos de esta clase:

- Campos:
    - minimapBorderTexture: imagen utilizada para dibujar la casilla en el mapa en miniatura cuando ésta forma parte del borde visible de la pantalla global. Esta textura será de un único color muy vistoso.
    - positionWithinMap: coordenadas de la casilla en el mapa (X e Y).
    - tileSize: tamaño en píxeles de la casilla.
- Propiedades:
    - PathConfig: propiedad que almacena si esta casilla es atravesable o no. Este dato será de vital importancia para calcular los caminos que seguirán las unidades al desplazarse por el mapa.
- Métodos:
    - Draw: método utilizado para dibujar la casilla a tamaño real.
    - DrawMinimap: método utilizado para dibujar la casilla a escala, ya que deberá ser dibujada en el mapa en miniatura.
    - IsInside: método utilizado para determinar si la casilla es actualmente visible o si no forma parte del punto de vista actual sobre el mapa.
    - SetMinimapPosition: método que determina la posición actual del mapa en miniatura en la pantalla de juego.
    - SetMinimapScale: método que configura la escala a la que se debe dibujar esta casilla en el mapa en miniatura. Dependiendo del tamaño del mapa global, este valor será mayor o menor.
    - SetOffset: método que indica a esta casilla el estado del punto de vista sobre el mapa. Este dato será el desplazamiento de la casilla origen sobre el píxel (0, 0) de la pantalla.

4.3.10. Clase UnitScreenElement

Esta clase modela las unidades y los edificios que forman el ejército de los jugadores. La mayor diferencia entre los dos conceptos es la velocidad de movimiento, aunque cada uno presenta matices que lo diferencian del resto. Por ejemplo, una unidad que ataque no tendrá valores en las variables de recolección, o los edificios no tendrán valores en las variables de ataque.

Una vez comentado el propósito de diseño, se ha de notar que esta clase contiene una enorme cantidad de información: más de 4000 líneas de código. Esto se debe al gran número de unidades, edificios y órdenes implementado. Sin embargo, se ha procurado aislar la lógica de cada proceso en sus respectivos métodos, por lo que es posible estudiar las funcionalidades de manera individual.

A continuación se describen los campos y métodos más importantes de esta clase:

- Campos:
    - additionalInfoFont, additionalInfoTitleFont…: todos los campos terminados en "Font" son las referencias a las fuentes utilizadas para dibujar unos u otros textos en pantalla: información adicional, progreso de la orden actual…
    - buttonPositions y buttons: campos que almacenan respectivamente la posición en pantalla y la imagen de los botones de acción.
    - clickableLimits: es el rectángulo de pantalla en el que se encuentra esta unidad. Todas las entradas sobre el mapa que ocurran dentro de ese espacio serán manejadas por esta unidad.
    - *ResourceAvailable: cantidad de recursos disponibles. Este dato será necesario para permitir o denegar la construcción de un edificio o la producción de una unidad.
    - currentHeading: indica la orientación actual de la unidad. Cada orientación tendrá, dentro de lo posible, una imagen en particular, por lo que los giros de las unidades podrán ser vistos en detalle.
    - currentHealthPoints: puntos de vida que tiene la unidad actualmente. Cuando este dato sea cero, la unidad morirá, dejando de ser dibujada en pantalla y quedando totalmente inactiva.
    - currentlyCarried*: cantidad de recursos que transporta la unidad. Este dato será significativo para las unidades de recolección de recursos.
    - cyclicPath: lista de casillas del mapa a recorrer de manera cíclica. Esta variable será utilizada cuando la unidad deba recorrer un camino de ida y vuelta (por ejemplo, durante el proceso de recolección).
    - damageDone: cantidad de puntos de vida que la unidad sustrae en cada ataque realizado.

- *SoundEffects, *SoundEffect: variables que almacenan los sonidos a reproducir. Existirán sonidos para ilustrar la selección de una unidad o edificio, los disparos realizados, la muerte de la unidad o edificio y la aceptación o rechazo de una orden concreta.
- doneMining: indica si la unidad ha terminado el proceso de obtener recursos. Este proceso es la mitad del proceso de recolección, ya que luego se debe depositar lo obtenido en un edificio de tipo base. Esta variable sirve para conocer el estado actual del proceso de recolección de recursos.
- informationalUnitText: campo que almacena el texto que muestra la unidad cuando es seleccionada: por ejemplo, el nombre de la unidad, su tipo…
- isAttacking, isBeingHit, isBeingProduced…: campos que indican si se está realizando una orden concreta.
- isSelected: campo que indica si la unidad está seleccionada actualmente. Será utilizado para determinar si se debe pintar información adicional, si se deben pintar las órdenes y qué acción se espera cuando se activa la entrada por defecto.
- lastUpdateTime: almacena el instante de la última llamada al método update(). Sirve para trasladar el tiempo transcurrido en la vida real dentro del juego. Es útil, por tanto, en un proceso de construcción que tarda unos minutos, en un proceso de recolección que tarda unos segundos, en la velocidad de movimiento de las unidades…
- map: almacena la instancia de MapScreenElement en la que se encuentra la unidad. Será utilizada para obtener la información necesaria para calcular el camino entre dos casillas.
- marker: imagen que ilustra la selección de la unidad. Cuando la unidad sea seleccionada, se dibujará esta imagen por debajo, ilustrando el proceso y dejando patente el bando de la unidad.
- miniatureVideoRectangle: posición de la pantalla donde se ha de dibujar la cara del piloto de la unidad.
- minimapScale: escala a la que se está dibujando el mapa en miniatura. Este dato servirá para dibujar la unidad en dicha miniatura con la proporción correcta.
- movementSpeed: cantidad de píxeles que la unidad es capaz de avanzar como máximo. Esta magnitud es proporcional al tiempo pero, como esta medición de tiempo depende de la cadencia de llamadas a update(), la rapidez que presenta la unidad en pantalla depende de la velocidad que XNA imponga a la aplicación.
- path: camino que la unidad ha de seguir. Este camino estará compuesto por una casilla de inicio, una casilla de fin y un conjunto conexo de casillas entre la de inicio y la de fin. Se almacena la referencia (X, Y) de la casilla en lugar de la instancia completa.

- targetedBase, targetedMine, targetedUnit: variables que almacenan las instancias de UnitScreenElement que intervienen en los procesos señalados.
- tileWithinMap: coordenadas de la casilla (X, Y) del mapa en la que se encuentra la unidad.
- timeNeededToMine: tiempo que la unidad necesita estar en el edificio de tipo recolección o de tipo base para completar con éxito su tarea (obtener/descargar recursos).
- timeSpent*: variables que almacenan el tiempo transcurrido desde el inicio de una actividad. Estas variables son clave a la hora de completar los procesos a los que hacen referencia. Por ejemplo, timeSpentBuilding determina el progreso de la construcción del edificio en curso.
- unit, unitBase: campos que almacenan las imágenes de la unidad. Si la unidad es simple, la imagen será "unit". Si la unidad está compuesta por una base y una imagen sobreimpresionada, se utilizará primero "unitBase" y luego "unit".
- unitBeingAttacked, unitBeingBuilt…: campos que almacenan las instancias de UnitScreenElement con las que se trabaja en los procesos a los que se hace referencia. Estas variables serán útiles a la hora de informar a los objetivos de algún suceso importante. Por ejemplo, si una unidad dispara a otra, deberá quitarle una cantidad de puntos de vida; en último caso, además, deberá notificar a su objetivo que ha muerto.
- UnitProduction*Cost: campos que almacenan el coste de producción de las unidades y edificios. Estas variables serán útiles al comenzar los procesos de construcción de unidades y edificios.
- unitSizeInTiles: tamaño de la unidad en términos de casillas de mapa ocupadas. Esta variable se utilizará a la hora de calcular el camino a seguir, ya que se desea evitar en la medida de lo posible las colisiones visuales.
- unitTextureHeightAndWidth: tamaño de la imagen de la unidad en píxeles. Este campo será utilizado para determinar si la unidad debe responder a una entrada de usuario o, por el contrario, la entrada de usuario está fuera de los límites apropiados.

- Propiedades:
  - UnitType: tipo de la unidad o edificio. Indica de qué unidad se trata: "BattleCruiser", "SCV", "Base"…
  - UnitReactionType: bando al que pertenece la unidad.
  - UnitMovementType: indica el movimiento natural de la unidad. Las posibilidades son caminar ("Walking"), volar ("Flying") y ninguno ("None").

Figura 80. Clase UnitScreenElement

- Métodos:
    - CalculatePathBetween, CalculateSimplePathBetween: métodos que calculan el camino más corto entre dos puntos del mapa. El segundo método es menos restrictivo que el primero, ya que permite cierto grado de colisión visual.
    - DoStopMoving, DoStopResourceRecollection: métodos que detienen los procesos a los que hacen referencia. Actualizan la propia clase de manera que la unidad permanezca quieta tan pronto como sea posible.
    - DrawBeingHit, DrawFiring: métodos de dibujado que ilustran los combates. La utilización de estos métodos está condicionada al proceso de ataque y a la velocidad de disparo de cada unidad concreta.
    - DrawMinimap: método que dibuja la unidad a escala en el mapa en miniatura.
    - DrawAdditionalInformation: método de dibujado que se activa cuando la unidad está seleccionada. Su tarea es ilustrar la selección y dar información al usuario.
    - HandleInput: método que reacciona ante la entrada de usuario. Su tarea consiste en discriminar si se debe reaccionar y, si es necesario, comenzar la acción solicitada: selección, acción por defecto u orden concreta.
    - IsClickWithinLimits: método que determina si la entrada de usuario está dentro de los límites que gestiona la unidad. Será utilizado para determinar si la unidad ha sido seleccionada, si únicamente el cursor está pasando por encima o si la unidad no debe hacer caso de la entrad actual.
    - IsTargetWithinRange: método que determina si la unidad o edificio objetivo está dentro del rango de disparo. Este método se utilizará durante el proceso de ataque para determinar cuándo puede una unidad empezar a disparar.
    - IsValidLocationToBuild: método que determina si la unidad se encuentra en una posición válida para construir el edificio indicado. Este nuevo edificio deberá tener espacio para sí mismo y para las unidades que entren o salgan de él.
    - UnitButtonsFactoryMethod, UnitFactoryMethod: métodos que configuran la instancia de UnitScreenElement actual con los valores propios de una unidad o edificio concretos.
    - Update*: métodos que actualizan el estado de la unidad. En primer lugar se llamará al método Update(). Dependiendo del estado actual de la unidad, éste llamará a unos u otros métodos de actualización. Por ejemplo, si se detecta que la unidad tiene todavía un camino por recorrer, se llamará a UpdateHeading() y a UpdatePosition().

4.3.11. Clase Image

Esta clase agrupa la base tecnológica más importante del proyecto: la imagen que se presenta por pantalla ante el usuario. La lógica diseñada permite que esta clase controle totalmente cualquier imagen utilizada en el proyecto.

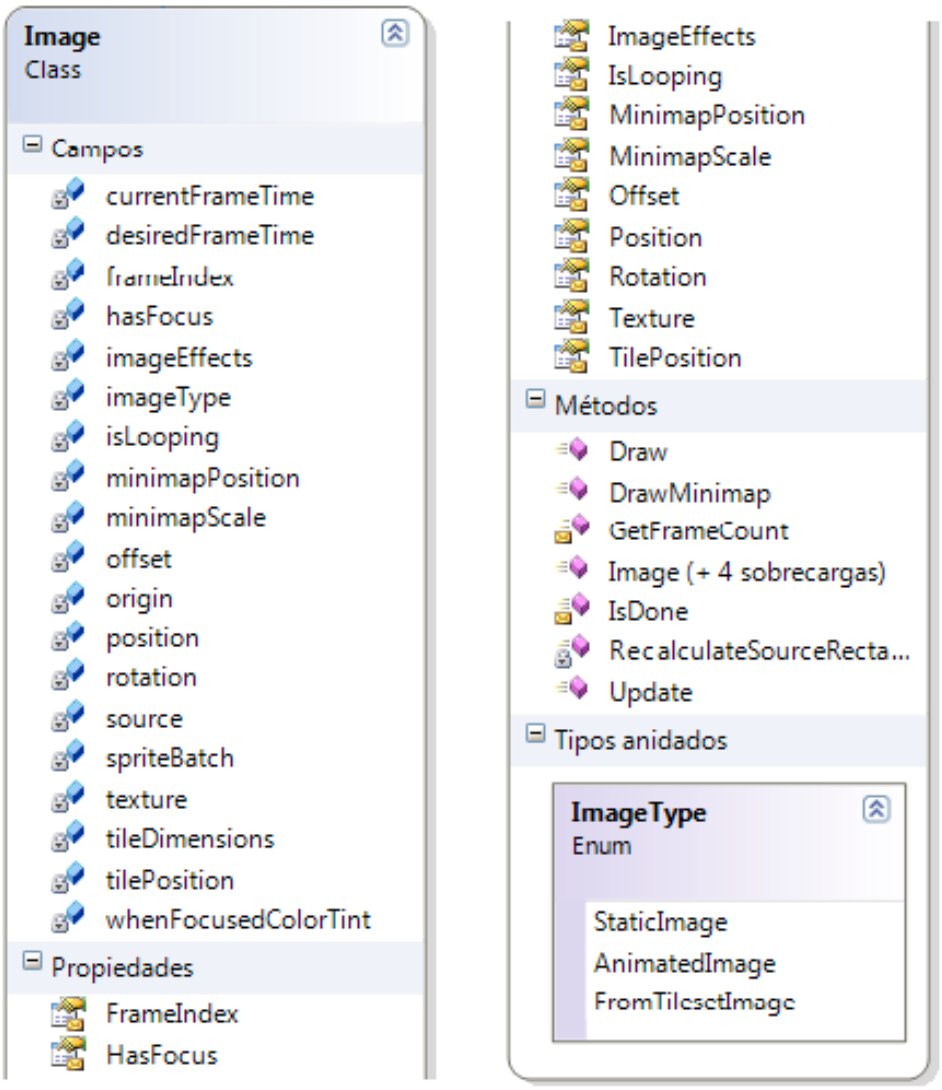

Figura 81. Clase Image

- Imagen estática: indicada por el campo "ImageType.StaticImage". Esta imagen se caracteriza por ocupar totalmente el archivo de datos en el que está almacenada. Además, se supone que no requiere actualización a lo largo del tiempo, por lo que únicamente es necesario cargar los datos y dibujarlos en pantalla.
- Imagen de mosaico: indicada por el campo "ImageType.FromTilesetImage". Esta imagen se caracteriza por compartir el archivo de datos con multitud de imágenes de dimensiones similares. Por tanto, será necesario indicar qué porción del archivo es la que ocupa la imagen deseada. El resto de aspectos es idéntico a lo que se supone en una imagen estática.

- Imagen animada: indicada por el campo “ImageType.AnimatedImage”. Esta imagen se caracteriza por compartir el archivo de origen con otras imágenes, pero todas pertenecientes a la misma serie. Se asume que la imagen forma parte de un conjunto dispuesto verticalmente, y se avanza dentro del conjunto según el tiempo especificado.

A continuación se detallan los campos y métodos más importantes de esta clase:

- Campos:
    - currentFrameTime, desiredFrameTime: controlan el tiempo que se muestra cada imagen cuando la imagen es animada.
    - frameIndex: almacena la imagen que se está mostrando dentro de las que componen la animación.
    - imageEffects: efectos de espejado aplicados a la imagen a la hora de dibujarla. Esta opción es útil cuando una imagen tiene una fuerte componente simétrica.
    - imageType: campo que discrimina el tipo de la imagen que maneja la instancia.
    - isLooping: indica si la animación que se muestra es cíclica.
    - origin: almacena el punto que actúa de origen al realizar algún tipo de efecto o rotación sobre la imagen.
    - position: indica la posición en pantalla a partir de la cual se debe empezar a dibujar la imagen.
    - rotation: almacena la rotación que se le aplicará a la imagen a partir de su punto de origen. Los valores serán interpretados angularmente.
    - source: campo que indica el rectángulo que se debe tomar como fuente de datos.
    - spriteBatch: instancia de SpriteBatch utilizada para dibujar en pantalla. Es parte de la abstracción que permite XNA sobre la tarjeta gráfica.
    - texture: referencia a la imagen cargada mediante ContentManager.
    - tileDimensions: cantidad de imágenes (X, Y) que se encuentran en el archivo de datos. Esta variable se utilizará para averiguar las dimensiones individuales de cada imagen cuando se indique que esta clase debe trabajar con una imagen de mosaico.
    - tilePosition: campo que indica la baldosa concreta a utilizar (X, Y) dentro del mosaico de imágenes que compone el archivo de datos de origen.
- Métodos:
    - GetFrameCount: método que calcula el número de imágenes que hay en el archivo de imagen animada.
    - IsDone: método que determina si la animación ha terminado.
    - RecalculateSourceRectangle: método que actualiza el campo “source” cuando la animación deba avanzar a la siguiente imagen.

Capítulo 5

Fase de implementación

Este capítulo recoge los aspectos técnicos del desarrollo, detallando la forma concreta que han tomado las ideas de diseño y las particularidades encontradas en el proceso.

El primer apartado, **5.1. Introducción**, detalla el propósito del capítulo actual e incluye un pequeño resumen de cada apartado.

El segundo apartado, **5.2. Imágenes**, contiene la abstracción realizada sobre las imágenes como aspecto básico de la aplicación.

El tercer apartado, **5.3. Dibujado en pantalla**, explica el proceso de pintado que se utiliza a lo largo del desarrollo.

El cuarto apartado, **5.4. Mapa**, presenta todo lo relativo a este concepto: definición, dibujado y desplazamiento.

Por último, el quinto apartado, **5.5. Unidades y edificios**, presenta todo lo relativo a estos dos conceptos: imágenes utilizadas, detección de colisiones, operaciones de movimiento y orientación, búsqueda de caminos y órdenes.

5.1. Introducción

Este capítulo se dedica a mostrar la implementación que finalmente han recibido las ideas originadas en las etapas de análisis y de diseño. La finalidad de este capítulo no es comentar cada línea de código: para ello se dispone de los comentarios del propio código. La función de este capítulo es profundizar en los procesos que tienen lugar durante la fase de desarrollo, detallando las decisiones tomadas a alto nivel.

Como resultado, se tratará la clase Image, base de todo el proyecto; se hablará del proceso de dibujado en pantalla, que permite entender la abstracción realizada y las funciones implementadas; se definirán en detalle los mapas; para concluir, se comentarán las unidades y los edificios junto con su movimiento y órdenes.

5.2. Imágenes

El punto de partida de este capítulo no podía ser otro que el elemento más básico de los que componen la jerarquía del sistema: la imagen. Esta imagen puede ser de diversos tipos. Todos los demás elementos del programa están orientados, directa o indirectamente, a producir la imagen adecuada en el momento preciso y en la posición concreta.

Desde el primer momento se quiso abstraer el trabajo con imágenes, de manera que trabajar con ellas fuera más sencillo, directo y dinámico. Esta motivación dio lugar a la clase Image, cuya estructura se ofreció en el capítulo anterior. Esta clase es capaz de tomar la referencia de una imagen, cargar los datos necesarios y ajustar la configuración. A partir de entonces, utilizar la imagen es tan sencillo como llamar a update(), y dibujarla simplemente consiste en llamar a draw().

A continuación se muestra el código necesario para crear una imagen estática.

```
public Image(SpriteBatch spriteBatch, Texture2D texture, Vector2
position, Color whenFocusedColorTint)
  {
    this.spriteBatch = spriteBatch;
    this.texture = texture;
    this.imageType = ImageType.StaticImage;
    this.position = position;
    this.offset = new Vector2(0, 0);
    this.whenFocusedColorTint = whenFocusedColorTint;
    this.origin = new Vector2(0, 0);
    this.source = new Rectangle(0, 0, texture.Width, texture.Height);
    this.hasFocus = false;
    this.imageEffects = SpriteEffects.None;
    this.rotation = 0.0f;
  }
```

Código 1: Creación de imágenes estáticas

Una imagen estática es aquella que está destinada a mostrarse a tamaño completo en la ventana de la aplicación. Los valores de offset y origin, por tanto, son cero. El valor de source serán los límites de ancho y alto que se indiquen en el archivo de datos. A continuación se muestra un ejemplo de imagen estática.

Figura 82. Imagen estática

La clase Image no tiene conciencia sobre el origen de datos: texture2D debe tener un valor previo a la construcción de la clase. Se ha decidido dejar la especificación y carga para las superclases dado que, de no hacerlo, se sobrecargaría enormemente Image y se haría más difícil cambiar aspectos puntuales. Además, si cada clase carga sus propios datos, los procesos de carga puntuales son más leves y están más orientados al contenido que se quiere mostrar.

Otro aspecto notable de Image es que spriteBatch debe tener un valor antes de construir un objeto utilizable. Este campo es la instancia de SpriteBatch utilizada a lo largo de todas las instancias del programa. Su función es operar con el subsistema gráfico, borrando, dibujando o modificando lo previamente dibujado. Aunque este modelo de instancia única no se impone por parte de XNA, es muy recomendable seguirlo: si únicamente hay una instancia de dibujado, no habrá conflictos en dicho proceso. Además, comenzar y detener el proceso de dibujado requiere una cantidad de computación no despreciable a gran escala.

Aparte de las imágenes estáticas ya comentadas, esta clase abstrae el manejo de dos tipos más: imágenes animadas e imágenes de mosaico. La diferencia principal entre estos dos grupos es que, mientras que el primero es intuitivo y natural, el

segundo es particular del caso que ocupa al presente proyecto. A continuación se destacan otras diferencias igualmente relevantes.

```
public Image(SpriteBatch spriteBatch, Texture2D texture, Vector2
position, Color whenFocusedColorTint, float desiredFrameTime)
  {
    this.spriteBatch = spriteBatch;
    this.texture = texture;
    this.imageType = ImageType.AnimatedImage;
    this.position = position;
    this.offset = new Vector2(0, 0);
    this.whenFocusedColorTint = whenFocusedColorTint;
    this.origin = new Vector2(this.texture.Width / 2.0f,
this.texture.Width / 2.0f);
    this.desiredFrameTime = new
TimeSpan((long)(desiredFrameTime*10000000));
    currentFrameTime = new TimeSpan(0);
    frameIndex = 0;
    this.source = new Rectangle(0, frameIndex * texture.Width,
texture.Width, texture.Width);
    this.hasFocus = false;
    this.imageEffects = SpriteEffects.None;
    this.rotation = 0.0f;
    this.isLooping = true;
  }
```

Código 2: Creación de imágenes animadas

El Código 2 permite que Image controle una imagen animada de la forma esperada: cada cierto tiempo, se dibujará una parte nueva de la imagen, dando sensación de animación.

Este código se ha preparado especialmente para el proyecto actual, ya que ha sido necesario adaptarse a los recursos ofrecidos en la creación original StarCraft. Estos recursos tienen la particularidad de ser un conjunto de imágenes almacenadas en vertical.

La clase Image controla el tiempo de dibujado de cada imagen (o “frame”). Cuando ese temporizador expira se actualizan los campos de la instancia para dibujar el siguiente frame. A continuación se exponen varios ejemplos de imágenes animadas.

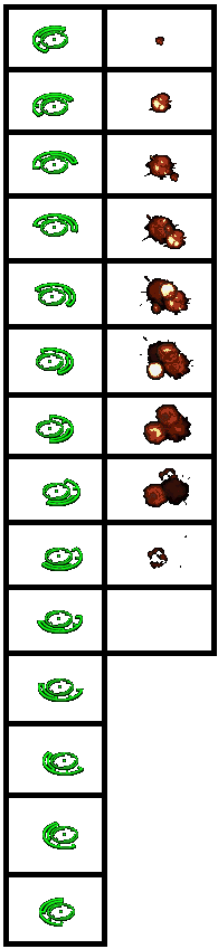

Figura 83. Imágenes animadas

En la Figura 83 se observan los dos tipos de imágenes animadas, señalando los frames que las componen. A la izquierda se muestra un cursor, que es una imagen animada cíclica. A la derecha se muestra una explosión, que es una imagen animada con un inicio y un final. El color blanco denota el color transparente: en todas las imágenes utilizadas hay un color que indica a XNA que no debe dibujar, sino que debe mantener el píxel con la tonalidad que actualmente tenga.

Como se enunció anteriormente, cada imagen animada posee sus propias características: ancho y alto de frame (píxeles), tiempo de exposición de frame (milisegundos), etc. Estos aspectos son recogidos por la clase Image, y serán tenidos en cuenta al ejecutar el método update().

A continuación se presenta el último tipo de imagen: aquella denominada “de mosaico”.

```
public Image(SpriteBatch spriteBatch, Texture2D texture, Vector2
position, Color whenFocusedColorTint, Vector2 tilePosition, Vector2
tileDimensions)
  {
    this.spriteBatch = spriteBatch;
    this.texture = texture;
    this.imageType = ImageType.FromTilesetImage;
    this.position = position;
    this.offset = new Vector2(0, 0);
    this.whenFocusedColorTint = whenFocusedColorTint;
    this.origin = new Vector2(0, 0);
    this.tilePosition = tilePosition;
    this.tileDimensions = tileDimensions;
    this.source = new Rectangle((int)(tilePosition.X *
tileDimensions.X), (int)(tilePosition.Y * tileDimensions.Y),
     (int)(tileDimensions.X), (int)(tileDimensions.Y));
    this.hasFocus = false;
    this.imageEffects = SpriteEffects.None;
    this.rotation = 0.0f;
  }
```

Código 3: Creación de imágenes de mosaico

La característica básica de las imágenes “de mosaico” es que comparten el origen de datos con multitud de imágenes de las mismas dimensiones. La clase Image debe ser informada de la disposición de imágenes en el archivo, así como de la imagen concreta que se quiere dibujar. A continuación se presenta un ejemplo de este tipo de imagen.

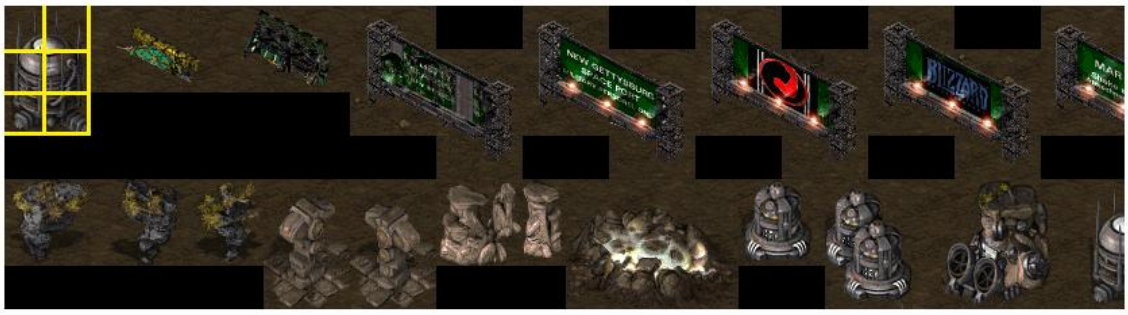

Figura 84. Imágenes de mosaico

En la Figura 84 se observa, marcado especialmente, el objeto de abstracción de la clase Image. El mismo archivo de datos es compartido por multitud de imágenes: de ahí el apelativo “mosaico”. La clase Image necesita conocer las dimensiones de las baldosas (ancho y alto), que se suponen iguales entre sí, y la posición de la baldosa escogida (X e Y, interpretados en orden de lectura). Con estos datos será capaz de dibujar el contenido seleccionado en la región de pantalla adecuada.

Multitud de instancias de Image de tipo mosaico pueden componer cualquier mosaico arbitrario. Esto da lugar, por ejemplo, a gran variedad de mapas. El espacio ocupado por estos mapas es mínimo: sólo se almacena una copia del conjunto de baldosas completo. Las baldosas individuales son referencias a la posición relativa de cada una.

Como última particularidad del tratamiento de imágenes, se quiere hacer hincapié en las zonas negras de la Figura 84. Estas regiones corresponden a baldosas vacías, actualmente sin uso concreto. Se ha seleccionado este único tono al crear el conjunto de baldosas para favorecer la compresión del origen de datos y la carga en memoria del mismo.

A continuación se ofrece el código del método update(), donde se aprecia el tratamiento que da la clase Image a cada uno de los tipos de imagen comentados.

```
public void Update(GameTime gameTime)
{
  if (imageType == ImageType.AnimatedImage)
  {
    currentFrameTime += gameTime.ElapsedGameTime.Duration();
    if (currentFrameTime > this.desiredFrameTime)
    {
      currentFrameTime = new TimeSpan(0);

      //Cálculo del siguiente frame (cíclica o no)
      if (isLooping)
      {
        frameIndex = (frameIndex + 1) % GetFrameCount();
      }
      else
      {
        if ((frameIndex + 1) != GetFrameCount())
        {
          frameIndex++;
        }
      }

      source.Y = frameIndex * Texture.Width;
      source.Width = Texture.Width;
      source.Height = Texture.Width;
    }
  }
}
```

Código 4: Actualización de imágenes

5.3. Dibujado en pantalla

El dibujado de los recursos gráficos en pantalla es un aspecto que se considera de vital importancia. Cualquier videojuego tiene muchísimo trabajo en la parte de interfaz de usuario, y el que ocupa este proyecto no es una excepción. Además, XNA impone una forma muy concreta de trabajar con el subsistema gráfico, por lo que el diseño e implementación están sujetos a estas condiciones.

Todas las operaciones de dibujado se encuentran entre las siguientes sentencias:

```
spriteBatch.Begin(SpriteBlendMode.AlphaBlend);

// código de pintado, por ejemplo:
// spriteBatch.Draw(Texture, minimapPosition, source,
whenFocusedColorTint, rotation, origin, minimapScale, imageEffects,
0.0f);

spriteBatch.End();
```

Código 5: Uso de SpriteBatch

La primera sentencia realiza las operaciones necesarias para iniciar las peticiones de dibujado al subsistema gráfico. Estas operaciones incluyen vaciar zonas de memoria previamente ocupadas, pintar la pantalla de un color "inicial"… También se especifica un modo de operación: AlphaBlend. Este modo de operación indica que se quiere tener en cuenta el canal alpha (los píxeles transparentes) a la hora de dibujar la imagen.

La última sentencia finaliza la operación de dibujado. Las operaciones necesarias son, por ejemplo, calcular la imagen resultante, inicializar el envío de datos hacia la tarjeta gráfica o el monitor…

Y, por último, lo más importante del proceso: las operaciones de dibujado que el desarrollador necesite hacer. Se adjunta una llamada sencilla a Draw() a modo de ejemplo. Todas las operaciones de dibujado deben estar contenidas entre Begin() y End().

Además de la estructura de llamadas, XNA impone otra restricción al desarrollador: las operaciones de pintado se realizarán en el orden especificado. Mientras que en otros sistemas es posible especificar la posición de una orden de dibujado dentro del conjunto de órdenes, en XNA siempre se dibuja desde abajo hasta arriba, como podría hacer un pintor sobre su lienzo.

Como conclusión, al utilizar XNA está altamente recomendada una jerarquía de instancias, objetos y llamadas muy concreta. A continuación se muestra el código de dibujado de la pantalla de juego.

```
public override void Draw(GameTime gameTime)
{
  //Se dibujan los elementos de pantalla con la instancia de
SpriteBatch general
  //(se utiliza SpriteBlendMode.AlphaBlend para aprovechar como tales
los píxeles transparentes)
  SpriteBatch spriteBatch = ScreenManager.SpriteBatch;
  spriteBatch.Begin(SpriteBlendMode.AlphaBlend);

  //se dibuja el mapa en su posición y, luego, en la posición que
diga el minimapa
```

Código 6: Dibujado de la Pantalla de Juego (parte 1ª)

```
  map.Draw(gameTime);
  if (controlBar.MinimapRedrawPosition.X != 0)
  {
    map.RedrawMinimap(controlBar.MinimapRedrawPosition.X,
controlBar.MinimapRedrawPosition.Y);
  }

  //se dibuja la barra de control con su minimapa y botones
  controlBar.Draw(gameTime);
  map.DrawMinimap(gameTime);

  //se dibujan los recursos disponibles para el jugador
  map.DrawResources(gameTime);

  //para terminar, se dibuja el cursor y el volumen
  mouse.Draw(gameTime);
  DrawVolume(gameTime);

  spriteBatch.End();

  //...

}
```

Código 7: Dibujado de la Pantalla de Juego (parte 2ª)

A lo largo del Código 6 se inicia la operación de dibujado en pantalla. Esta operación concluye al final del método Draw() (Código 7). Entre ambas llamadas se producen invocaciones a los métodos Draw() de las instancias a las que tiene acceso GameplayScreen.

En primer lugar se solicita al mapa que se dibuje a sí mismo. El mapa es el elemento más bajo de la Pantalla de Juego. El mapa, por su parte, se ocupará de dibujar a las unidades de tierra, luego a los edificios, y luego a las unidades de aire.

Después se solicita a la barra de control que realice su operación de dibujado. Esta operación implicará obstruir parcialmente la visibilidad del mapa. Al concluir esta operación, la esquina inferior izquierda quedará dibujada con el contenido de la barra de control. Pero este contenido estará parcialmente completo: falta el mapa en miniatura. Por tanto, será la siguiente llamada de dibujado.

Para terminar se dibujan los recursos disponibles, el puntero del ratón y los valores de volumen de música y efectos de sonido. Todo el proceso queda ilustrado como sigue.

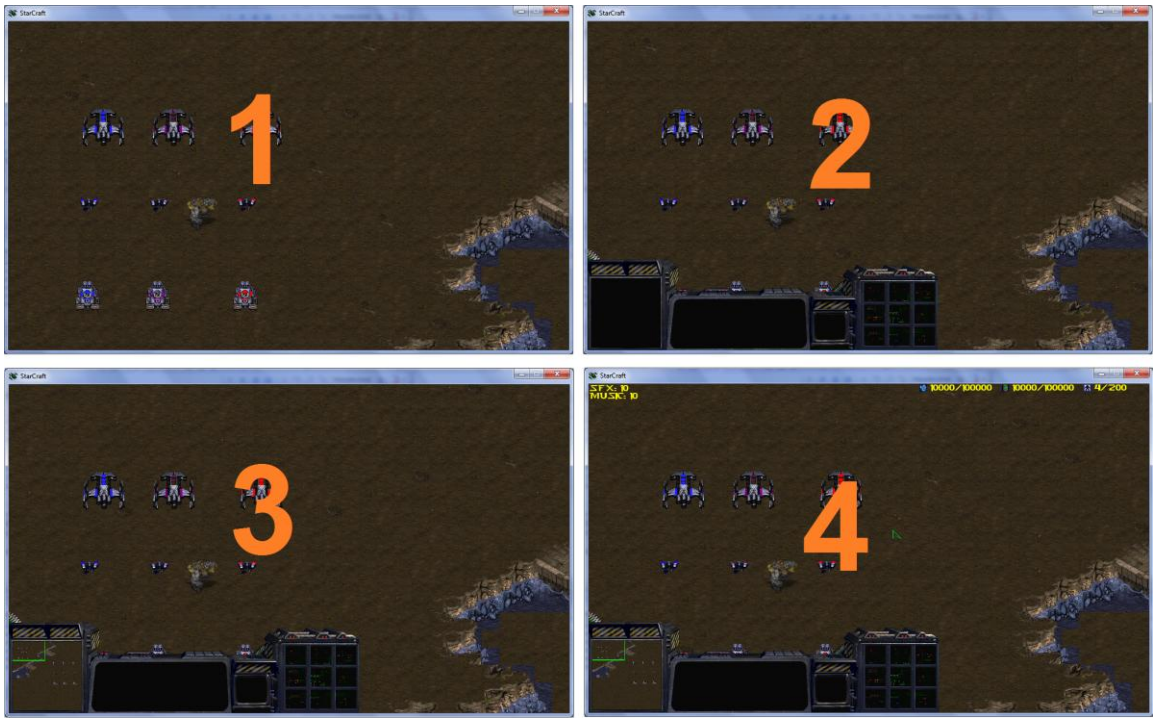

Figura 85. Proceso de dibujado

5.4. Mapa

Uno de los elementos base del proyecto es la existencia de un mapa. Este objeto, utilizando el símil del juego de mesa, es la abstracción del tablero de juego. El mapa estará compuesto por casillas y las unidades y edificios se dispondrán sobre él durante la partida.

La única diferencia con el símil propuesto, quizá, estriba en que las dimensiones del mapa escapan a las de la propia ventana de juego. Por tanto, en cualquier momento del juego se disfruta de un punto de vista concreto, pudiendo cambiarlo según las necesidades de cada momento.

Esta sección se dedica a comentar los aspectos de implementación más importantes acerca del mencionado tablero de juego (o mapa).

5.4.1. Definición

Un mapa es un conjunto de casillas. Habrá N casillas a lo ancho y M casillas a lo alto en un mapa de NxM dimensiones. Computacionalmente, cada casilla será una instancia de la clase Image. Visualmente, cada casilla será un cuadrado de A píxeles de ancho por B píxeles de alto. Además, una casilla podrá ser atravesada por una unidad de tierra (si es “transitable”) o no (si es “bloqueante”).

Ajustándose a la definición proporcionada se ha creado un conjunto de casillas posibles. A partir de ese conjunto se podrá definir el mapa a utilizar. A continuación se muestra el formato de fichero de definición de mapa, para luego definir cada aspecto individualmente.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="map">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="tileSetName" type="xs:string" />
        <xs:element name="tileSize">
          <xs:complexType>
            <xs:attribute name="x" type="xs:unsignedByte"
use="required" />
            <xs:attribute name="y" type="xs:unsignedByte"
use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element name="mapDimensions">
          <xs:complexType>
            <xs:attribute name="x" type="xs:unsignedByte"
use="required" />
            <xs:attribute name="y" type="xs:unsignedByte"
use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element name="randomizeUndefinedTilesRange">
          <xs:complexType>
            <xs:attribute name="x1" type="xs:unsignedByte"
use="required" />
            <xs:attribute name="y1" type="xs:unsignedByte"
use="required" />
            <xs:attribute name="x2" type="xs:unsignedByte"
use="required" />
            <xs:attribute name="y2" type="xs:unsignedByte"
use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element name="mapConfiguration">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="tile">
                <xs:complexType>
                  <xs:attribute name="positionWithinMapX"
type="xs:unsignedByte" use="required" />
                  <xs:attribute name="positionWithinMapY"
type="xs:unsignedByte" use="required" />
                  <xs:attribute name="tileImageX"
type="xs:unsignedByte" use="required" />
                  <xs:attribute name="tileImageY"
type="xs:unsignedByte" use="required" />
                  <xs:attribute name="pathConfig" type="xs:string"
use="required" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
```

Código 8: Formato de fichero de mapa (parte 1ª)

```
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Código 9: Formato de fichero de mapa (parte 2ª)

El formato de fichero expuesto define un conjunto de atributos:

- tileSetName: indica el nombre del recurso que contiene las imágenes que se utilizarán como base para crear el mapa. Típicamente, este nombre es el mismo que tiene el archivo de imagen sin extensión y en minúsculas.
- tileSize: tamaño de las casillas en píxeles. Se tomarán casillas de este tamaño desde el fichero indicado por tileSetName. Atributos: x (horizontal), y (vertical).
- mapDimensions: cantidad de casillas que debe tener el mapa en horizontal (X) y vertical (Y).
- randomizeUndefinedTilesRange: parámetro que indica las casillas que están especialmente diseñadas para actuar de relleno. Cuando no se especifique una imagen para una casilla, se tomará una según este parámetro y se configurará como "atravesable". Atributos: x1 (horizontal, inicio), y1 (vertical, inicio), x2 (horizontal, final), y2 (vertical, final).
- mapConfiguration: parámetro que agrupa la definición de las casillas. Puede contener entre 0 y N definiciones de tipo tile.
    - tile: parámetro que configura una casilla concreta del mapa, indicando qué imagen utilizar y si es o no transitable. Atributos: positionWithinMapX (horizontal), positionWithinMapY (vertical), tileImageX (horizontal), tileImageY (vertical), pathConfig (passable, blocking).

A continuación se ofrece un fichero de definición de mapa a modo de ejemplo:

```
<map>
  <tileSetName>tileset2</tileSetName>
  <tileSize x="32" y="32" />
  <mapDimensions x="70" y="70" />
  <randomizeUndefinedTilesRange x1="00" y1="00" x2="13" y2="25" />
  <mapConfiguration>
    <!--Random Tree-->
    <tile positionWithinMapX="10" positionWithinMapY="10"
tileImageX="16" tileImageY="04" pathConfig="blocking" />
    <tile positionWithinMapX="11" positionWithinMapY="10"
tileImageX="17" tileImageY="04" pathConfig="blocking" />
    <tile positionWithinMapX="10" positionWithinMapY="11"
tileImageX="16" tileImageY="05" pathConfig="blocking" />
    <tile positionWithinMapX="11" positionWithinMapY="11"
tileImageX="17" tileImageY="05" pathConfig="blocking" />
  </mapConfiguration>
</map>
```

Código 10: Ejemplo de fichero de mapa

5.4.2. Dibujado y desplazamiento

Esta sección se quiere dedicar a comentar el proceso de dibujado del mapa. Esta actividad se dividirá en dos partes: el dibujado del mapa en sí y los pasos necesarios para desplazar el punto de vista sobre el mapa.

En ambos casos, un código muy interesante es el del método Draw() de la clase MapScreenElement. En él se configura el punto de vista actual y se dibuja el mapa y sus elementos.

```
public override void Draw(GameTime gameTime)
  {
    //se calcula el offset que produce el viewport actual
    currentOffset.X = viewport.X;
    currentOffset.Y = viewport.Y;

    //Se dibuja el mosaico que compone el tablero de juego
    foreach (TileScreenElement tile in tiles)
    {
      tile.SetOffset(currentOffset);
      if (tile.IsInside(viewport))
      {
        tile.Draw(gameTime);
      }
    }

    //Se pintan las unidades terrestres
    foreach (UnitScreenElement unit in units)
    {
      if(unit.MovementType == UnitMovementType.Walking)
      {
        unit.SetOffset(currentOffset);
        unit.Draw(gameTime);
      }
    }

    //Se pintan los edificios
    foreach (UnitScreenElement unit in units)
    {
      if (unit.MovementType == UnitMovementType.None)
      {
        unit.SetOffset(currentOffset);
        unit.Draw(gameTime);
      }
    }

    //Se pintan las unidades voladoras
    foreach (UnitScreenElement unit in units)
    {
      if (unit.MovementType == UnitMovementType.Flying)
      {
        unit.SetOffset(currentOffset);
        unit.Draw(gameTime);
      }
    }
  }
```

Código 11: Dibujado del mapa en pantalla

En primer lugar se quiere comentar el proceso de dibujado. Como se aprecia en el Código 11, existe un bucle para cada grupo de elementos dibujables:

- Primero se disponen en pantalla las casillas que aparecen dentro del punto de vista actual. La comprobación de visibilidad evita hacer trabajar al subsistema gráfico, ya que existe la posibilidad real de sobrecargarlo.
- Después se dibujan las unidades terrestres. La comprobación de visibilidad ya no es necesaria, dado que la cantidad de unidades o edificios no se prevé demasiado alta.
- Más tarde, los edificios son mostrados en pantalla.
- Finalmente, las unidades aéreas reciben su orden de dibujado.

La disposición de los bucles es intencionada. Aprovechando que se pinta sobre la pantalla como si se hiciera sobre un lienzo, las unidades terrestres son pintadas antes que los edificios. Esto hace posible que dichas unidades "entren" en los edificios. Por otro lado, las unidades aéreas cubrirán las demás figuras, mostrando así que se encuentran en el aire.

El resto del Código 11 está dedicado casi en exclusiva al manejo del punto de vista sobre el mapa. Este concepto no es otra cosa que el desplazamiento realizado sobre las imágenes que se dibujan ("offset"). Justo antes de realizar el dibujado, cada elemento recibe la información de desplazamiento necesaria. Si este desplazamiento es pequeño, todavía se verán en pantalla; si es grande, desaparecerán por algún borde.

Los métodos que actúan sobre el desplazamiento son DoScrollUp, DoScrollDown, DoScrollRight y DoScrollLeft. A continuación se muestra este último.

```
private void DoScrollLeft()
  {
    if (viewport.X != 0)
    {
      viewport.X -= this.scrollingSpeed;
    }
    if (viewport.X < 0)
    {
      viewport.X = 0;
    }
  }
```

Código 12: Método DoScrollLeft

La actividad desarrollada por los métodos de desplazamiento del mapa es sencilla: desplazan el punto de vista una cantidad igual a scrollingSpeed. Si llegan al límite, entonces no desplazan más el punto de vista.

Estos métodos están unidos a las entradas de usuario mediante la colaboración de GameplayScreen. Por ejemplo, si la clase MouseScreenElement detecta que el jugador tiene el puntero del ratón muy cerca de un borde, entonces informará a GameplayScreen de que el usuario quiere cambiar el punto de vista sobre el mapa. Entonces, GameplayScreen transmitirá la información a MapScreenElement, que

activará los métodos de desplazamiento comentados, variando así lo que el jugador observa en pantalla. A continuación se ofrecen tres recortes de código para visualizar el proceso comentado.

```
public override void Update(GameTime gameTime)
{
  //Se considera el cursor apropiado según el scroll
  if (InputState.LastMouseState.X <
(ScreenManager.GraphicsDevice.Viewport.Width *
isScrollingPixelPercent) &&
      InputState.LastMouseState.Y <
(ScreenManager.GraphicsDevice.Viewport.Height *
isScrollingPixelPercent))
  {
    actual = cursorScrollUpLeft;
    scrollState = ScrollState.ScrollUpLeft;
  }

  // Se comprueba el resto de direcciones de desplazamiento

}
```

Código 13: Desplazamiento del mapa en MouseScreenElement

```
public override void Update(GameTime gameTime, bool
otherScreenHasFocus, bool coveredByOtherScreen)
{
  //Primero se llama al Update de la clase base
  base.Update(gameTime, otherScreenHasFocus, coveredByOtherScreen);

  //y luego, si es necesario, se actualizan los elementos de esta
instancia
  if (IsActive)
  {
    mouse.Update(gameTime);
    map.ScrollState = mouse.GetScrollState;
    map.Update(gameTime);

    // (...más actualizaciones...)
  }
}
```

Código 14: Desplazamiento del mapa en GameplayScreen

```
public override void Update(GameTime gameTime)
{
  if (ScrollState != ScrollState.NoScroll)
  {
    switch (ScrollState)
    {
      case ScrollState.ScrollUpLeft:
        DoScrollUp();
        DoScrollLeft();
        break;

        //Se comprueban otras direcciones...
    }
  //Se realizan más actualizaciones...
  }
}
```

Código 15: Desplazamiento del mapa en MapScreenElement

5.5. Unidades y edificios

Las unidades y los edificios son la parte viva del videojuego. Son los elementos de interacción con el usuario, a través de los cuales éste expresa su voluntad y obtiene el buscado entretenimiento.

En términos de diseño, las unidades y edificios son equivalentes. Cada uno es una instancia concreta de UnitScreenElement con sus datos específicos y debe ocuparse de su propia realidad visual.

En esta sección se comentan los aspectos más relevantes acerca de las unidades y edificios: qué imágenes se utilizan, cómo se han conseguido, por qué son como son, cómo se cargan, cómo se analizan las colisiones, cómo se mueven a lo largo y ancho del mapa y qué proceso es necesario para realizar las órdenes del usuario.

5.5.1. Imágenes

El aspecto visual de las unidades y de los edificios es un aspecto muy importante en las creaciones de ocio digital. Por este motivo, y aprovechando que se parte de la creación StarCraft original, se han tomado parte de las imágenes (o “sprites”) utilizadas en dicho videojuego.

Aparte de tener en cuenta la técnica de píxeles transparentes, ya comentada, se tuvo otra consideración en cuenta: las imágenes debían mostrar todos los ángulos posibles y ser aptas para realizar una secuencia animada. Esto supone una limitación, ya que algunas unidades tenían patas, brazos o alas, y eso complica enormemente el proceso de animación. Por tanto, se eligieron unidades que no tuvieran extremidades, pero que al mismo tiempo cumplieran los requisitos y fueran vistosas.

A continuación se ofrece una composición de las imágenes utilizadas para dibujar las unidades que finalmente han sido incluidas en el proyecto. De izquierda a derecha son BattleCruiser, SCV, Vulture y Tank. Nótese que se muestran también los colores de cada bando. Esta coloración fue realizada a posteriori en cada imagen individual.

En el caso de BattleCruiser, SCV y Vulture, las imágenes son interpretadas como imágenes de mosaico: únicamente se muestra una de ellas en un instante dado. En el caso de Tank ocurre algo similar, pero se dibuja primero la base común y luego la parte que lleva el color del bando.

Figura 86. Unidades y bandos

De la misma manera que se han acometido las unidades, a continuación se muestran los edificios implementados. En orden de lectura son Base, Resource Production Plant y Barracks. Los dos primeros se utilizan como imágenes de mosaico, mientras que el último es una imagen animada no cíclica. Es necesario destacar que esta diferencia en el tratamiento de edificios se ha transmitido de manera íntegra respecto de la implementación original de StarCraft.

Figura 87. Edificios y bandos

Para concluir, se quiere mostrar el código necesario para cargar estos recursos en memoria, pudiendo así más tarde representarlos en pantalla.

```
public override void LoadContent()
{
  //base tank texture
  if (unitType == UnitType.Tank)
  {
    Texture2D unitBaseTexture =
Content.Load<Texture2D>(unitBaseTextureName);
    unitBase = new Image(ScreenManager.SpriteBatch, unitBaseTexture,
          new Vector2(TileWithinMap.X * 32, TileWithinMap.Y * 32),
Color.White, new Vector2(0, 0), new
Vector2(unitTextureHeightAndWidth.X, unitTextureHeightAndWidth.Y));
  }
```

Código 16: Carga de imágenes de unidades (parte 1ª)

```
  //unit texture
  Texture2D unitTexture = Content.Load<Texture2D>(unitTextureName);
  unit = new Image(ScreenManager.SpriteBatch, unitTexture,
                   new Vector2(TileWithinMap.X * 32,
TileWithinMap.Y*32), Color.White, new Vector2(0, 0), new
Vector2(unitTextureHeightAndWidth.X, unitTextureHeightAndWidth.Y));

  //resto de carga de recursos...

}
```

Código 17: Carga de imágenes de unidades (parte 2ª)

En primer lugar, como se observa en el Código 16, se carga la textura que forma la base. Esto únicamente se realiza para las unidades tipo Tank, que están compuestas por una base sin color y una parte superior coloreada. Todas las unidades utilizan el Código 17, que inicializa la imagen que se dibujará en pantalla cuando sea necesario. Los dos extractos utilizan variables genéricas (unitTextureName, unitTexture, unitTextureHeightAndWeight) para ser capaces de cargar cualquier archivo de imagen de unidades o edificios.

5.5.2. Detección de colisiones

La detección de colisiones es uno de los aspectos más importantes de cualquier videojuego. Es el proceso necesario para comprobar si hay diferentes elementos visuales a punto de superponerse. Esta actividad es obligatoria, ya que la composición de imágenes en pantalla debe estar en todo momento bajo control.

En este proyecto, el proceso de detección se realiza en dos fases. La primera fase (“Fase 1”) consiste en discernir qué figuras deben ser evaluadas y cuáles no, evitando cálculos innecesarios. La segunda fase (“Fase 2”) está destinada a averiguar el resultado de la colisión entre dos figuras.

El hecho de dividir la detección de colisiones en dos etapas se debe a la ralentización producida por la comprobación repetitiva de multitud de elementos que pueden colisionar. Por ejemplo, un mapa cuadrado de 50 casillas cuadradas con una unidad presente puede dar lugar, por ejemplo, a no menos de 50*50*32*32*15*15 = 576000000 comprobaciones (una por cada píxel del mapa y por cada píxel de la unidad). Por supuesto, este es el caso extremo, pero sirve para ilustrar el problema.

Por tanto, la Fase 1 consiste en aplicar un método lógico. La posición de cada elemento del mapa (casillas, unidades y edificios) es conocida, ya que cada objeto se encuentra almacenado en una lista junto a sus homólogos. Para evitar cálculos innecesarios, se consultarán solamente los datos de aquellos elementos que tengan relación con la actividad que se desea llevar a cabo. De esta manera se simplifica el proceso enormemente, ahorrando el valiosísimo y limitado tiempo de computación. Un ejemplo de este caso es el movimiento por el mapa: si la unidad que desea

moverse es aérea, entonces solamente se comprobarán las colisiones con otras unidades aéreas cercanas. Si la unidad fuera terrestre, el número de comprobaciones sería mayor: casillas del mapa, edificios, unidades de tierra…

Para implementar la Fase 2 se han considerado varios métodos:

- Detección mediante celdas: cada elemento se encuentra situado en una celda. Dos elementos colisionarán si pretenden ocupar la misma celda en el mismo instante de tiempo. Este proceso de detección es muy sencillo, computacionalmente muy ligero y se adapta de manera natural a los conceptos desarrollados.
- Detección mediante figuras geométricas: cada elemento se engloba en una figura geométrica sencilla. El proceso de detección consiste en utilizar funciones matemáticas de conjuntos (inclusión, pertenencia) para ver si existe solapamiento entre dos objetos. Es un método sencillo y computacionalmente ligero, pero tiene la desventaja de ser menos eficiente cuanto más preciso (cuanto más detallada sea la figura geométrica, más comprobaciones es necesario hacer). Además, la generación de figuras geométricas adecuadas es un proceso no trivial.
- Detección mediante píxeles: cada elemento tiene una realidad visual compuesta por píxeles. La detección consiste en evaluar si dos píxeles no transparentes de dos elementos diferentes tienen la misma posición en pantalla. Este método es más complejo que el anterior, y la computación requerida es la mayor de los tres métodos reseñados.

El método finalmente escogido para implementar la Fase 2 es el de detección mediante celdas. Los motivos que justifican esta decisión son dos: en primer lugar, este método se adapta de manera natural y sencilla a los conceptos modelados; en segundo lugar, pero no por ello menos importante, es el método escogido para el desarrollo StarCraft original.

Este método se basa en el concepto de que cada elemento del juego estará situado en una casilla del mapa. Además, algunos elementos podrán ocupar varias casillas según sean sus dimensiones particulares. Las situaciones de colisión serán aquellas que den lugar a la superposición parcial o total de dos elementos que no deben superponerse. A continuación se muestra un ejemplo gráfico de este algoritmo de colisiones.

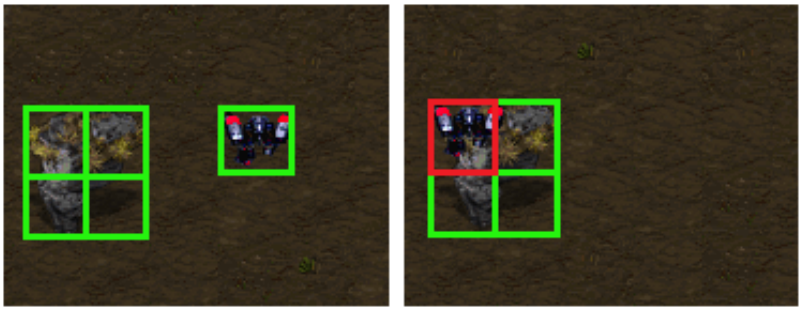

Figura 88. Detección de colisiones

En la parte izquierda de la Figura 88 se aprecia cómo la unidad y el obstáculo del mapa no comparten ninguna casilla, por lo que no se puede afirmar que exista colisión entre ambos elementos. En la parte derecha, sin embargo, existe una casilla coincidente, marcada en color rojo. Esto ilustra una detección de colisiones positiva, por lo que el sistema evitará esta situación impidiendo a la unidad desplazarse a esa casilla.

Una vez ilustrados los dos métodos implementados, el pseudocógido del proceso que realiza la aplicación es el siguiente:

```
Para todas las unidades y edificios
  Si están llevando a cabo una orden
    Comprobar colisión con elementos cercanos
      Si es una unidad aérea
        Fase 2 (resto de unidades aéreas)
      Si es una unidad terrestre
        Fase 2 (unidades terrestres, edificios, mapa)
      Si es un edificio
        Fase 2 (unidades terrestres, mapa)

Finalmente
  Si la detección ha sido positiva
    Si la orden no puede detenerse o si debe ser llevada a cabo
      Aplicar Fase 2 (edificios, mapa) //Fase 2 "relajada"
```

Código 18: Pseudocódigo de detección de colisiones

El Código 18 muestra el proceso aplicado por el sistema cuando se quiere ejecutar o se está ejecutando alguna orden sobre las unidades o edificios. El algoritmo descrito como Fase 1 se utiliza de manera implícita, mientras que la Fase 2 se hace explícita y utiliza únicamente ciertos grupos de datos.

Para concluir, debe destacarse la "Fase 2 relajada". En el videojuego StarCraft original se observó que varios elementos podían superponerse en una posición de manera legal. Esto se vio beneficioso, por ejemplo, para las siguientes situaciones:

- Una unidad recibe la orden de parada, por lo que se detiene. Si esta parada se produce en una situación de colisión, la Fase 2 impediría a la unidad volver a moverse.

- Una gran cantidad de unidades reciben la orden de atacar un objetivo. Si únicamente se aplicara la Fase 2, sólo un pequeño conjunto de unidades podría atacar el objetivo designado, mientras que las demás esperarían a tener espacio suficiente.
- Un edificio está produciendo una unidad. Sin embargo, justo antes de terminar el proceso, todas las casillas a su alrededor se llenan con otras unidades. Entonces, la Fase 2 impediría a la nueva unidad situarse en el mapa.

Estas y otras situaciones problemáticas han sido solucionadas mediante la relajación del algoritmo "Fase 2". Esta modificación consiste en ignorar ciertas colisiones. De esta manera se permite, temporalmente, que las actividades transcurran con normalidad, aumentando así el disfrute del usuario.

5.5.3. Movimiento y orientación

Una vez que se han ilustrado convenientemente las unidades se hace posible la descripción de los algoritmos de movimiento y orientación. Interpretados en su forma más básica, estos dos sistemas son los encargados de escoger la posición en el mapa correcta y la imagen adecuada de una unidad en cada momento. De esta manera se ofrece una fluidez de movimiento visual de cara al jugador, lo cual enriquece la experiencia de juego.

En primer lugar se comentará el algoritmo de movimiento de la unidad. Analizando el camino de la unidad en el más mínimo detalle, se tiene la necesidad de desplazar la unidad píxel a píxel de una casilla a otra.

Para que este movimiento sea fluido, los desplazamientos no deben ser superiores a un píxel: en caso contrario, la imagen se percibirá cortada, avanzando grandes espacios en un intervalo de tiempo muy breve. El código que realiza esta actividad de manera controlada es el siguiente:

```
//Se calcula el número de píxeles que se ha de avanzar
TimeSpan currentTime = new TimeSpan(DateTime.Now.Ticks);
int pixelsToMove = (int)(movementSpeed *
timeSinceLastMovement.TotalSeconds);

//Se avanzan los píxeles necesarios. Si son 0 píxeles, simplemente se
actualiza el contador de tiempo
if (pixelsToMove > 0)
{
  timeSinceLastMovement = new TimeSpan();

  //Se actualiza la posición dentro de la casilla mediante el valor
  //pixelsToMove. Si es necesario, se cambia de casilla
```

Código 19: Cálculo de movimiento en base al tiempo

Siguiendo lo mostrado en el Código 19, la variable pixelsToMove es la cantidad de píxeles que se avanza en la llamada actual al método. Esta variable toma normalmente el valor cero, ocasionalmente el uno y, muy rara vez, dos.

También es destacable la utilización del parámetro movementSpeed. Este campo, particular a cada unidad, almacena el factor por el que se multiplicará el intervalo de tiempo transcurrido (timeSinceLastMovement) para averiguar cuántos píxeles se ha de mover la unidad. Por supuesto, movementSpeed está directamente relacionado con la velocidad que muestra la unidad al ser utilizada.

Respecto al algoritmo de orientación de las unidades se destacan varios aspectos. En primer lugar, cada una de las imágenes mostradas ilustra a la unidad con su parte delantera hacia un ángulo concreto. Estos recursos son muy limitados y costosos, por lo que será necesario utilizar las imágenes existentes para simular la existencia de aquellas orientaciones que físicamente no existan.

Por tanto, se ha trabajado en todo momento con un total de 32 orientaciones posibles a partir de 17 orientaciones existentes físicamente. El algoritmo de orientación define 32 arcos de circunferencia, cada uno abarcando el mismo espacio angular que los demás. El código mostrado a continuación ilustra la creación de los ángulos.

```
private void UpdateHeading(Vector2 previousPosition, Vector2
nextPosition)
{
  //Se preparan los datos necesarios para calcular la orientación de
la unidad
  Vector2 v = new Vector2(nextPosition.X - previousPosition.X,
nextPosition.Y - previousPosition.Y);

  //Se calcula el ángulo y se normaliza: cero grados = hacia arriba,
crecimiento horario
  float angle = MathHelper.ToDegrees((float)Math.Atan2(v.Y, v.X));
  angle += 90;
  if (angle < 0) angle += 360;

  //Se ajusta la nueva imagen de la unidad en base a la orientación
  float sector = 360 / 32; //grados totales / nº de imágenes
existentes en el fichero
  Vector2 finalHeading = new Vector2(0, (int)Math.Floor(angle /
sector));
```

Código 20: Cálculo de arcos de circunferencia para orientación

La selección del ángulo actual es un proceso sencillo: se toma la posición previa de la unidad y la posición actual, y se determina el ángulo de avance. Se toman únicamente dos posiciones porque, por un lado, un dato no daría el conocimiento suficiente y, por otro lado, si se toman más datos puede ocurrir que los giros de las unidades sean excesivamente lentos.

Para cada uno de los arcos de circunferencia, el método de orientación asigna una imagen concreta y unos efectos específicos. El código que realiza esta actividad es similar al que se muestra a continuación.

```
//Se considera si es necesario girar la unidad
if (finalHeading.Y == currentHeading.Y)
{
  return;
}
else
{
  //Ya que es necesario girar la unidad, se considera qué camino es
más corto para hacerlo
  int currentQuadrant = 0, finalQuadrant = 0;
  switch ((int)currentHeading.Y)
  {
    case 0: case 1: case 2: case 3: case 4: case 5: case 6: case 7:
      currentQuadrant = 1;
      break;

    //...
  }

  switch ((int)finalHeading.Y)
  {
    case 0: case 1:case 2: case 3: case 4: case 5: case 6: case 7:
      finalQuadrant = 1;
      break;

    //...
  }

  if (currentQuadrant == 1)
  {
    if (finalQuadrant == 1)
    {
      //ambas posiciones están en el mismo cuadrante
      //no es necesario modificar la orientación
    }
    else if (finalQuadrant == 2)
    {
      currentHeading.Y++;
    }
    else if (finalQuadrant == 3)
    {
      currentHeading.Y++;

    //...

  //Se calcula y asigna la imagen que tendrá la unidad en pantalla
  Vector2 aux = new Vector2(0, 0);
  if (currentHeading.Y >= 0 && currentHeading.Y <= 16)
  {
    unitBase.ImageEffects = SpriteEffects.None;
    unit.ImageEffects = SpriteEffects.None;
    unitBase.TilePosition = currentHeading;
    unit.TilePosition = currentHeading;
  }
```

Código 21: Cambio de orientación (parte 1ª)

```
else if (currentHeading.Y == 17)
{

  aux.Y = 15;
  unitBase.ImageEffects = SpriteEffects.FlipHorizontally;
  unit.ImageEffects = SpriteEffects.FlipHorizontally;
  unitBase.TilePosition = aux;
  unit.TilePosition = aux;
}

//...
```

Código 22: Cambio de orientación (parte 2ª)

El punto crítico del algoritmo de orientación son los giros de las unidades. Es por ello que la primera parte del Código 21 se dedica a considerar los cuadrantes que está visitando la unidad. En base a este dato se intentará ofrecer una continuidad visual satisfactoria variando la imagen utilizada poco a poco. Precisamente este es el cometido de las operaciones realizadas con currentHeading: avanzar o retroceder una imagen cada iteración, de manera que el cambio apreciado sea mínimo.

La segunda parte del código mostrado consigue seleccionar la imagen necesaria en cada momento. Los primeros ángulos considerados (de 0 a 16, ambos inclusive) se encuentran disponibles en el propio archivo de datos, por lo que simplemente se selecciona la porción de imagen a utilizar. Para el caso 17 y siguientes, cuya imagen físicamente no existe, es necesario tomar una imagen existente y aplicarle un efecto de espejo: en vez de dibujar los datos de izquierda a derecha se hará al revés, de derecha a izquierda. Las unidades han sido especialmente seleccionadas debido a que soportan gráficamente esta operación: todas tienen un alto grado de simetría.

5.5.4. Búsqueda de caminos

En apartados anteriores se expuso el mapa de juego: definición, composición… También se apuntaron los fundamentos de unidades y edificios: imágenes, movimiento, orientación…

En esta sección se aprovechan los conceptos explorados previamente para comentar en detalle el algoritmo de búsqueda de caminos implementado. Como se podrá observar, todo el conjunto forma un único elemento cohesivo: se tiene el mapa definido en base a sus casillas componentes; cada unidad se encuentra en una o varias casillas en un instante dado; el movimiento se realiza de casilla en casilla.

El algoritmo de búsqueda utilizado es A* (“A estrella”). Dicho sistema tiene su origen a finales de los años 60. Su principal característica es que es capaz de encontrar el camino de menor coste (distancia, en este caso) entre un nodo inicial y uno final siempre que dicha ruta exista.

Aunque el algoritmo es bastante sencillo una vez que se consigue implementar, comprender los cálculos intermedios puede ser un proceso bastante complicado. Esta sección del documento se dedica a ilustrar el comportamiento del algoritmo implementado y su relación con los conceptos modelados en la aplicación.

Introducción

Para poder ilustrar adecuadamente la explicación se propone la siguiente situación inicial (ver Figura 89). Se tiene una casilla inicial coloreada en rojo, una casilla destino coloreada en verde, varias casillas que impiden el paso coloreadas en azul y varias casillas transitables del mapa coloreadas en negro. La casilla inicial podría ser una unidad, la casilla destino podría ser un punto cualquiera del mapa, las casillas azules podrían ser un lago y las casillas negras podrían ser las casillas transitables del mapa.

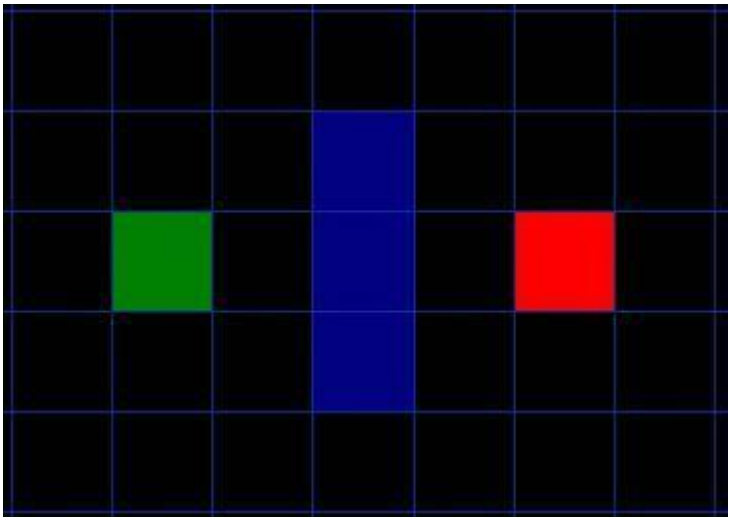

Figura 89. Situación inicial A*

El primer hecho notable que se debe destacar es la división realizada en base a casillas cuadradas. Aunque el mapa consta de un conjunto enorme de píxeles, simplificar el área de búsqueda aumenta la velocidad de búsqueda de caminos. La posibilidad de sobrecargar el sistema mediante la búsqueda de caminos píxel a píxel fue comprobada empíricamente, por lo que se realizó la simplificación comentada. La diferencia de resultados entre el método complejo y el simplificado es inapreciable.

La simplificación realizada reduce las posibilidades área de búsqueda a la cantidad de casillas existentes en la matriz del mapa. Cada elemento de la matriz representa una de las casillas del mapa, y podrá ser atravesada por unidades terrestres o no.

El camino es la lista de casillas que debe ocupar la unidad para ir desde la casilla inicial hasta la casilla destino. Una vez que se haya encontrado el camino, la unidad avanzará píxel a píxel hasta recorrer todas la casillas del camino: unas veces hacia

arriba, otras hacia abajo, otras en diagonal… Así hasta encontrarse en la casilla destino. A continuación se adjunta el código de comprobación del avance entre casillas.

```
//Se comprueba si se ha llegado al destino
if (step.X == newTileWithinMap.X && step.Y == newTileWithinMap.Y)
{
  if (isCollectingResources && path.Count > 0)
  {
    path.RemoveAt(0);
  }
  else if(isCollectingResources && cyclicPath.Count > 0)
  {
    cyclicPath.Add(cyclicPath.ElementAt(0));
    cyclicPath.RemoveAt(0);
  } else
  {
    path.RemoveAt(0);
  }
}
```

Código 23: Actualización del camino a seguir

El Código 23 muestra la actividad realizada cuando la unidad atraviesa una nueva casilla. Primero se comprueba si la unidad ha llegado a la siguiente casilla del camino. En base al resultado y a la actividad que esté realizando la unidad se puede, bien eliminar la casilla de la lista (camino normal), o bien añadir la casilla a la última posición del camino (camino cíclico).

En adelante, las casillas del mapa podrán ser referidas con el término genérico "nodo", más adecuado a la terminología común de la búsqueda de caminos.

Iniciando la búsqueda

Una vez descrito el aspecto conceptual del mapa y su consideración por parte del algoritmo, el proceso es sencillo. Se ha de tomar el nodo inicial y explorar las direcciones en las que es posible navegar. Este proceso se repetirá hasta que se encuentre el nodo destino o hasta que se pruebe que no existe ningún camino.

Los pasos iniciales de la búsqueda se pueden describir mediante la siguiente secuencia:

- Se añade el nodo actual a la lista de nodos abiertos. Esta lista contiene las casillas que podrían formar parte del camino final, por lo que el contenido de la misma son los nodos que necesitan ser comprobados.
- Para cada nodo navegable desde el actual:
  - Guardar el nodo actual como padre del nuevo nodo.
  - Introducir el nuevo nodo en la lista abierta.
- Ya que el nodo actual ha sido explorado, se cambia de la lista de nodos abiertos a la lista de nodos cerrados.

El aspecto conceptual del primer paso se ilustra en la siguiente figura:

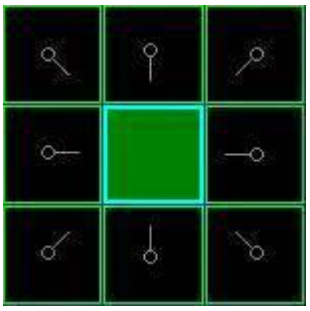

Figura 90. Primera exploración de nodos

Eligiendo el siguiente nodo

Tras concluir la exploración de nodos posibles, el siguiente paso en la búsqueda es repetir el proceso para los nodos de la lista abierta. Antes de comenzar una nueva iteración es necesario evaluar cada nodo abierto mediante una ecuación similar a F = G + H, donde F es el valor asociado al nodo abierto, G es el coste de moverse desde el nodo padre al nodo abierto y H es el coste estimado (heurístico) para ir desde el nodo abierto hasta el nodo final.

Una vez aclarado este aspecto, el camino final se construye mediante la repetición iterativa de los pasos descritos. Por tanto, los valores G y H de cada nodo tienen una importancia crítica a la hora de calcular un camino bueno y rápido.

En el presente proyecto, el valor G de cada celda es siempre uno. Esto significa que cualquier unidad puede avanzar en cualquier dirección con un coste similar. Sin embargo, las siguientes ideas fueron barajadas:

- Se puede subir el valor de G para las casillas “cuesta arriba” del mapa y bajarlo para las casillas “cuesta abajo”. De esta manera se primarán algunos caminos: en ocasiones llanos, en ocasiones más trabajosos.
- Se puede aumentar el valor de G de las casillas que queden por detrás y a los lados de la unidad. Esto dará lugar a cierto efecto de inercia: las unidades preferirán las líneas rectas y los giros lentos a los caminos que impliquen giros bruscos.

El valor de G de cada celda será, por tanto, el valor de G del nodo padre más el valor de G del nodo hijo. En relación al factor H también existe un amplio abanico de posibilidades de cálculo: en el presente proyecto se utiliza la distancia en línea recta entre la casilla actual y la casilla destino. Se ha escogido este valor por ser la función más simple capaz de producir caminos adecuados.

En esta sección se asignarán valores arbitrarios a G y H para ilustrar el cálculo de búsqueda de caminos con mayor riqueza en cuanto a la variedad que puede presentar. A continuación se presenta la primera exploración de nodos con sus respectivos valores de F, G y H calculados.

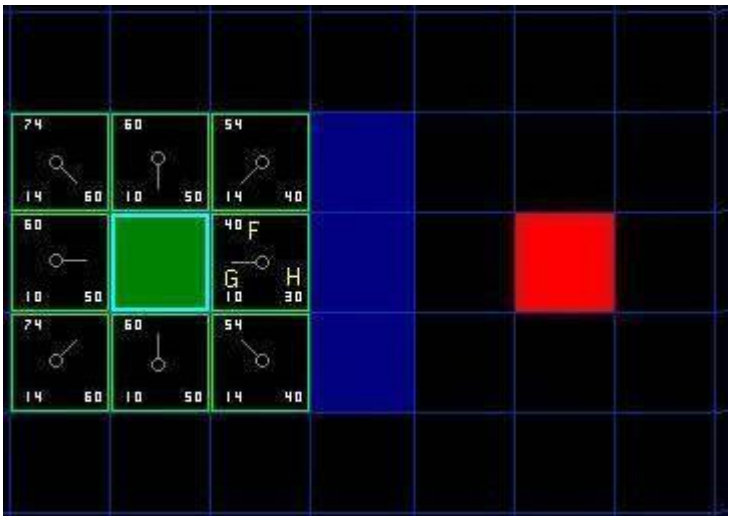

Figura 91. Distancias estimadas

Como se puede observar en la Figura 91, el cuadro marcado con las letras F, G y H presenta la menor estimación. Por tanto, este nodo será el primero en ser explorado.

Continuando la búsqueda

En adelante, el proceso de búsqueda seguirá los siguientes pasos:

- Escoger el nodo con el valor de F más bajo.
- Extraer este nodo de la lista abierta e introducirlo en la lista cerrada.
- Explorar todos los nodos adyacentes al nodo seleccionado excepto aquellos que estén en la lista abierta, en la lista cerrada o que no sean atravesables por la unidad.
    - Configurar cada nodo que coincida con el criterio indicando que el nodo actual es el nodo padre.
    - Añadir el nodo a la lista abierta.
- Para todos los nodos que se haya descubierto que ya están en la lista abierta, comprobar si el camino a ese nodo es mejor que el actual. Es decir, si el valor de G del nodo de la lista abierta es menor que el valor de G del nodo actual.
    - Si el valor de G es mayor o igual, continuar.
    - Si el valor de G es menor, se debe almacenar el nodo actual en la lista cerrada.
- Proseguir con el nodo de la lista abierta que tenga el valor de F más bajo.

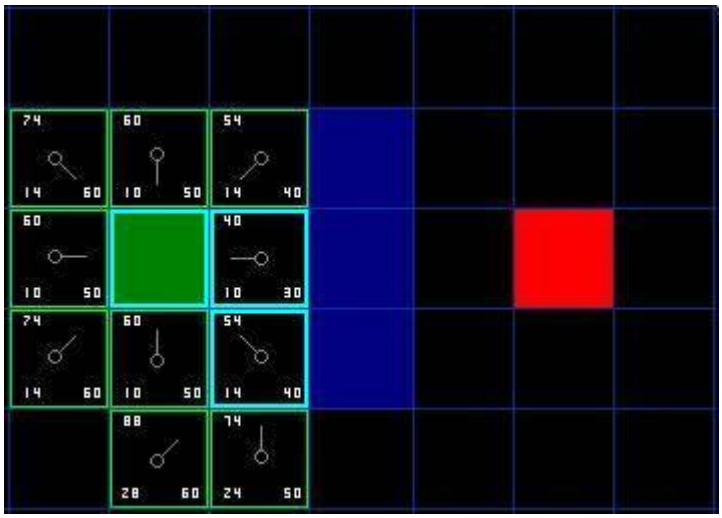

Figura 92. A después de explorar dos nodos*

La Figura 92 muestra el resultado de haber explorado dos nodos. En primer lugar se exploró el nodo que está a la derecha del nodo inicial (verde). A partir de este nodo no se añadió ninguno a la lista abierta: unos ya estaban en dicha lista y otros (azul, verde) no son transitables. Por tanto, el algoritmo explora el siguiente nodo de la lista abierta con el valor F más bajo: el que está abajo y a la derecha del nodo verde.

Concluyendo el proceso

Tras sucesivas iteraciones del método, los nodos adyacentes al inicial han sido explorados. Esto ha ocurrido debido a que su valor F era el más bajo de la lista abierta. Sin embargo, esta tendencia finaliza cuando alguno de los nodos explorados se acerca a los bordes de los nodos marcados en azul. Estos nodos están muy próximos al nodo final (rojo), por lo que su valor de F es cada vez menor.

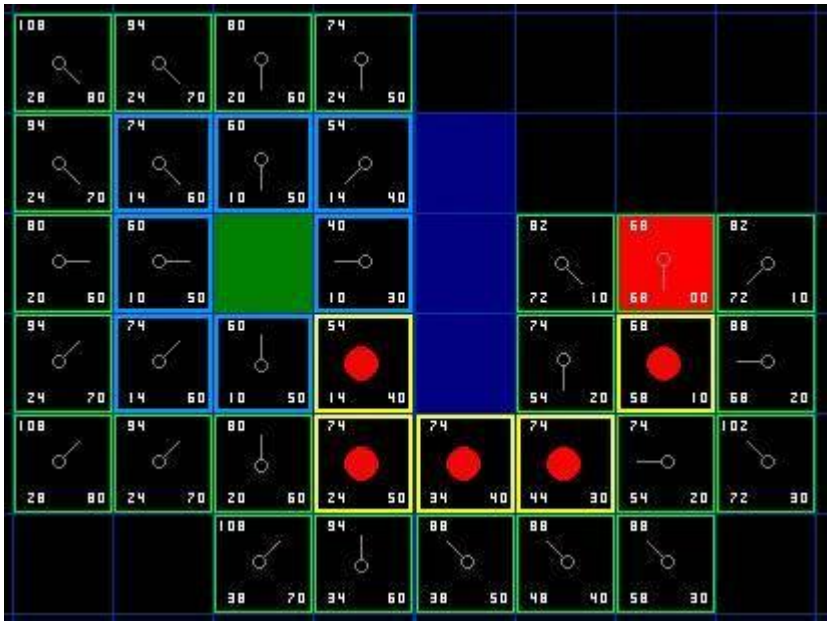

Figura 93. Camino encontrado

Tras iterar suficientemente, se tiene que uno de los nodos explorados es el nodo final (rojo). En este caso, el algoritmo detecta automáticamente que ha encontrado el mejor camino.

Para poder utilizar el camino encontrado se debe recorrer el árbol jerárquico hacia arriba: primero el nodo rojo, luego su padre, luego su padre… y así hasta llegar al nodo verde. El Código 24 muestra la utilización del camino una vez ordenado desde la casilla de inicio hasta la casilla final.

```
List<Vector2> pathAux = new List<Vector2>(200);
pathAux = CalculateSimplePathBetween(new Vector2(end.X, end.Y),
start, end);
if (pathAux != null && (path.Count == 0 || pathAux.Count <
path.Count))
{
  path = pathAux;
}
if (pathAux == null)
{
  path = new List<Vector2>();
  Log.Write("NO PATH!! (end " + end.X + " " + end.Y + ")");
}
```

Código 24: Utilización del camino encontrado

Generalización para grupos de unidades

En los apartados anteriores se ha descrito el algoritmo utilizado para el cálculo del camino de una unidad. En este apartado se analiza el trabajo realizado para que varias unidades se desplacen a un punto del mapa.

La idea planteada es simple: se tiene la necesidad de mover varias unidades a un punto concreto del mapa. En la creación StarCraft original, esta necesidad se resuelve de la siguiente manera: se seleccionan N unidades y se solicita que se desplacen a un punto del mapa. Entonces, la unidad más cercana calcula el camino a seguir, y el resto de unidades calculan su camino respecto a esta primera unidad. De esta manera se simplifican los cálculos, evitando sobrecargar el sistema con demasiadas operaciones.

En el desarrollo actual se ha implementado el concepto original. Para impartir órdenes a varias unidades, lo primero que se debe realizar es la selección de las mismas como grupo. Una vez realizada esta operación se podrán impartir las órdenes que tienen en común: mover o atacar.

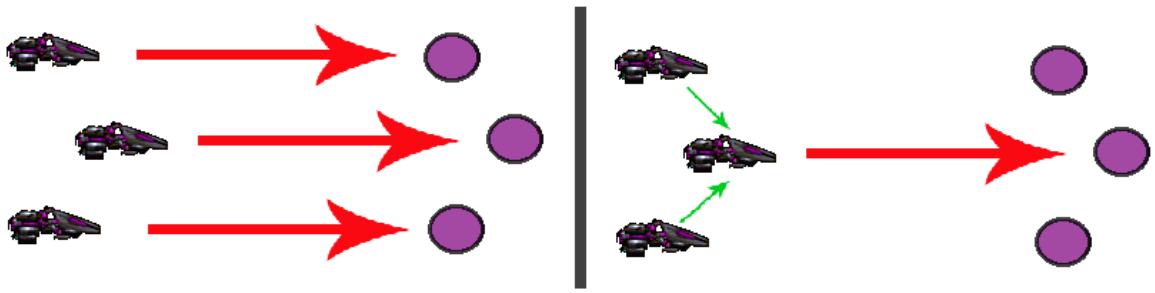

Figura 94. Diagrama de actuación en grupo

La Figura 94 muestra las dos posibles situaciones al actuar como un grupo. En la parte izquierda se calculan tres caminos largos, representados por flechas rojas. En la parte derecha, sin embargo, se ahorra tiempo de cálculo aprovechando el camino largo (flecha roja) y calculando un camino muy corto (flecha verde). En el desarrollo de este proyecto se ha implementado el diagrama de la parte derecha.

Si la orden impartida al grupo de unidades es mover, la unidad más cercana calculará el camino que ha de seguir para llegar a su objetivo. El resto de unidades del grupo aprovecharán este cálculo: su ruta será el camino hallado por la unidad más cercana más el camino necesario para llegar a la unidad más cercana. Al concluir la operación, las unidades habrán llegado a su destino, siendo necesaria únicamente una búsqueda de caminos.

En el caso de la orden de ataque, el comportamiento será similar: la unidad más cercana calculará su camino, y las demás unidades aprovecharán este cálculo para añadirlo a su ruta. Al concluir el viaje, las unidades comenzarán a atacar al objetivo común.

La funcionalidad descrita se mantiene fiel al espíritu de la creación StarCraft original: trata de evitar todo el cálculo que sea posible. El comportamiento tiene una gran virtud: permite que el jugador distribuya sus órdenes en un corto espacio de tiempo, aumentando así el tiempo disponible para otras actividades (planear la estrategia a mayor escala, construir unidades…).

5.5.5. Órdenes

El último aspecto que se analizará en esta fase de implementación son las órdenes que pueden ser dadas tanto a unidades como a edificios. Estos comandos son los que trasladan la voluntad del jugador al terreno de juego, por lo que forman una parte muy importante del juego.

Las órdenes implementadas son mover, atacar, parar, seguir, construir edificio, recolectar y producir unidad. En este apartado se explorará en mayor profundidad la orden de ataque, ya que todas comparten el mismo mecanismo de activación.

En primer lugar, la comprobación sobre las órdenes se lleva a cabo en ControlBarScreenElement. A continuación se muestra parte del código de HandleInput.

```
if (buttonRectangles[0, 0].Contains(inputState.CurrentMouseState.X,
inputState.CurrentMouseState.Y) &&
inputState.LastMouseState.LeftButton == ButtonState.Pressed &&
inputState.CurrentMouseState.LeftButton == ButtonState.Released)
{
  clickedButton.X = 0;
```

Código 25: Método HandleInput de ControlBarScreenElement (parte 1ª)

```
  clickedButton.Y = 0;
  //Log.Write("clicked button 0 0");
}
else if
(buttonRectangles[0,1].Contains(inputState.CurrentMouseState.X,
inputState.CurrentMouseState.Y) &&
inputState.LastMouseState.LeftButton == ButtonState.Pressed &&
inputState.CurrentMouseState.LeftButton == ButtonState.Released)
{
  clickedButton.X = 0;
  clickedButton.Y = 1;
  //Log.Write("clicked button 0 1");
}

//...
```

Código 26: Método HandleInput de ControlBarScreenElement (parte 2ª)

Los Códigos 25 y 26 recogen la sección de ControlBarScreenElement que comprueba si el usuario ha hecho clic con el botón izquierdo del ratón sobre alguno de los botones de la barra. De ser así, la variable clickedButton almacenaría el valor apropiado, que sería transmitido por GameplayScreen.

```
//...

if (controlBar.HandleInput(input))
{
  //Dado que el clic ha sido gestionado por controlBar, se comprueba
si se ha pulsado un botón
  if (controlBar.ClickedButton.X != -1)
  {
    map.SetClickedButton(controlBar.ClickedButton);
  }
  return;
}
//...
```

Código 27: Pulsación de botones en GameplayScreen

El Código 27 muestra la comunicación de qué botón ha sido pulsado a MapScreenElement en el método HandleInput de GameplayScreen. El conocimiento se transmitirá hasta la unidad seleccionada como se muestra a continuación.

```
internal void SetClickedButton(Vector2 clickedButton)
{
  foreach (UnitScreenElement unit in unitsList)
  {
    if (unit.IsSelected)
    {
      unit.SetClickedButton(clickedButton);
    }
  }
}
```

Código 28: Pulsación de botones en MapScreenElement

El Código 28, finalmente, transmite la orden directamente a la unidad seleccionada. Dicha unidad comprobará si es posible realizar la orden y, en caso afirmativo, comenzará a trabajar en ello. A continuación se muestra el comienzo de la orden "atacar" en el método HandleInput de la clase UnitScreenElement.

```
if (targetedUnit != null && reactionType !=
targetedUnit.reactionType)
{//Si hay una unidad targeteada y es de otro bando, se comienza el
ataque sobre la unidad targeteada
  //isAttacking = true;
  if(DoStopMoving()){
    Log.Write("stopping everything: battle mode on!!");
    isAttacking = true;
    unitBeingAttacked = targetedUnit;
  }
```

Código 29: Comienzo de la orden "atacar"

A partir de este momento, la unidad actualizará el estado de la orden cada vez que se llame a su método Update(). Un extracto de dicho método se adjunta a continuación.

```
public override void Update(GameTime gameTime)
{
  //Se actualiza el estado de la producción si la unidad está
entrenando a una nueva unidad.
  if (isBeingProduced != UnitType.None)
  {
    UpdateProduction(gameTime);
  }

  //Se actualiza el estado de la construcción si la unidad está
construyendo un edificio nuevo.
  if (isBuilding)
  {
    UpdateBuildingProcess(gameTime);
  }

  //Se actualiza el estado del ataque si la unidad está atacando a
otra.
  if (isAttacking)
  {
    UpdateAttackProcess(gameTime);
  }

  //...
}
```

Código 30: Extracto del método Update de la clase UnitScreenElement

El Código 30 muestra las operaciones periódicas de actualización de cualquier unidad y edificio del sistema. A continuación se expone el pseudocódigo de la operación de ataque, ya que el método UpdateAttackProcess es demasiado complejo y voluminoso.

```
Si todavía existe el objetivo y no está en rango
  Entonces
    Si mi objetivo o yo mismo hemos cambiado de casilla desde la
última llamada a UpdateAttackProcess
      Entonces calcular el camino necesario para ir hasta el objetivo
      Configurar el camino a seguir
      Terminar la llamada

Si todavía existe el objetivo y está en rango
  Si no se ha cumplido el tiempo de recarga
    Entonces aumentar el contador de tiempo
  Si se ha cumplido el tiempo de recarga
    Entonces realizar un disparo contra el objetivo (restar puntos de
vida, reproducir un sonido, visualizar el disparo realizado)
    Si el objetivo ha muerto a causa del disparo
      Marcar el objetivo como muerto (isDying = true, quitar de la
lista de unidades de MapScreenElement, mostrar animación y sonido de
muerte)
      Detener la orden de ataque
```

Código 31: Pseudocódigo de la orden de ataque

El Código 31 muestra cómo las unidades perseguirán a sus objetivos hasta poder dispararles. Para ello será necesario tiempo y una corta distancia. Sin embargo, el extracto de código está limitado a la teoría. Bastantes aspectos de las órdenes se encuentran ilustrados gráfica y/o auditivamente, como es el caso de los disparos y la muerte en la orden de ataque. A continuación se muestran algunas de las imágenes utilizadas para visualizar la orden de ataque en la práctica. Nótese el último frame siempre en blanco: se trata de animaciones no cíclicas, por lo que este frame sirve para hacer invisible la imagen temporalmente.

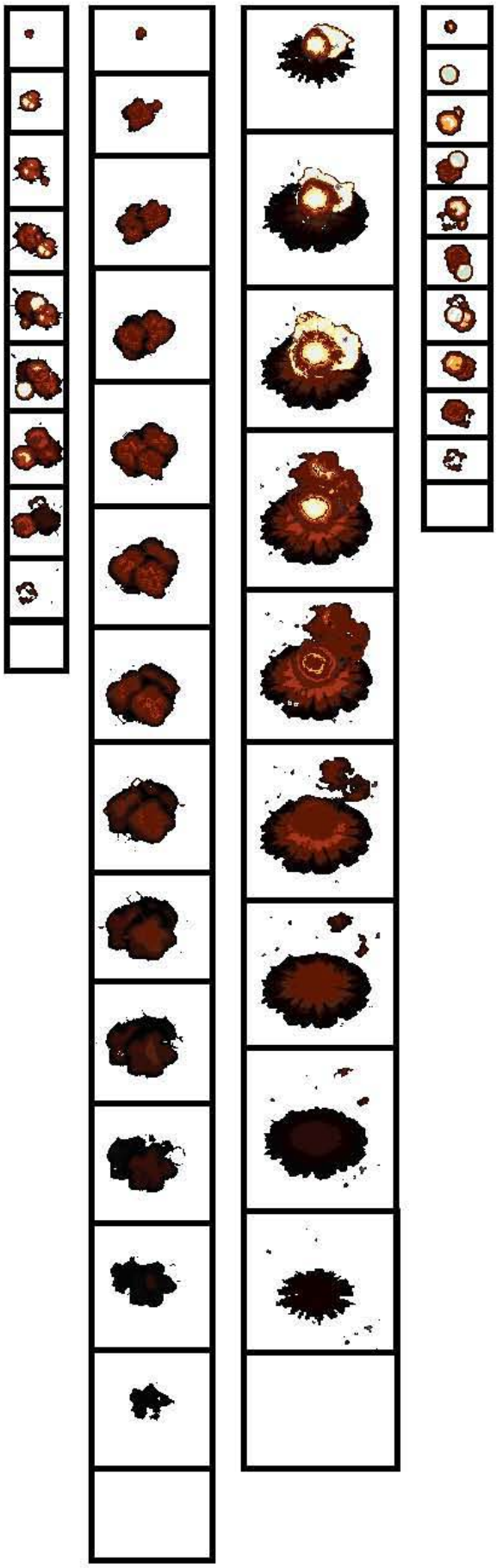

Figura 95. Ilustraciones de la orden "atacar"

Capítulo 6

Conclusiones

Este capítulo recoge las conclusiones extraídas del trabajo realizado. Para esta actividad se tienen en cuenta las expectativas planteadas al comienzo.

6.1. Conclusiones

Con carácter general, al concluir este proyecto se puede afirmar que el trabajo realizado cumple de manera satisfactoria con los objetivos planteados. A continuación se particularizan las conclusiones a diversos temas que ha sido necesario manejar a lo largo del proyecto.

En cuanto al videojuego en sí, es necesario destacar la gran similitud con el programa original, llevando a confusión a más de una persona no experimentada. El cliente de StarCraft desarrollado es, por tanto, bastante fiel a su versión original.

Sin embargo, no se trata de una réplica 1:1. La idea inicial, seguida al pie de la letra, era implementar suficientes funcionalidades como para tener la base necesaria para poder desarrollar un clon completo. En este sentido, como se ha detallado, se ha incluido un conjunto de pantallas, un sistema de mapas modificables a voluntad, un conjunto de unidades y edificios y un conjunto de órdenes. Mientras que el videojuego original es más amplio en todos los aspectos, lo que se ha implementado forma un subconjunto con sentido propio, siendo posible tanto tener una buena experiencia de juego como aumentar las funcionalidades en un futuro.

Respecto a la labor de desarrollo, es necesario destacar el papel de apoyo de Microsoft y de la enorme comunidad de usuarios de XNA. La existencia de una base de contenido funcional, ejemplos, documentación, tutoriales, recursos… es uno de los factores que seguramente haya influido más en la buena experiencia de trabajo día a día.

En lo referente a XNA, hay que reconocer que se proporciona un amplio abanico de soluciones a cada problema planteado. El trabajo realizado ha sido mayoritariamente en dos dimensiones, pero esta conclusión se supone igualmente válida para el trabajo en 3D. Desde los primeros pasos (dibujar una casilla en pantalla y desplazarla con el teclado) hasta el desarrollo completo, XNA siempre ha presentado alguna solución con la que empezar a trabajar.

El framework utilizado es muy completo, proporcionando por ejemplo acceso a todo tipo de entrada de usuario: teclado, ratón, cualquier mando de juego… También ofrece soluciones para la carga, modificación y dibujado de imágenes 2D y 3D. Las funciones matemáticas adicionales permiten manejar el control de colisiones sin problemas, al tiempo que permiten realizar otras implementaciones con libertad. Por último, es necesario destacar que el juego desarrollado no explota la sección de funciones de red de la librería, aunque el apoyo que ésta ofrece es extensivo y está bien documentado.

En cuanto a la motivación del proyecto y cumplimiento de objetivos personales se ha de reconocer que se han cubierto sobremanera las expectativas iniciales. Lo que comenzó siendo un hobby ha terminado siendo nada menos que un Proyecto de Fin de

Carrera, y el desarrollo actual no ha hecho sino acentuar el interés del alumno por el tema. Además, el trabajo realizado ha permitido conocer aspectos del día a día del desarrollo de proyectos y de videojuegos: planificación, estimación de recursos, análisis del futuro sistema, animación de personajes, control de movimiento, evaluación de colisiones, interacción con distintos periféricos, limitaciones de pantalla según la plataforma, etc.

Capítulo 7

Líneas futuras

Tras concluir el desarrollo que da vida al presente proyecto se decide realizar un análisis de posibles ampliaciones que es posible aplicar.

Este capítulo recoge las líneas de trabajo que, sin duda, abundan y por uno u otro motivo no se han visto implementadas.

7.1. Líneas futuras

Todo proyecto tiene unos plazos y unas formas, debiendo ceñirse a estos límites en la medida de lo posible. Existen muchas funcionalidades interesantes para un videojuego: un creador entusiasta y original podría dedicar semanas, meses e incluso años a imaginar e implementar. Esta sección del proyecto se dedica a recoger algunas de estas posibles mejoras, detallando los requisitos de implementación en la medida de lo posible.

En primer lugar, ya que el videojuego es una versión simplificada de otro, conviene destacar esta fuente de inclusión de funcionalidades. Estos trabajos tienen la mayor parte de la lógica programada. Algunos ejemplos son:

- Implementar una mayor diversidad de unidades. Sería necesario modificar la clase UnitScreenElement o producir una subclase. Los trabajos pendientes serían conseguir recursos gráficos y de sonido, permitir que fueran cargados, darles unos valores concretos de vida, daño, coste, etc. y permitir que algún edificio produjera estas nuevas unidades.
- Implementar una mayor diversidad de edificios. Esta mejora es muy similar a la anterior. Sólo se debe puntualizar que serán las unidades las que construyan los nuevos edificios, por lo que alguna de ellas deberá sufrir alguna modificación puntual.
- Implementar una mayor selección de colores. Esta modificación implica volver a colorear los recursos gráficos asociados a un bando e incluir la nueva opción en OptionsMenuScreen. Así mismo, la clase UnitScreenElement deberá tener constancia de los cambios, ya que es quien se ocupa de cargar los recursos gráficos de cada unidad.
- Implementar diferentes modos de juego. La clase MapScreenElement puede ser modificada para controlar las condiciones de victoria de alguno de los bandos. Algunas opciones disponibles se encuentran detalladas en el apartado **2.3.2. StarCraft en detalle**.
- Implementar otras razas. Esta modificación permitiría un parecido inigualable con la versión original del juego, pero también es la que comporta una mayor cantidad de trabajo. En este sentido se aconseja replicar la mayor parte de la lógica existente en UnitScreenElement, aunque es necesario cambiar algunos aspectos para permitir funcionalidades específicas de cada raza. Por ejemplo, se ha implementado la construcción Terran, pero el proceso constructivo Protoss es muy diferente.

Por otro lado, se podría permitir el juego en red. Esta funcionalidad está fuertemente relacionada con el videojuego original. La mayoría de los juegos del catálogo de XBOX 360 no disponen de funcionalidad en red. Esto se debe a que las versiones de XNA anteriores a la 3.1 no ofrecían un soporte demasiado amplio. Por

tanto, se puede conseguir una clara ventaja sobre otros desarrollos mediante la comunicación de los datos de MapScreenElement a otros clientes de StarCraft.

En el sentido de otorgar funcionalidad en red, se podría llevar a cabo de manera sencilla mediante la dirección IP de los otros jugadores. Sin embargo, para sacar el mayor provecho posible se aconseja la implementación de un servidor que tenga constancia de los clientes conectados, pudiendo crear partidas igualadas en función del nivel de experiencia, aleatoriamente o permitiendo a los jugadores intercambiar opiniones.

Otra posible funcionalidad sería guardar la partida en curso. Esta funcionalidad consistiría en escribir en disco el estado actual de la partida: las instancias creadas y los valores de las variables. Para recuperar una partida sería necesario leer el fichero donde fue guardada y trasladar los valores leídos de nuevo a sus variables. Para llevar a cabo esta funcionalidad también será necesario modificar algún menú: por ejemplo, MainMenuScreen puede tener una opción de “Cargar partida”, mientras que PauseMenuScreen presenta la opción “Guardar partida”.

Por último, una funcionalidad muy utilizada en el videojuego StarCraft original es el editor de mapas. Para realizar esta mejora se aconseja crear un programa separado que permita elegir la imagen asociada a cada casilla, así como especificar si las casillas son transitables. También será necesario crear mosaicos de casillas variados, posiblemente simulando otros terrenos (nieve, espacio exterior, jungla…). Finalmente, este proyecto detalla el formato de los ficheros de mapa, así que sería necesario producir la salida del editor de mapas siguiendo la especificación propuesta.

Anexo A

Planificación

A.1. Introducción

La base común a todo proyecto es la planificación. Después de comenzar y realizar un pequeño estudio de viabilidad, siempre es necesario establecer las tareas que se han de acometer y los tiempos asignados para ello. Esta planificación podrá ser alterada y debe ser actualizada en función de la marcha del proyecto.

El motivo principal para realizar esta contabilidad es tener el control sobre la marcha del proyecto. Si se divide en etapas o fases se facilitan mucho los cálculos de plazos y recursos a emplear, y se tendrá mejor aproximación acerca del coste del proyecto (y, por tanto, del riesgo económico inherente).

Este Anexo se estructura de la siguiente manera: en primer lugar se analiza el ciclo de vida más común en el sector; después se detallará la planificación inicial y final del proyecto; para terminar se ofrecerán los presupuestos inicial y final y se investigará sobre la publicación del juego en el mercado de XBOX 360.

A.2. Ciclo de vida de un videojuego comercial

Esta sección se dedica a detallar el ciclo de vida típico de un videojuego. Se aportan estos conocimientos a título informativo, aunque también se espera proporcionar un punto de referencia para comprender la planificación realizada.

A continuación se detallan las fases que, secuencialmente, atravesará un proyecto dedicado a crear un videojuego comercial.

- **Concepto**: en una fase tan temprana sólo cabe definir el concepto que se encontrará en el corazón del desarrollo. Se parte de un conjunto de ideas, se crea una propuesta de juego y se ilustra con dibujos artísticos muy refinados. Si la idea prospera, en esta fase se podrá definir el género de videojuego, el ambiente general, la plataforma de desarrollo y de despliegue…

- **Pre-Producción**: tras haber aceptado un nuevo proyecto, esta etapa se dedica a realizar el necesario estudio de viabilidad, así como la planificación que se aplicará. Otro producto de esta fase es la guía del juego: un documento de análisis que detalla la mecánica de juego, los niveles, los personajes, la interfaz de usuario, la trama histórica… Para concluir se detalla un plan de hitos y prototipos.

- **Producción**: en esta fase da comienzo la construcción real del producto final. Las tareas principales son de escritura de código, creación de gráficos, edición

de niveles y sonidos… En cuanto sea posible, el personal encargado comenzará a probar el sistema. En esta fase puede modificarse el concepto inicial del videojuego por motivos técnicos, aunque los cambios serán menores.

- **Alfa**: al llegar a este punto, el desarrollo puede ser jugado de principio a fin, aunque pueden quedar detalles sin terminar. Se tendrá terminada buena parte del sistema gráfico, de la interfaz de usuario y de otros subsistemas importantes (acceso a disco, niveles…). En este punto la producción de material adquiere un papel secundario a favor de la verificación, la calidad y la realización de pruebas.

- **Beta**: este hito es alcanzado cuando todas las funcionalidades están desarrolladas. El desarrollo se detiene, activándose únicamente a causa de los defectos observados durante las pruebas. El objetivo es tener un producto muy estable y con la mínima cantidad de defectos ya que, al liberar el juego, todos ellos serán solucionados en base a actualizaciones.

- **Congelación de código**: cuando se da por concluida la fase Beta se obtiene el código para la liberación final. Esta obtención se conoce como congelación de código. El programa así obtenido es declarado apto para ser etiquetado, embalado y distribuido a los medios de venta al público.

- **Liberación**: etapa consistente en la comercialización del producto propiamente dicha. Abarca desde la congelación de código hasta la venta de los productos en tiendas o por Internet.

- **Parches**: sistema de corrección de defectos que entra en acción después de liberar y comercializar el producto. Los propios usuarios colaboran con la empresa para corregir los problemas que vayan surgiendo a medida que se juega. Esto forma parte del sistema de garantía de cualquier compra, por lo que la compañía no es remunerada debido a esta actividad.

- **Actualizaciones**: en el mismo entorno que los parches, las actualizaciones son modificaciones sustanciales del producto original destinadas a ofrecer nuevas funcionalidades. Pueden contener cambios de todo tipo: desde mejoras de código hasta nuevas tramas históricas, pasando por habilitar el juego para varios jugadores, introducir nuevas reglas de juego… A diferencia de los parches, este tipo de contenido puede (y suele) ser de pago.

A.3. Planificación y etapas del proyecto

Este proyecto fue concebido desde su inicio para ser realizado mediante el Modelo de Desarrollo Evolutivo (o Modelo de Prototipado Evolutivo). Esta decisión, si bien arbitraria, se adapta naturalmente a lo expuesto en el apartado A.2, **Ciclo de vida de un desarrollo comercial**.

Este modelo toma como base la suposición de que los requisitos del proyecto no son conocidos completamente y con exactitud al comenzar el proyecto. Por tanto, se crean desarrollos parciales que serán incrementales y evolutivos, produciendo finalmente una versión completa y estable.

La forma de trabajo descrita ha regido la evolución del proyecto que ocupa este documento de manera muy apropiada. Los motivos para afirmar esto son dos:

- Se ha podido validar el progreso cada poco tiempo. Esto lleva a la corrección temprana de defectos de implementación, errores de planteamiento… ahorrando así una buena cantidad de tiempo.

- Ha sido posible que el conjunto de requisitos no fuera cerrado y estable desde el principio del proyecto. Esta medida se cree adecuada para la asignación de tiempo que un proyecto de este tipo suele recibir.

En definitiva, el prototipado dinámico realizado ha probado ser muy beneficioso para la realización de un Proyecto de Fin de Carrera orientado al desarrollo de un sistema software.

Siguiendo el modelo descrito, el proyecto es cuidadosamente analizado. Esto da lugar al conjunto inicial de requisitos. Los más significativos son seleccionados para el primer incremento, que dará lugar a la primera construcción (parcial) del sistema.

Esta versión parcial es evaluada, lo que provoca un flujo de información que ayuda tanto al análisis como al desarrollo. La siguiente versión incluye la información descubierta, incrementa el conjunto de requisitos cumplido y continúa el flujo de información. Este proceso es repetido hasta conseguir una versión completa que incluya todos los requisitos (versiones “Alfa” y/o “Beta”).

Todo lo reflejado anteriormente puede ser visualizado de forma gráfica en la siguiente imagen (Figura 96).

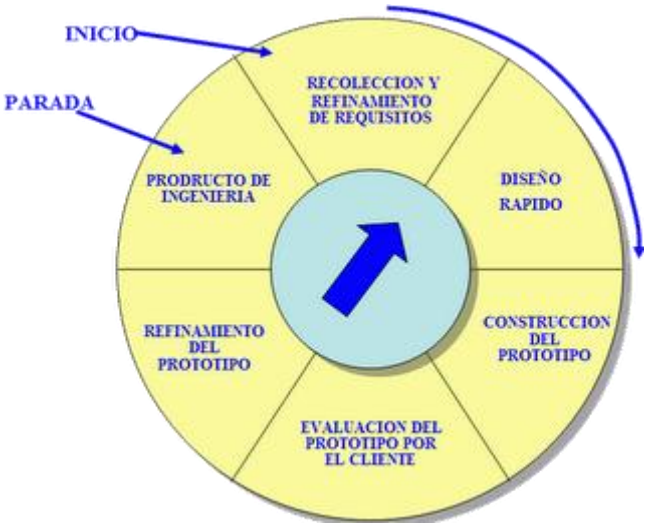

Figura 96. Modelo de Prototipado Evolutivo

Tras establecer el modelo de desarrollo utilizado se pasa a detallar las fases del proyecto actual. Cada una de ellas dará lugar al mencionado prototipo, así como a la información necesaria para comenzar siguiente fase. Las cinco fases establecidas en función del grupo de requisito que tratan son:

- Generales: menús, opciones, tamaño de ventana, establecimiento del sistema de desarrollo y de ejecución…
- Mapa: análisis, diseño, desarrollo y validación del mapa (casillas, interacción…).
- Unidades y edificios: análisis, diseño, desarrollo y validación de las unidades y edificios implementados.
- Barra de control y órdenes: análisis, diseño, desarrollo y validación de la barra de control y de su contenido (mapa en miniatura, información adicional, órdenes…), así como las órdenes propiamente dichas.
- Música, efectos de sonido y otros: incorporación al proyecto de aspectos finales (recursos, música, efectos de sonido, volumen de ambos, teclas de volumen…).

A.3.1. Planificación inicial

En esta sección se presenta la estimación inicial del tiempo y recursos asignados a las fases del proyecto.

Formación preparatoria sobre la tecnología de desarrollo	**25 días**
Estudio documentación Visual Studio.NET	25 días
Estudio documentación XNA	25 días
Desarrollo de documentación y memoria	**155 días**
Fase 1 – Requisitos Generales	**20 días**
Requisitos Fase 1:	5 días
Aspecto y dimensiones de ventana	4 días
Utilización básica de imágenes	5 días
Realización de menús interconectados	5 días
Interrupción y salida del juego	5 días
Diseño	5 días
Implementación	8 días
Evaluación	2 días
Fase 2 – Requisitos sobre el Mapa	**25 días**
Requisitos Fase 2:	5 días
Diseño gráfico y adaptación de imágenes	2 días
Lógica de niveles	5 días
Lógica de casillas	2 días
Diseño	5 días
Implementación	12 días
Evaluación	3 días
Fase 3 – Requisitos sobre Unidades y Edificios	**25 días**
Requisitos Fase 3:	5 días
Diseño gráfico y adaptación de imágenes	2 días
Control y desplazamiento	5 días
Lógica de presentación en pantalla	5 días

Tabla 166: Tareas iniciales del proyecto (parte 1ª)

Detección y tratamiento de colisiones	5 días
Diseño	5 días
Implementación	12 días
Evaluación	3 días
Fase 4 – Requisitos sobre Barra de control y órdenes	**35 días**
Requisitos Fase 4:	10 días
Diseño gráfico y adaptación de imágenes	4 días
Mapa en miniatura	5 días
Implementación de órdenes	10 días
Presentación de información adicional	5 días
Diseño	5 días
Implementación	15 días
Evaluación	5 días
Fase 5 – Otros requisitos	**20 días**
Requisitos Fase 5:	5 días
Adaptación de sonidos y música	4 días
Implementación de sonidos y música	5 días
Finalización de menús: opciones	1 días
Diseño	5 días
Implementación	5 días
Evaluación	5 días
Evaluación y pruebas finales	**5 días**

Tabla 167: Tareas iniciales del proyecto (parte 2ª)

Cada tarea señalada debe tener asociado un conjunto de recursos para ser realizada (material, humano…). Por otro lado, se decidió realizar una reunión periódica de seguimiento del proyecto aproximadamente cada tres semanas, de forma que se observara cuidadosamente la evolución de la planificación y del proyecto. Estos dos

aspectos han sido tenidos en cuenta para realizar el diagrama de Gantt inicial que se ofrece a continuación.

Id	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
1	**Formación preparatoria sobre la tecnología de desarrollo**	**25 días**	**07/09/10**	**11/10/10**	**PC**
2	Estudio documentación Visual Studio.NET	25 días	07/09/10	11/10/10	Programador
3	Estudio documentación XNA	25 días	07/09/10	11/10/10	Programador
4	**Desarrollo de documentación y memoria**	**155 días**	**07/09/10**	**11/04/11**	**PC**
5	Desarrollo de documentación y memoria	155 días	07/09/10	11/04/11	Programador
6	**Fase 1 – Requisitos Generales**	**20,25 días**	**12/10/10**	**09/11/10**	**PC**
7	**Requisitos Fase 1:**	**5 días**	**12/10/10**	**18/10/10**	
8	Aspecto y dimensiones de ventana	4 días	12/10/10	15/10/10	Artista Conceptual;Adobe Photoshop CS3
9	Utilización básica de imágenes	5 días	12/10/10	18/10/10	Analista
10	Realización de menús interconectados	5 días	12/10/10	18/10/10	Analista
11	Interrupción y salida del juego	5 días	12/10/10	18/10/10	Analista
12	Diseño	5 días	19/10/10	25/10/10	Analista
13	Implementación	8 días	26/10/10	04/11/10	Programador
14	Evaluación	2 días	05/11/10	08/11/10	Testeador
15	Reunión	2 horas	09/11/10	09/11/10	Analista;Jefe de Proyecto
16	**Fase 2 – Requisitos sobre el Mapa**	**25,25 días**	**09/11/10**	**14/12/10**	**PC**
17	**Requisitos Fase 2:**	**5 días**	**09/11/10**	**16/11/10**	
18	Diseño gráfico y adaptación de imágenes	2 días	09/11/10	11/11/10	Artista Conceptual;Adobe Photoshop CS3
19	Lógica de niveles	5 días	09/11/10	16/11/10	Analista
20	Lógica de casillas	2 días	09/11/10	11/11/10	Analista
21	Diseño	5 días	16/11/10	23/11/10	Analista
22	Implementación	12 días	23/11/10	09/12/10	Programador
23	Evaluación	3 días	09/12/10	14/12/10	Testeador
24	Reunión	2 horas	14/12/10	14/12/10	Analista;Jefe de Proyecto
25	**Fase 3 – Requisitos sobre Unidades y Edificios**	**25,25 días**	**14/12/10**	**18/01/11**	**PC**
26	**Requisitos Fase 3:**	**5 días**	**14/12/10**	**21/12/10**	
27	Diseño gráfico y adaptación de imágenes	2 días	14/12/10	16/12/10	Artista Conceptual;Adobe Photoshop CS3
28	Control y desplazamiento	5 días	14/12/10	21/12/10	Analista;Programador
29	Lógica de presentación en pantalla	5 días	14/12/10	21/12/10	Analista;Programador
30	Detección y tratamiento de colisiones	5 días	14/12/10	21/12/10	Analista
31	Diseño	5 días	21/12/10	28/12/10	Analista
32	Implementación	12 días	28/12/10	13/01/11	Programador
33	Evaluación	3 días	13/01/11	18/01/11	Testeador
34	Reunión	2 horas	18/01/11	18/01/11	Analista;Jefe de Proyecto
35	**Fase 4 – Requisitos sobre Barra de control y órdenes**	**35,25 días**	**18/01/11**	**08/03/11**	**PC**
36	**Requisitos Fase 4:**	**10 días**	**18/01/11**	**01/02/11**	
37	Diseño gráfico y adaptación de imágenes	4 días	18/01/11	24/01/11	Artista Conceptual;Adobe Photoshop CS3
38	Mapa en miniatura	5 días	18/01/11	25/01/11	Analista;Programador
39	Implementación de órdenes	10 días	18/01/11	01/02/11	Analista;Programador
40	Presentación de información adicional	5 días	18/01/11	25/01/11	Analista
41	Diseño	5 días	01/02/11	08/02/11	Analista
42	Implementación	15 días	08/02/11	01/03/11	Programador
43	Evaluación	5 días	01/03/11	08/03/11	Testeador
44	Reunión	2 horas	08/03/11	08/03/11	Analista;Jefe de Proyecto
45	**Fase 5 – Otros requisitos**	**20,25 días**	**09/03/11**	**06/04/11**	**PC**
46	**Requisitos Fase 5:**	**5 días**	**09/03/11**	**15/03/11**	
47	Adaptación de sonidos y música	4 días	09/03/11	14/03/11	Artista Conceptual
48	Implementación de sonidos y música	5 días	09/03/11	15/03/11	Analista;Programador
49	Finalización de menús: opciones	1 día	09/03/11	09/03/11	Analista;Programador
50	Diseño	5 días	16/03/11	22/03/11	Analista
51	Implementación	5 días	23/03/11	29/03/11	Programador
52	Evaluación	5 días	30/03/11	05/04/11	Testeador
53	Reunión	2 horas	06/04/11	06/04/11	Analista;Jefe de Proyecto
54	**Evaluación**	**5,25 días**	**06/04/11**	**13/04/11**	**Programador;Testeador**
55	Evaluación y pruebas finales	5 días	06/04/11	13/04/11	Programador;Testeador
56	Reunión final	2 horas	13/04/11	13/04/11	Analista;Artista Conceptual;Programador;Testeador;Jefe de Proyecto

Figura 97. Diagrama de Gantt inicial - Tareas

Id	Nombre de tarea
1	**Formación preparatoria sobre la tecnología de desarrollo**
2	Estudio documentación Visual Studio.NET
3	Estudio documentación XNA
4	**Desarrollo de documentación y memoria**
5	Desarrollo de documentación y memoria
6	**Fase 1 – Requisitos Generales**
7	**Requisitos Fase 1:**
8	Aspecto y dimensiones de ventana
9	Utilización básica de imágenes
10	Realización de menús interconectados
11	Interrupción y salida del juego
12	Diseño
13	Implementación
14	Evaluación
15	Reunión
16	**Fase 2 – Requisitos sobre el Mapa**
17	**Requisitos Fase 2:**
18	Diseño gráfico y adaptación de imágenes
19	Lógica de niveles
20	Lógica de casillas
21	Diseño
22	Implementación
23	Evaluación
24	Reunión
25	**Fase 3 – Requisitos sobre Unidades y Edificios**
26	**Requisitos Fase 3:**
27	Diseño gráfico y adaptación de imágenes
28	Control y desplazamiento
29	Lógica de presentación en pantalla
30	Detección y tratamiento de colisiones
31	Diseño
32	Implementación
33	Evaluación
34	Reunión
35	**Fase 4 – Requisitos sobre Barra de control y órdenes**
36	**Requisitos Fase 4:**
37	Diseño gráfico y adaptación de imágenes
38	Mapa en miniatura
39	Implementación de órdenes
40	Presentación de información adicional
41	Diseño
42	Implementación
43	Evaluación
44	Reunión
45	**Fase 5 – Otros requisitos**
46	**Requisitos Fase 5:**
47	Adaptación de sonidos y música
48	Implementación de sonidos y música
49	Finalización de menús: opciones
50	Diseño
51	Implementación
52	Evaluación
53	Reunión
54	**Evaluación**
55	Evaluación y pruebas finales
56	Reunión final

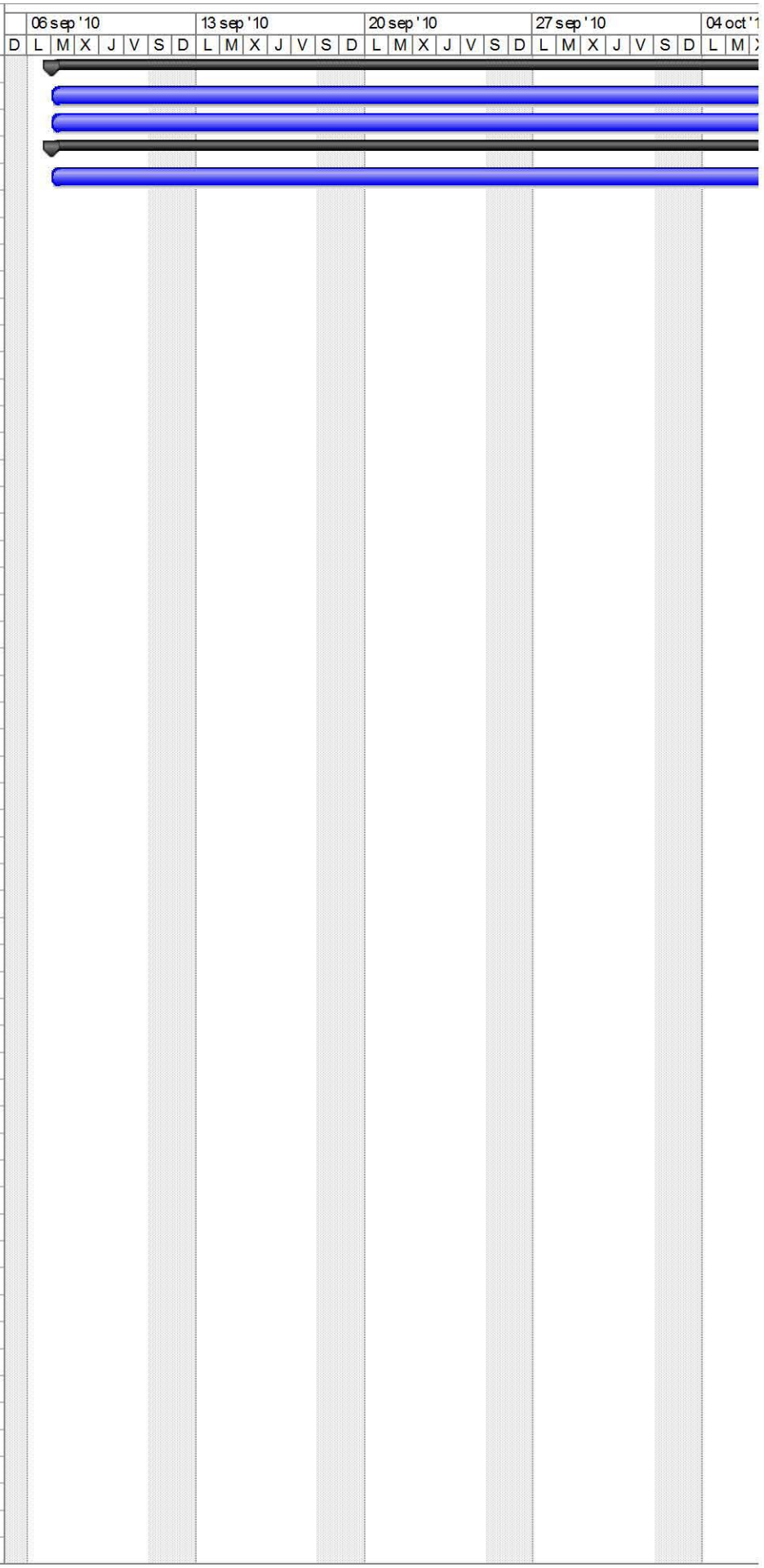

Figura 98. Diagrama de Gantt inicial – Planificación 1

Id	Nombre de tarea
1	**Formación preparatoria sobre la tecnología de desarrollo**
2	Estudio documentación Visual Studio.NET
3	Estudio documentación XNA
4	**Desarrollo de documentación ymemoria**
5	Desarrollo de documentación y memoria
6	**Fase 1 – Requisitos Generales**
7	**Requisitos Fase 1:**
8	Aspecto ydimensiones de ventana
9	Utilización básica de imágenes
10	Realización de menús interconectados
11	Interrupción y salida del juego
12	Diseño
13	Implementación
14	Evaluación
15	Reunión
16	**Fase 2 – Requisitos sobre el Mapa**
17	**Requisitos Fase 2:**
18	Diseño gráfico y adaptación de imágenes
19	Lógica de niveles
20	Lógica de casillas
21	Diseño
22	Implementación
23	Evaluación
24	Reunión
25	**Fase 3 – Requisitos sobre Unidades yEdificios**
26	**Requisitos Fase 3:**
27	Diseño gráfico y adaptación de imágenes
28	Control y desplazamiento
29	Lógica de presentación en pantalla
30	Detección y tratamiento de colisiones
31	Diseño
32	Implementación
33	Evaluación
34	Reunión
35	**Fase 4 – Requisitos sobre Barra de control yórdenes**
36	**Requisitos Fase 4:**
37	Diseño gráfico y adaptación de imágenes
38	Mapa en miniatura
39	Implementación de órdenes
40	Presentación de información adicional
41	Diseño
42	Implementación
43	Evaluación
44	Reunión
45	**Fase 5 – Otros requisitos**
46	**Requisitos Fase 5:**
47	Adaptación de sonidos ymúsica
48	Implementación de sonidos ymúsica
49	Finalización de menús: opciones
50	Diseño
51	Implementación
52	Evaluación
53	Reunión
54	**Evaluación**
55	Evaluación ypruebas finales
56	Reunión final

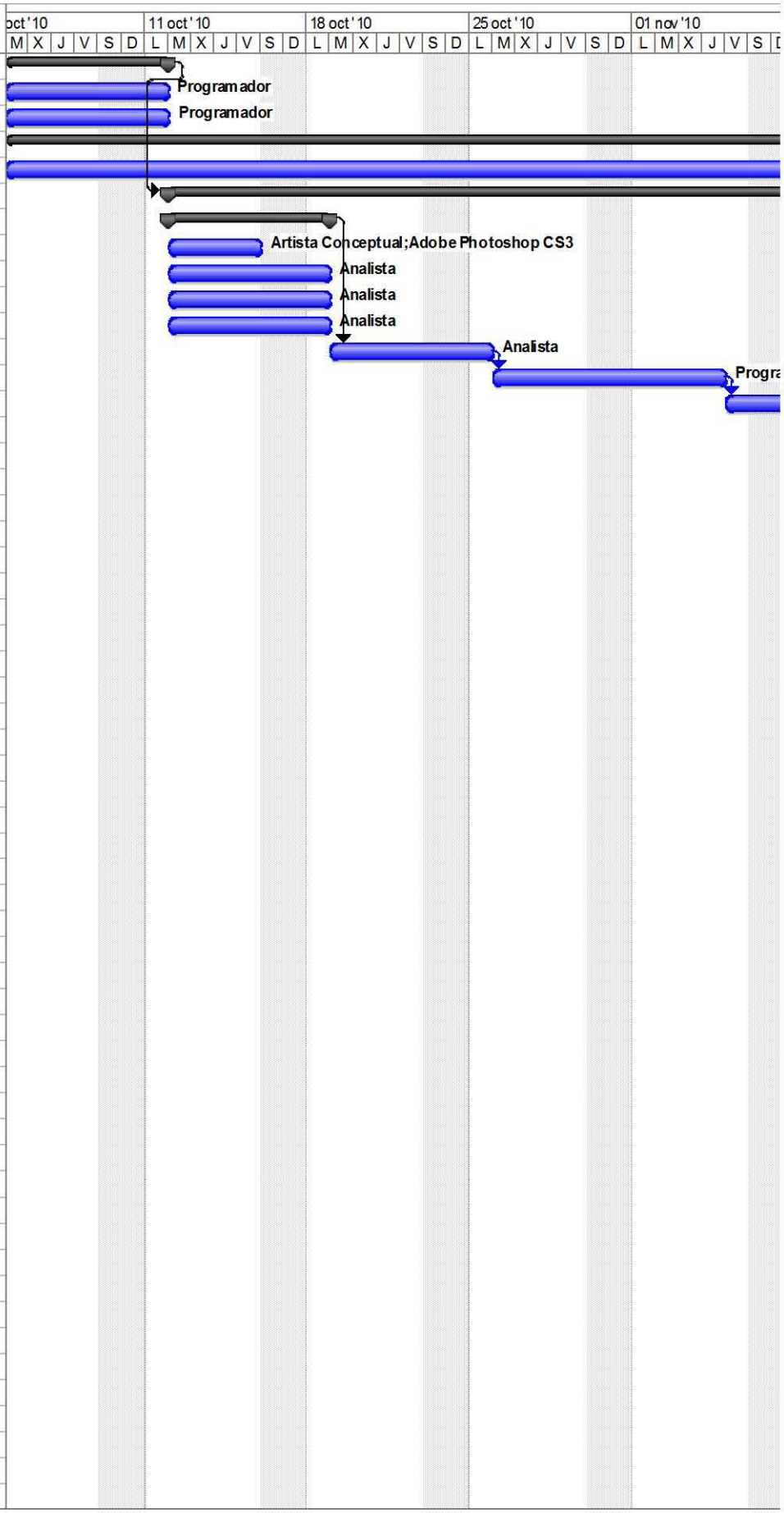

Figura 99. Diagrama de Gantt inicial – Planificación 2

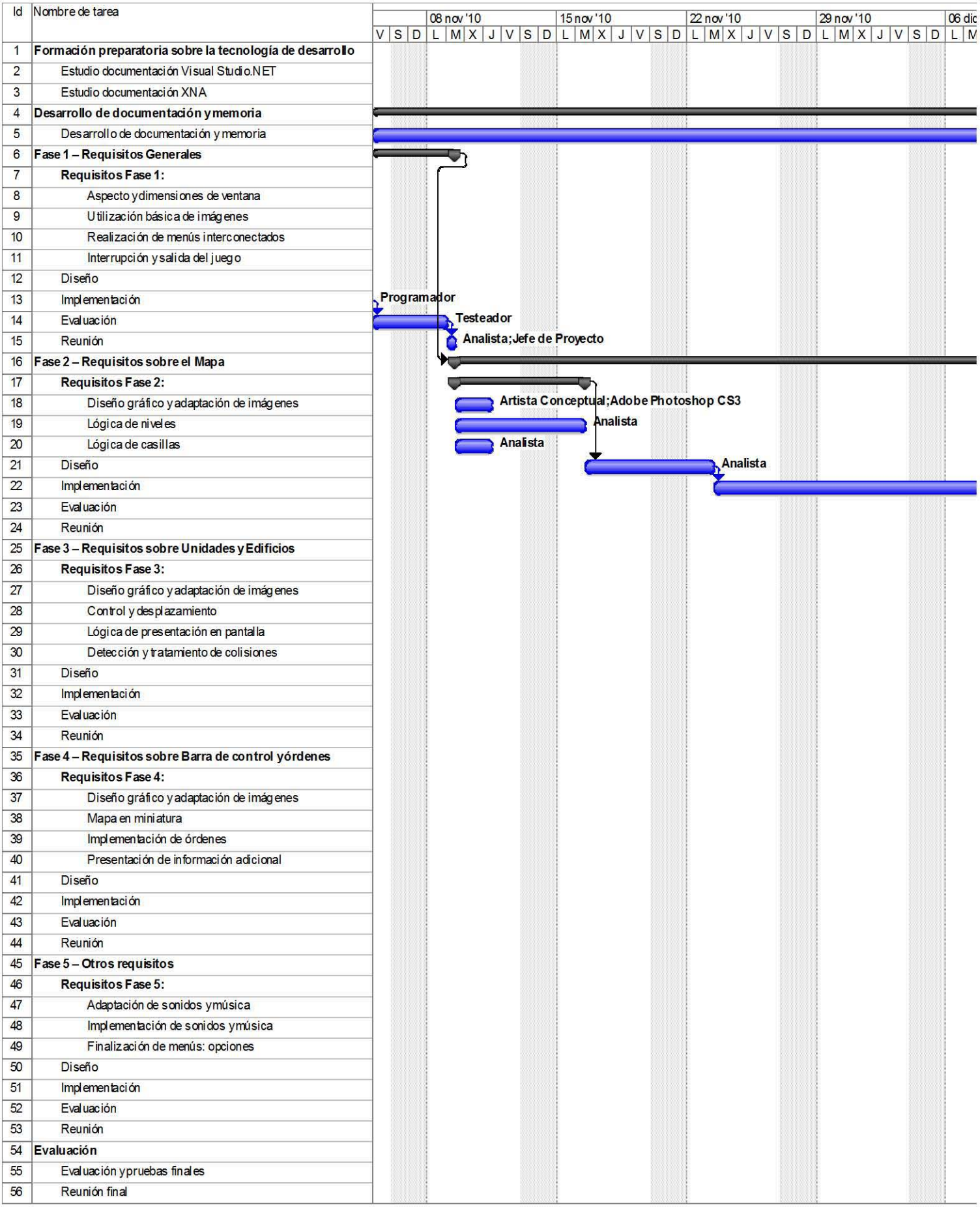

Figura 100. Diagrama de Gantt inicial – Planificación 3

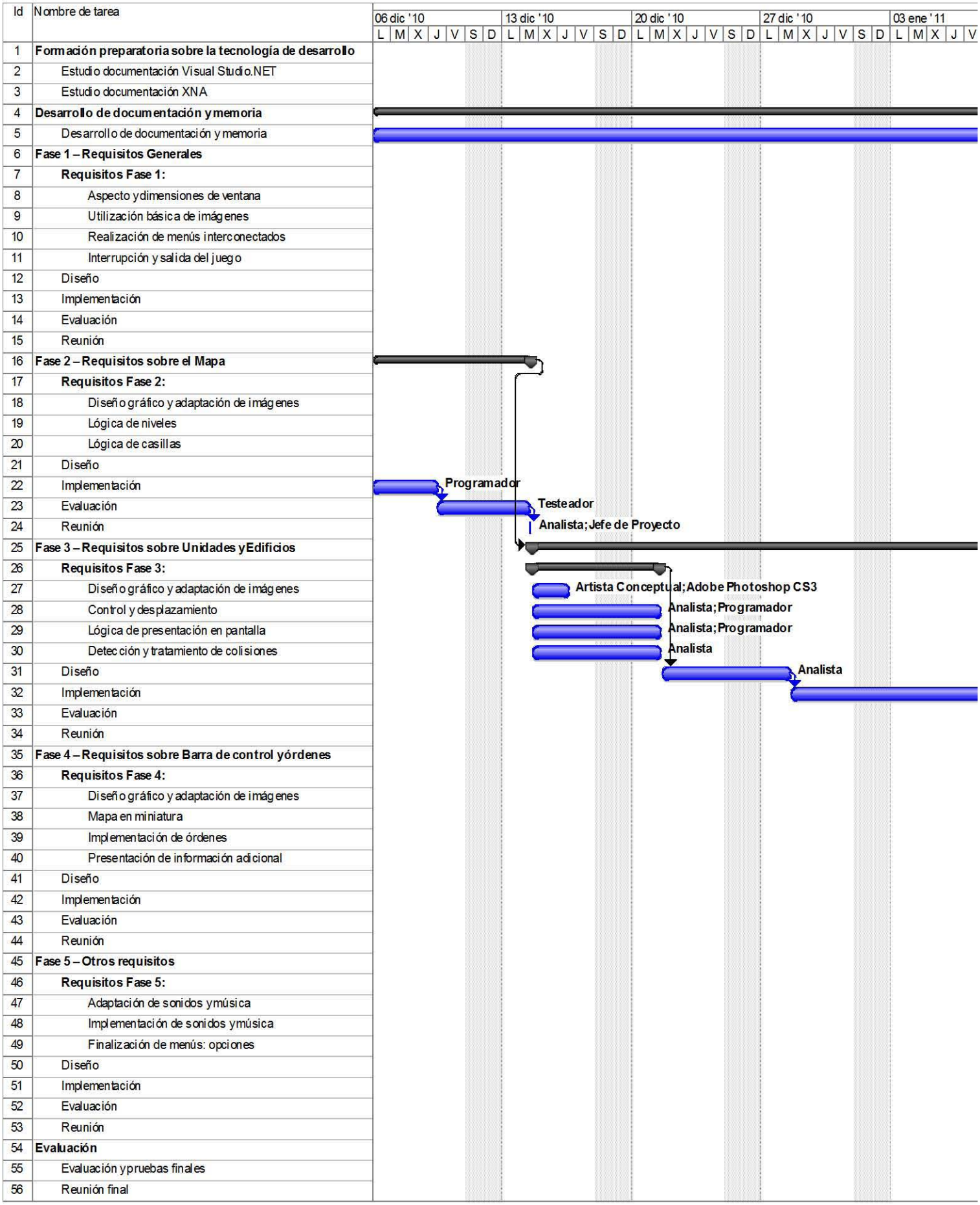

Figura 101. Diagrama de Gantt inicial – Planificación 4

Id	Nombre de tarea
1	**Formación preparatoria sobre la tecnología de desarrollo**
2	Estudio documentación Visual Studio.NET
3	Estudio documentación XNA
4	**Desarrollo de documentación y memoria**
5	Desarrollo de documentación y memoria
6	**Fase 1 – Requisitos Generales**
7	**Requisitos Fase 1:**
8	Aspecto y dimensiones de ventana
9	Utilización básica de imágenes
10	Realización de menús interconectados
11	Interrupción y salida del juego
12	Diseño
13	Implementación
14	Evaluación
15	Reunión
16	**Fase 2 – Requisitos sobre el Mapa**
17	**Requisitos Fase 2:**
18	Diseño gráfico y adaptación de imágenes
19	Lógica de niveles
20	Lógica de casillas
21	Diseño
22	Implementación
23	Evaluación
24	Reunión
25	**Fase 3 – Requisitos sobre Unidades y Edificios**
26	**Requisitos Fase 3:**
27	Diseño gráfico y adaptación de imágenes
28	Control y desplazamiento
29	Lógica de presentación en pantalla
30	Detección y tratamiento de colisiones
31	Diseño
32	Implementación
33	Evaluación
34	Reunión
35	**Fase 4 – Requisitos sobre Barra de control y órdenes**
36	**Requisitos Fase 4:**
37	Diseño gráfico y adaptación de imágenes
38	Mapa en miniatura
39	Implementación de órdenes
40	Presentación de información adicional
41	Diseño
42	Implementación
43	Evaluación
44	Reunión
45	**Fase 5 – Otros requisitos**
46	**Requisitos Fase 5:**
47	Adaptación de sonidos y música
48	Implementación de sonidos y música
49	Finalización de menús: opciones
50	Diseño
51	Implementación
52	Evaluación
53	Reunión
54	**Evaluación**
55	Evaluación y pruebas finales
56	Reunión final

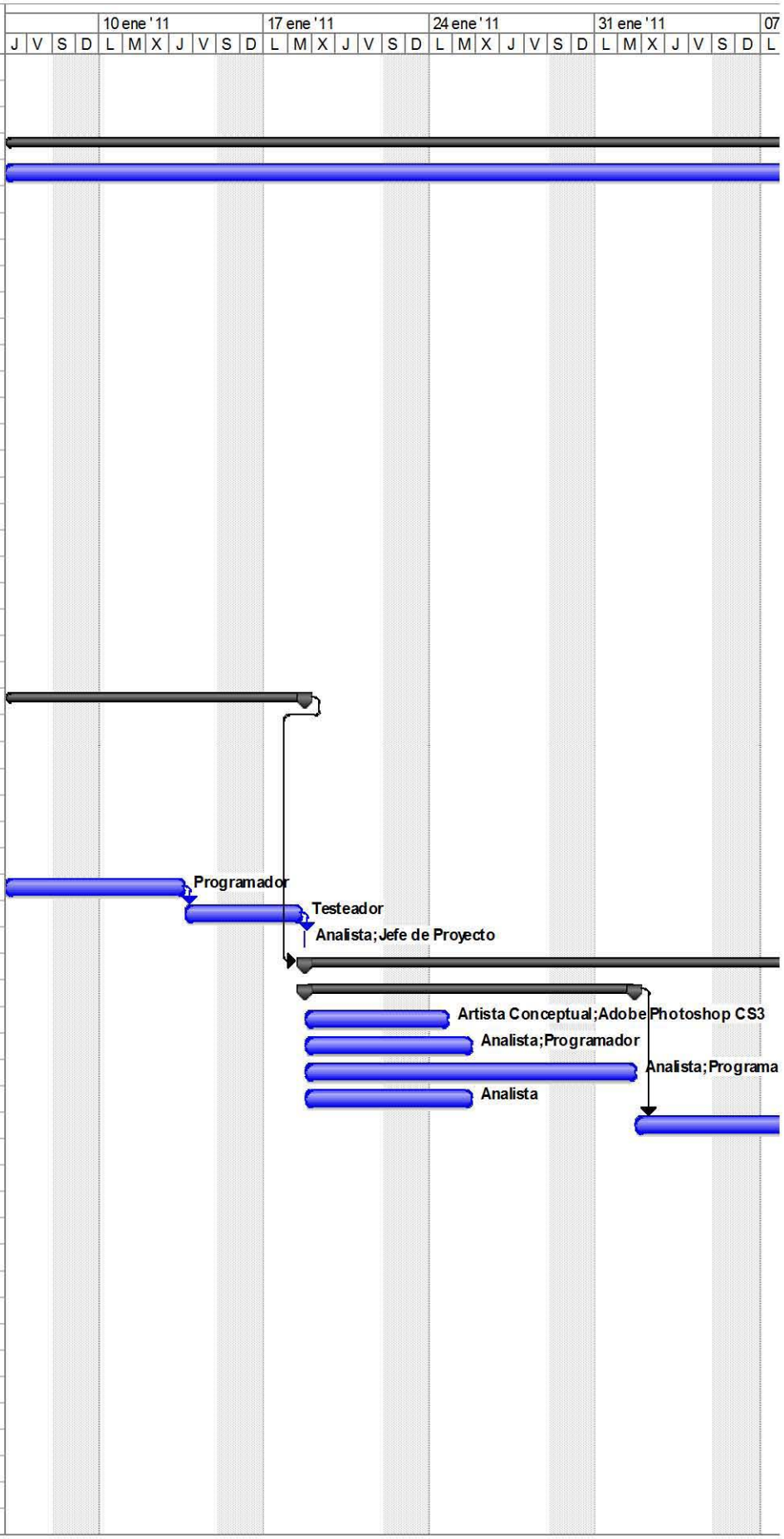

Figura 102. Diagrama de Gantt inicial – Planificación 5

Id	Nombre de tarea
1	**Formación preparatoria sobre la tecnología de desarrollo**
2	Estudio documentación Visual Studio.NET
3	Estudio documentación XNA
4	**Desarrollo de documentación y memoria**
5	Desarrollo de documentación y memoria
6	**Fase 1 – Requisitos Generales**
7	**Requisitos Fase 1:**
8	Aspecto y dimensiones de ventana
9	Utilización básica de imágenes
10	Realización de menús interconectados
11	Interrupción y salida del juego
12	Diseño
13	Implementación
14	Evaluación
15	Reunión
16	**Fase 2 – Requisitos sobre el Mapa**
17	**Requisitos Fase 2:**
18	Diseño gráfico y adaptación de imágenes
19	Lógica de niveles
20	Lógica de casillas
21	Diseño
22	Implementación
23	Evaluación
24	Reunión
25	**Fase 3 – Requisitos sobre Unidades y Edificios**
26	**Requisitos Fase 3:**
27	Diseño gráfico y adaptación de imágenes
28	Control y desplazamiento
29	Lógica de presentación en pantalla
30	Detección y tratamiento de colisiones
31	Diseño
32	Implementación
33	Evaluación
34	Reunión
35	**Fase 4 – Requisitos sobre Barra de control y órdenes**
36	**Requisitos Fase 4:**
37	Diseño gráfico y adaptación de imágenes
38	Mapa en miniatura
39	Implementación de órdenes
40	Presentación de información adicional
41	Diseño
42	Implementación
43	Evaluación
44	Reunión
45	**Fase 5 – Otros requisitos**
46	**Requisitos Fase 5:**
47	Adaptación de sonidos y música
48	Implementación de sonidos y música
49	Finalización de menús: opciones
50	Diseño
51	Implementación
52	Evaluación
53	Reunión
54	**Evaluación**
55	Evaluación y pruebas finales
56	Reunión final

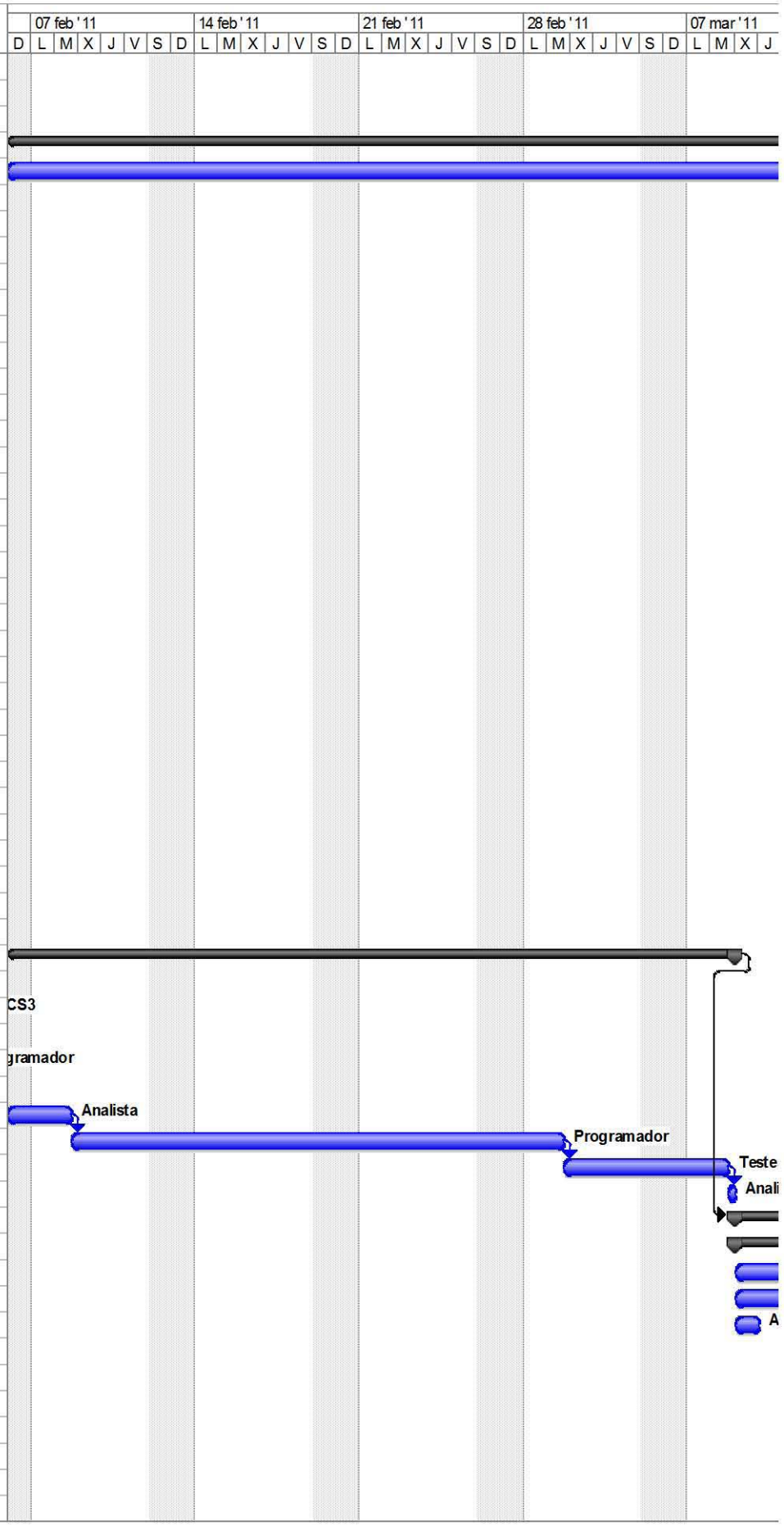

Figura 103. Diagrama de Gantt inicial – Planificación 6

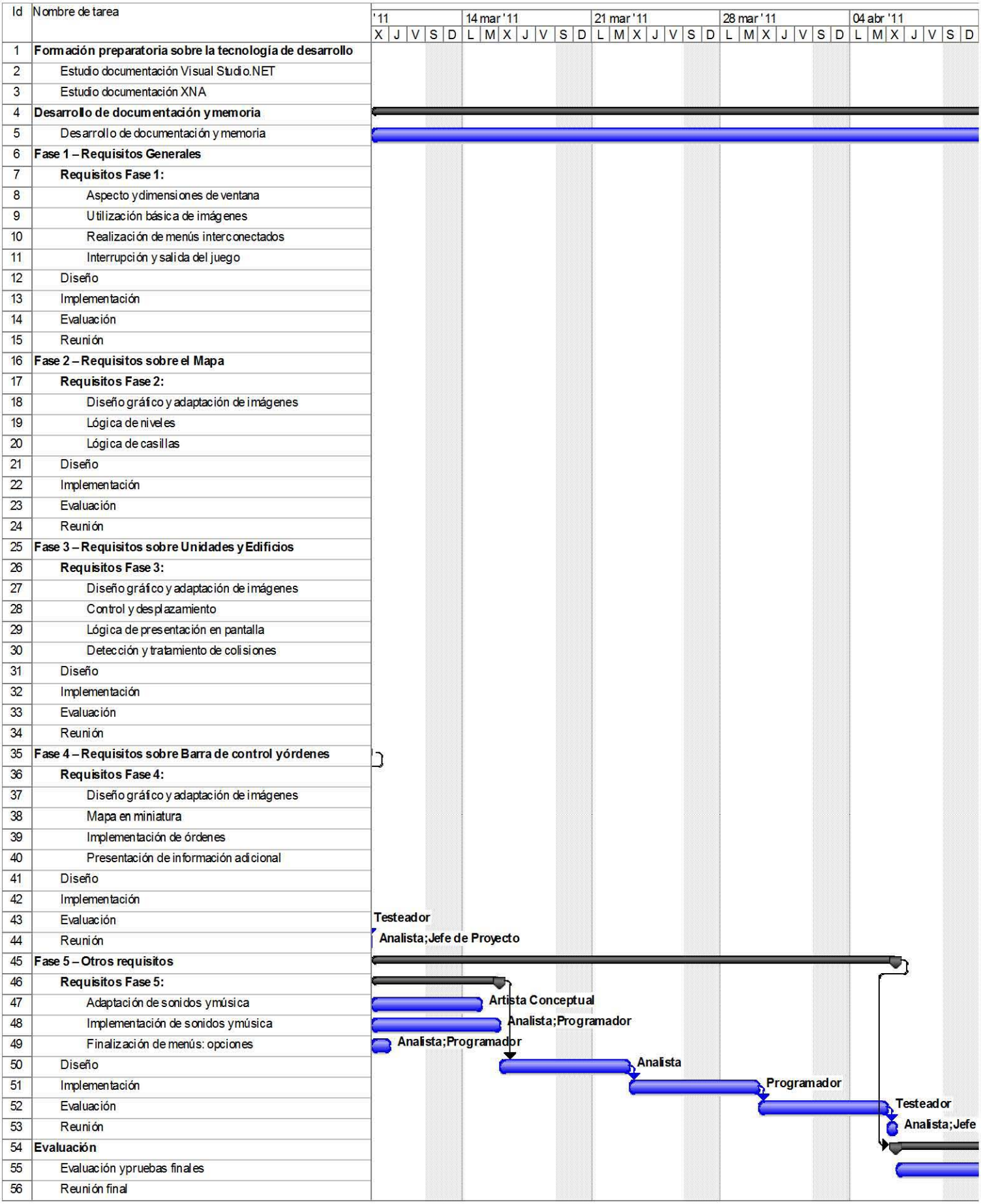

Figura 104. Diagrama de Gantt inicial – Planificación 7

Id	Nombre de tarea
1	**Formación preparatoria sobre la tecnología de desarrollo**
2	Estudio documentación Visual Studio.NET
3	Estudio documentación XNA
4	**Desarrollo de documentación y memoria**
5	Desarrollo de documentación y memoria
6	**Fase 1 – Requisitos Generales**
7	**Requisitos Fase 1:**
8	Aspecto y dimensiones de ventana
9	Utilización básica de imágenes
10	Realización de menús interconectados
11	Interrupción y salida del juego
12	Diseño
13	Implementación
14	Evaluación
15	Reunión
16	**Fase 2 – Requisitos sobre el Mapa**
17	**Requisitos Fase 2:**
18	Diseño gráfico y adaptación de imágenes
19	Lógica de niveles
20	Lógica de casillas
21	Diseño
22	Implementación
23	Evaluación
24	Reunión
25	**Fase 3 – Requisitos sobre Unidades y Edificios**
26	**Requisitos Fase 3:**
27	Diseño gráfico y adaptación de imágenes
28	Control y desplazamiento
29	Lógica de presentación en pantalla
30	Detección y tratamiento de colisiones
31	Diseño
32	Implementación
33	Evaluación
34	Reunión
35	**Fase 4 – Requisitos sobre Barra de control y órdenes**
36	**Requisitos Fase 4:**
37	Diseño gráfico y adaptación de imágenes
38	Mapa en miniatura
39	Implementación de órdenes
40	Presentación de información adicional
41	Diseño
42	Implementación
43	Evaluación
44	Reunión
45	**Fase 5 – Otros requisitos**
46	**Requisitos Fase 5:**
47	Adaptación de sonidos y música
48	Implementación de sonidos y música
49	Finalización de menús: opciones
50	Diseño
51	Implementación
52	Evaluación
53	Reunión
54	**Evaluación**
55	Evaluación y pruebas finales
56	Reunión final

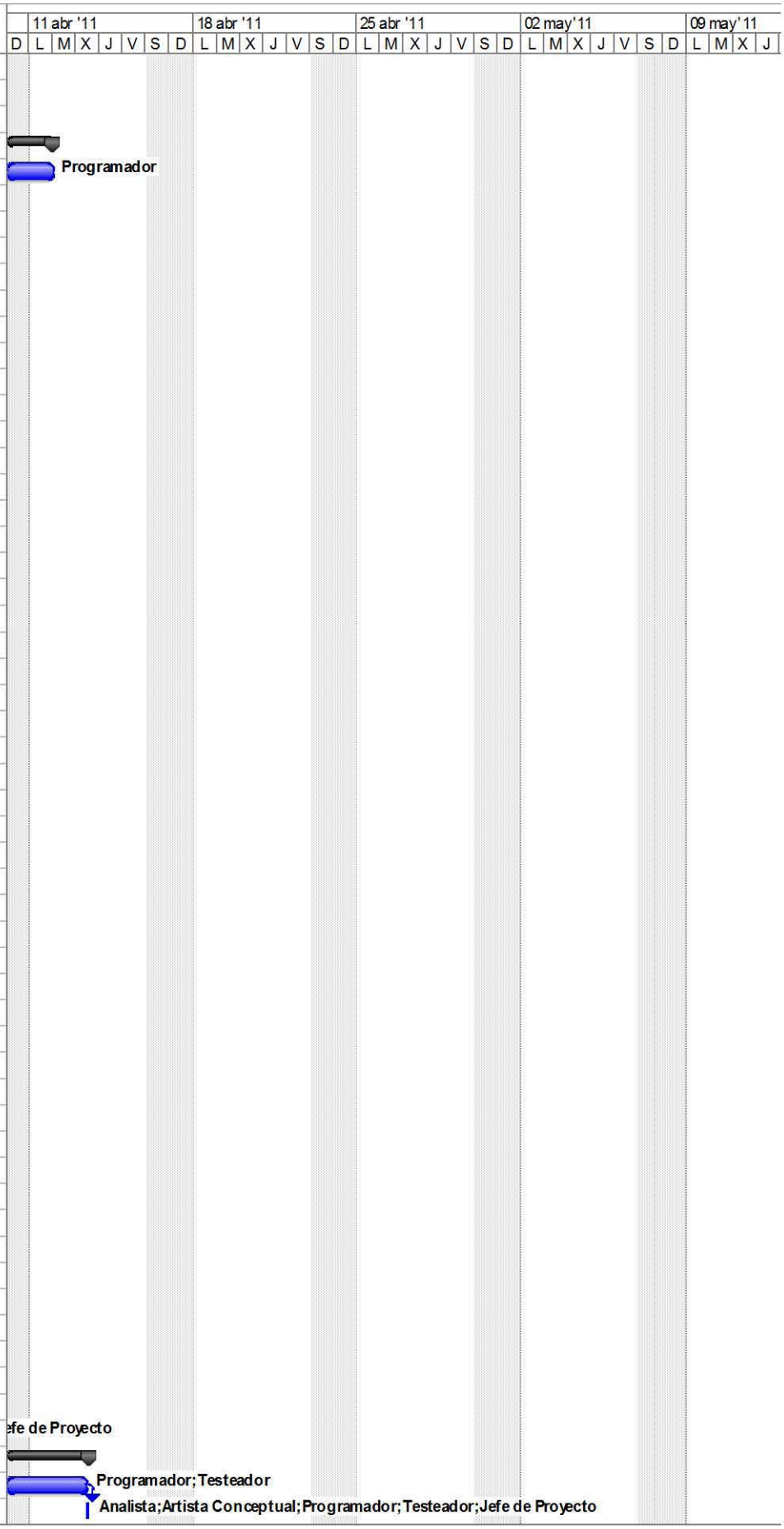

Figura 105. Diagrama de Gantt inicial – Planificación 8

A.3.2. Planificación final

Una de las actividades realizadas durante el cierre del proyecto es la actualización final de la planificación, pues los cambios en la misma son parte natural de toda actividad a largo plazo y pueden ser estudiados para estimar futuros desarrollos. Esta sección recoge el diagrama de Gantt final del proyecto anotando las razones de las desviaciones experimentadas.

Formación preparatoria sobre la tecnología de desarrollo	**20 días**
Estudio documentación Visual Studio.NET	15 días
Estudio documentación XNA	20 días
Desarrollo de documentación y memoria	**200 días**
Fase 1 – Requisitos Generales	**29 días**
Requisitos Fase 1:	5 días
Aspecto y dimensiones de ventana	3 días
Utilización básica de imágenes	2 días
Realización de menús interconectados	5 días
Interrupción y salida del juego	1 días
Diseño	7 días
Implementación	13 días
Evaluación	4 días
Fase 2 – Requisitos sobre el Mapa	**31 días**
Requisitos Fase 2:	6 días
Diseño gráfico y adaptación de imágenes	6 días
Lógica de niveles	3 días
Lógica de casillas	2 días
Diseño	7 días
Implementación	14 días

Tabla 168: Tareas finales del proyecto (parte 1ª)

Evaluación	4 días
Fase 3 – Requisitos sobre Unidades y Edificios	**36 días**
Requisitos Fase 3:	8 días
Diseño gráfico y adaptación de imágenes	8 días
Control y desplazamiento	7 días
Lógica de presentación en pantalla	2 días
Detección y tratamiento de colisiones	4 días
Diseño	6 días
Implementación	17 días
Evaluación	5 días
Fase 4 – Requisitos sobre Barra de control y órdenes	**41 días**
Requisitos Fase 4:	8 días
Diseño gráfico y adaptación de imágenes	8 días
Mapa en miniatura	2 días
Implementación de órdenes	6 días
Presentación de información adicional	2 días
Diseño	7 días
Implementación	20 días
Evaluación	6 días
Fase 5 – Otros requisitos	**24 días**
Requisitos Fase 5:	5 días
Adaptación de sonidos y música	5 días
Implementación de sonidos y música	4 días
Finalización de menús: opciones	1 días
Diseño	6 días
Implementación	10 días

Tabla 169: Tareas finales del proyecto (parte 2ª)

Evaluación	3 días
Evaluación y pruebas finales	**6 días**

Tabla 170: Tareas finales del proyecto (parte 3ª)

Durante el cierre del proyecto se observa cómo ciertas tareas han aumentado su extensión en tiempo mientras que otras han resultado más cortas de lo esperado. Las causas que se apuntan como principales son dos: en primer lugar, ninguna planificación es exacta; por otro lado, otro factor influyente ha sido la inexperiencia del alumno en este tipo de desarrollos. Analizando en profundidad aspectos concretos se tiene que:

- El tiempo planificado para la formación preparatoria ha sido menor al esperado porque Visual Studio .NET era conocido previamente y porque las operaciones con XNA en el entorno de las dos dimensiones son sencillas.

- La creación de la documentación asociada al proyecto se ha visto retrasada por encontrarse el producto en un estado previo a la finalización.

- La creación de la presentación del producto y los ensayos asociados se han recogido en una nueva tarea. Esto se debe a la moderada cantidad de tiempo invertido en este propósito.

- Con carácter general se puede afirmar que se ha disminuido el tiempo dedicado a la planificación de requisitos de cada fase. La causa de este suceso es el conocimiento del juego StarCraft original que tiene el alumno, dado que facilita la especificación rápida y precisa de las necesidades de usuario y de software del juego a desarrollar.

- La mayoría de fases de diseño, implementación y evaluación han requerido más tiempo para ser completadas. Esto ha ocurrido, en buena parte, por la inexperiencia del alumno a la hora de planificar un desarrollo de ocio digital. Sin embargo, otro factor que ha tenido mucho peso es la ingente cantidad de detalles que es posible incluir: para cada elemento (p.e., una unidad), siempre es posible añadir funcionalidades o retocar las existentes (p.e., mayor detalle gráfico, más sonidos de respuesta, fragmentos de vídeo, nuevas órdenes…).

- Otro retraso importante ha sido el acusado por el trabajo con recursos gráficos y de sonido. Estos elementos, dado que son heredados de la creación StarCraft original, han requerido un laborioso proceso de ingeniería inversa para ser utilizables en el proyecto actual. Sin embargo, este trabajo es menos costoso en tiempo que la alternativa de desarrollar todo el contenido desde cero.

Id	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos
1	**Formación perparatoria sobre la tecnología de desarrollo**	**20 días**	**07/09/10**	**04/10/10**	**PC**
2	Estudio documentación Visual Studio.NET	15 días	07/09/10	27/09/10	Programador
3	Estudio documentación XNA	20 días	07/09/10	04/10/10	Programador
4	**Desarrollo de documentación y memoria**	**200 días**	**07/09/10**	**13/06/11**	**PC**
5	Desarrollo de documentación y memoria	190 días	07/09/10	30/05/11	Programador
6	Desarrollo de presentación de proyecto	10 días	31/05/11	13/06/11	Analista
7	**Fase 1 – Requisitos Generales**	**29,25 días**	**05/10/10**	**15/11/10**	**PC**
8	**Requisitos Fase 1:**	**5 días**	**05/10/10**	**11/10/10**	
9	Aspecto ydimensiones de ventana	3 días	05/10/10	07/10/10	Artista Conceptual;Adobe Photoshop CS3
10	Utilización básica de imágenes	2 días	05/10/10	06/10/10	Analista
11	Realización de menús interconectados	5 días	05/10/10	11/10/10	Analista
12	Interrupción y salida del juego	1 día	05/10/10	05/10/10	Analista
13	Diseño	7 días	12/10/10	20/10/10	Analista
14	Implementación	13 días	21/10/10	08/11/10	Programador
15	Evaluación	4 días	09/11/10	12/11/10	Testeador
16	Reunión	2 horas	15/11/10	15/11/10	Analista;Jefe de Proyecto
17	**Fase 2 – Requisitos sobre el Mapa**	**31,25 días**	**15/11/10**	**28/12/10**	**PC**
18	**Requisitos Fase 2:**	**6 días**	**15/11/10**	**23/11/10**	
19	Diseño gráfico y adaptación de imágenes	6 días	15/11/10	23/11/10	Artista Conceptual;Adobe Photoshop CS3
20	Lógica de niveles	3 días	15/11/10	18/11/10	Analista
21	Lógica de casillas	2 días	15/11/10	17/11/10	Analista
22	Diseño	7 días	23/11/10	02/12/10	Analista
23	Implementación	14 días	02/12/10	22/12/10	Programador
24	Evaluación	4 días	22/12/10	28/12/10	Testeador
25	Reunión	2 horas	28/12/10	28/12/10	Analista;Jefe de Proyecto
26	**Fase 3 – Requisitos sobre Unidades y Edificios**	**36,25 días**	**28/12/10**	**16/02/11**	**PC**
27	**Requisitos Fase 3:**	**8 días**	**28/12/10**	**07/01/11**	
28	Diseño gráfico y adaptación de imágenes	8 días	28/12/10	07/01/11	Artista Conceptual;Adobe Photoshop CS3
29	Control y desplazamiento	7 días	28/12/10	06/01/11	Analista;Programador
30	Lógica de presentación en pantalla	2 días	28/12/10	30/12/10	Analista;Programador
31	Detección y tratamiento de colisiones	4 días	28/12/10	03/01/11	Analista
32	Diseño	6 días	07/01/11	17/01/11	Analista
33	Implementación	17 días	17/01/11	09/02/11	Programador
34	Evaluación	5 días	09/02/11	16/02/11	Testeador
35	Reunión	2 horas	16/02/11	16/02/11	Analista;Jefe de Proyecto
36	**Fase 4 – Requisitos sobre Barra de control yórdenes**	**41,25 días**	**16/02/11**	**14/04/11**	**PC**
37	**Requisitos Fase 4:**	**8 días**	**16/02/11**	**28/02/11**	
38	Diseño gráfico y adaptación de imágenes	8 días	16/02/11	28/02/11	Artista Conceptual;Adobe Photoshop CS3
39	Mapa en miniatura	2 días	16/02/11	18/02/11	Analista;Programador
40	Implementación de órdenes	6 días	16/02/11	24/02/11	Analista;Programador
41	Presentación de información adicional	2 días	16/02/11	18/02/11	Analista
42	Diseño	7 días	28/02/11	09/03/11	Analista
43	Implementación	20 días	09/03/11	06/04/11	Programador
44	Evaluación	6 días	06/04/11	14/04/11	Testeador
45	Reunión	2 horas	14/04/11	14/04/11	Analista;Jefe de Proyecto
46	**Fase 5 – Otros requisitos**	**24,25 días**	**15/04/11**	**19/05/11**	**PC**
47	**Requisitos Fase 5:**	**5 días**	**15/04/11**	**21/04/11**	
48	Adaptación de sonidos ymúsica	5 días	15/04/11	21/04/11	Artista Conceptual
49	Implementación de sonidos ymúsica	4 días	15/04/11	20/04/11	Analista;Programador
50	Finalización de menús: opciones	1 día	15/04/11	15/04/11	Analista;Programador
51	Diseño	6 días	22/04/11	29/04/11	Analista
52	Implementación	10 días	02/05/11	13/05/11	Programador
53	Evaluación	3 días	16/05/11	18/05/11	Testeador
54	Reunión	2 horas	19/05/11	19/05/11	Analista;Jefe de Proyecto
55	**Evaluación**	**6,25 días**	**19/05/11**	**27/05/11**	**Programador;Testeador**
56	Evaluación ypruebas finales	6 días	19/05/11	27/05/11	Programador;Testeador
57	Reunión final	2 horas	27/05/11	27/05/11	Analista;Artista Conceptual;Programador;Testeador;Jefe de Proyecto

Figura 106. Diagrama de Gantt final – Tareas

Id	Nombre de tarea
1	**Formación preparatoria sobre la tecnología de desarrollo**
2	Estudio documentación Visual Studio.NET
3	Estudio documentación XNA
4	**Desarrollo de documentación y memoria**
5	Desarrollo de documentación y memoria
6	Desarrollo de presentación de proyecto
7	**Fase 1 – Requisitos Generales**
8	**Requisitos Fase 1:**
9	Aspecto y dimensiones de ventana
10	Utilización básica de imágenes
11	Realización de menús interconectados
12	Interrupción y salida del juego
13	Diseño
14	Implementación
15	Evaluación
16	Reunión
17	**Fase 2 – Requisitos sobre el Mapa**
18	**Requisitos Fase 2:**
19	Diseño gráfico y adaptación de imágenes
20	Lógica de niveles
21	Lógica de casillas
22	Diseño
23	Implementación
24	Evaluación
25	Reunión
26	**Fase 3 – Requisitos sobre Unidades y Edificios**
27	**Requisitos Fase 3:**
28	Diseño gráfico y adaptación de imágenes
29	Control y desplazamiento
30	Lógica de presentación en pantalla
31	Detección y tratamiento de colisiones
32	Diseño
33	Implementación
34	Evaluación
35	Reunión
36	**Fase 4 – Requisitos sobre Barra de control y órdenes**
37	**Requisitos Fase 4:**
38	Diseño gráfico y adaptación de imágenes
39	Mapa en miniatura
40	Implementación de órdenes
41	Presentación de información adicional
42	Diseño
43	Implementación
44	Evaluación
45	Reunión
46	**Fase 5 – Otros requisitos**
47	**Requisitos Fase 5:**
48	Adaptación de sonidos y música
49	Implementación de sonidos y música
50	Finalización de menús: opciones
51	Diseño
52	Implementación
53	Evaluación
54	Reunión
55	**Evaluación**
56	Evaluación y pruebas finales
57	Reunión final

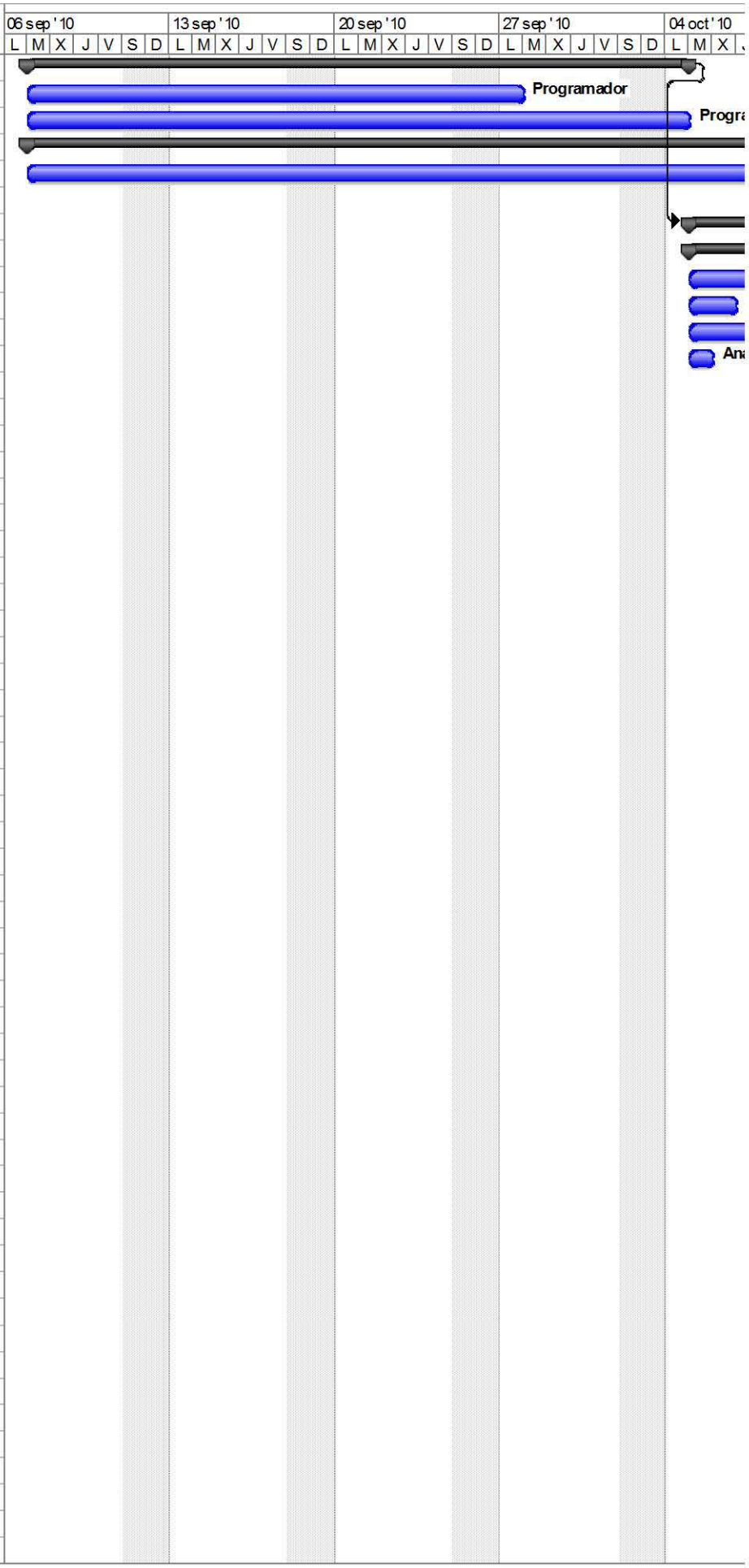

Figura 107. Diagrama de Gantt final – Planificación 1

Id	Nombre de tarea
1	**Formación preparatoria sobre la tecnología de desarrollo**
2	Estudio documentación Visual Studio.NET
3	Estudio documentación XNA
4	**Desarrollo de documentación y memoria**
5	Desarrollo de documentación y memoria
6	Desarrollo de presentación de proyecto
7	**Fase 1 – Requisitos Generales**
8	**Requisitos Fase 1:**
9	Aspecto y dimensiones de ventana
10	Utilización básica de imágenes
11	Realización de menús interconectados
12	Interrupción y salida del juego
13	Diseño
14	Implementación
15	Evaluación
16	Reunión
17	**Fase 2 – Requisitos sobre el Mapa**
18	**Requisitos Fase 2:**
19	Diseño gráfico y adaptación de imágenes
20	Lógica de niveles
21	Lógica de casillas
22	Diseño
23	Implementación
24	Evaluación
25	Reunión
26	**Fase 3 – Requisitos sobre Unidades y Edificios**
27	**Requisitos Fase 3:**
28	Diseño gráfico y adaptación de imágenes
29	Control y desplazamiento
30	Lógica de presentación en pantalla
31	Detección y tratamiento de colisiones
32	Diseño
33	Implementación
34	Evaluación
35	Reunión
36	**Fase 4 – Requisitos sobre Barra de control y órdenes**
37	**Requisitos Fase 4:**
38	Diseño gráfico y adaptación de imágenes
39	Mapa en miniatura
40	Implementación de órdenes
41	Presentación de información adicional
42	Diseño
43	Implementación
44	Evaluación
45	Reunión
46	**Fase 5 – Otros requisitos**
47	**Requisitos Fase 5:**
48	Adaptación de sonidos y música
49	Implementación de sonidos y música
50	Finalización de menús: opciones
51	Diseño
52	Implementación
53	Evaluación
54	Reunión
55	**Evaluación**
56	Evaluación y pruebas finales
57	Reunión final

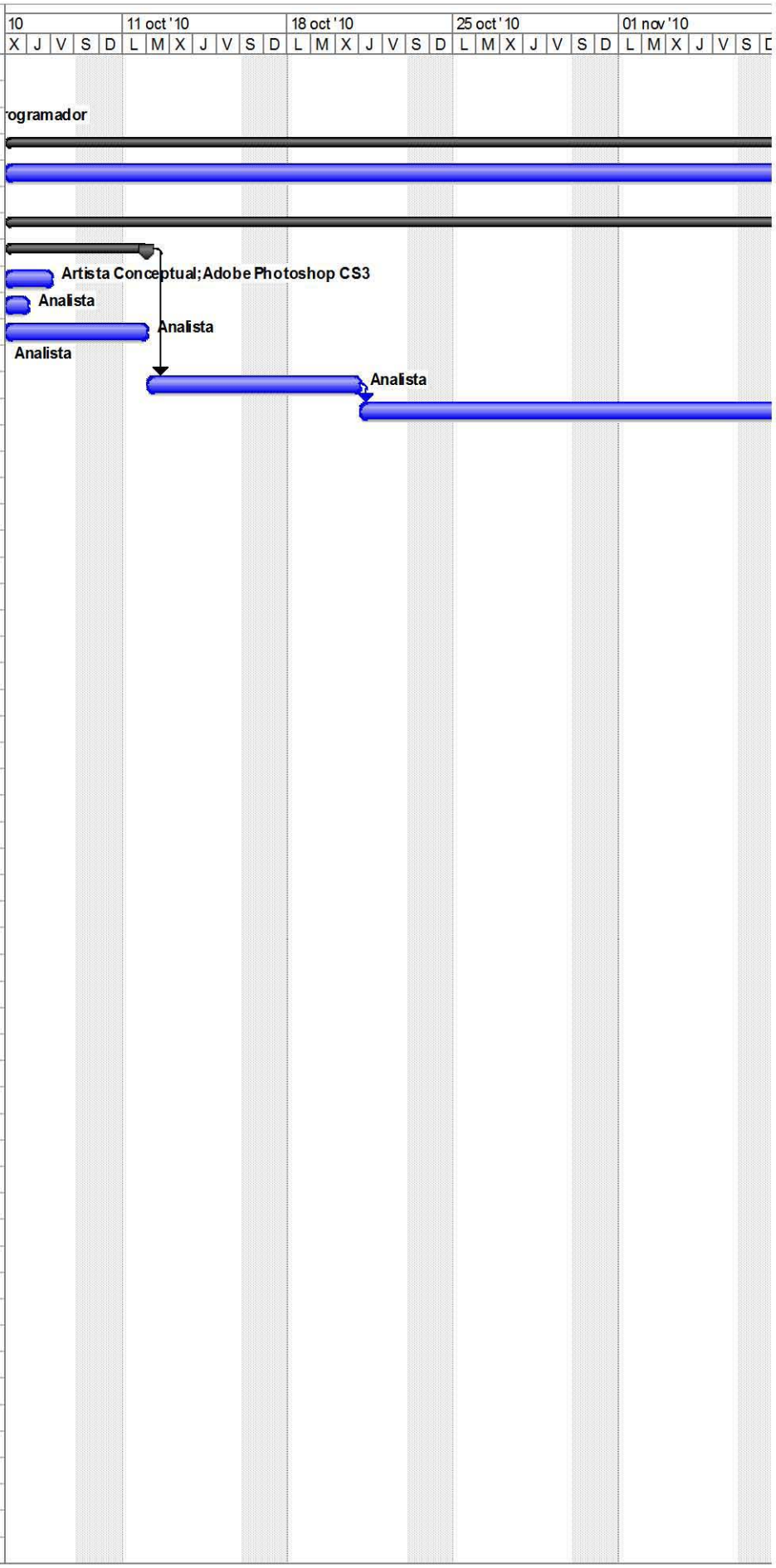

Figura 108. Diagrama de Gantt final – Planificación 2

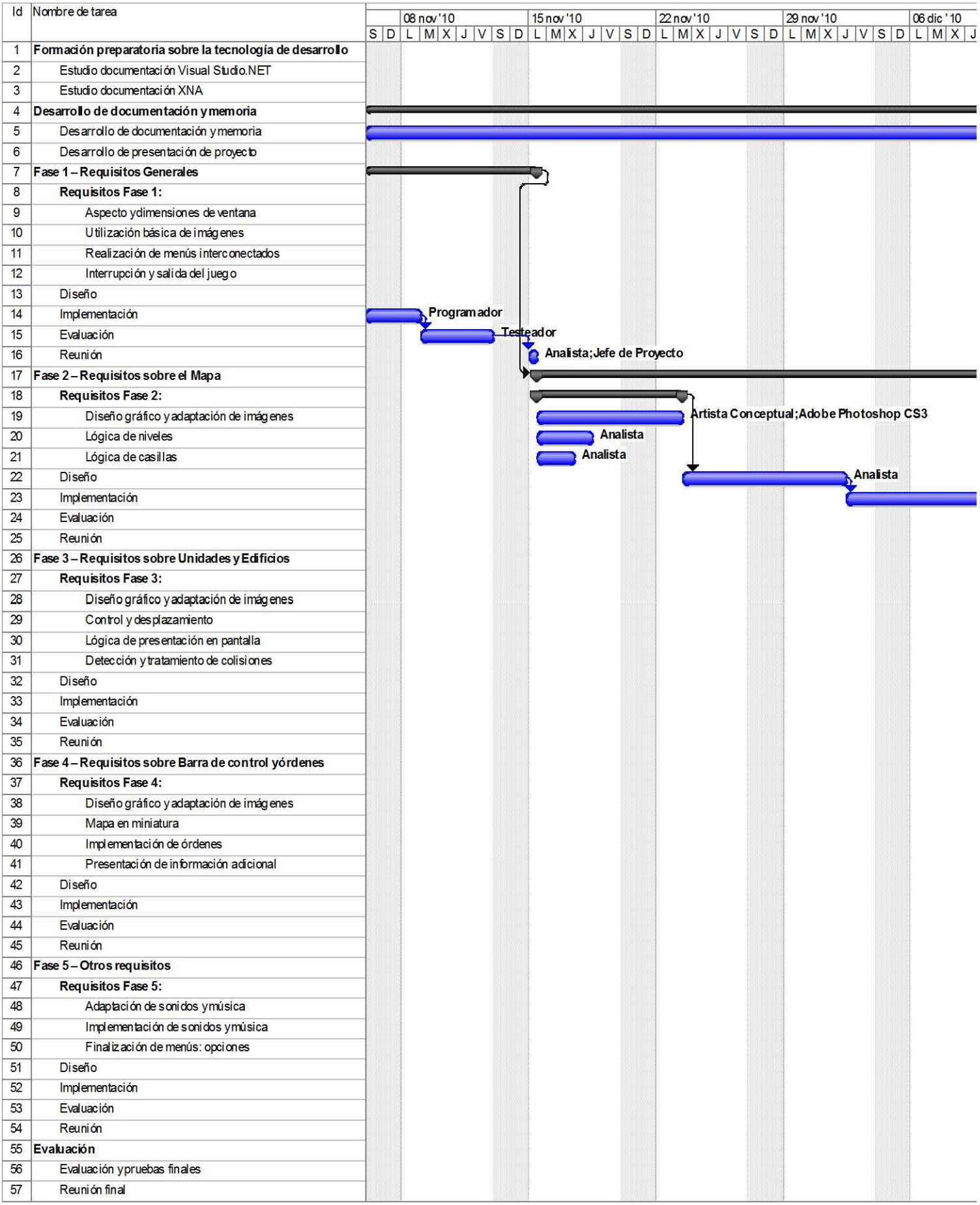

Figura 109. Diagrama de Gantt final – Planificación 3

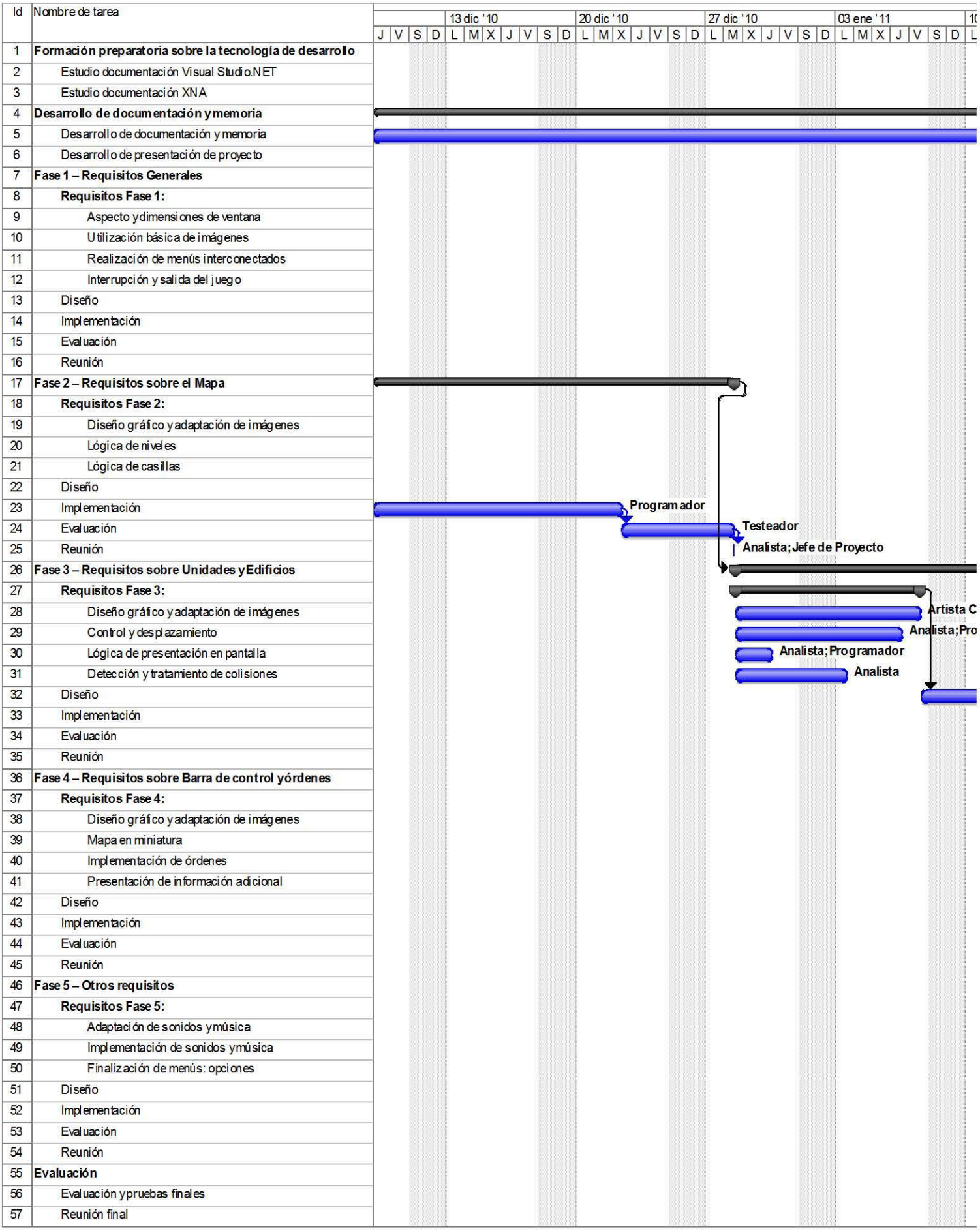

Figura 110. Diagrama de Gantt final – Planificación 4

Id	Nombre de tarea
1	**Formación preparatoria sobre la tecnología de desarrollo**
2	Estudio documentación Visual Studio.NET
3	Estudio documentación XNA
4	**Desarrollo de documentación y memoria**
5	Desarrollo de documentación y memoria
6	Desarrollo de presentación de proyecto
7	**Fase 1 – Requisitos Generales**
8	**Requisitos Fase 1:**
9	Aspecto y dimensiones de ventana
10	Utilización básica de imágenes
11	Realización de menús interconectados
12	Interrupción y salida del juego
13	Diseño
14	Implementación
15	Evaluación
16	Reunión
17	**Fase 2 – Requisitos sobre el Mapa**
18	**Requisitos Fase 2:**
19	Diseño gráfico y adaptación de imágenes
20	Lógica de niveles
21	Lógica de casillas
22	Diseño
23	Implementación
24	Evaluación
25	Reunión
26	**Fase 3 – Requisitos sobre Unidades y Edificios**
27	**Requisitos Fase 3:**
28	Diseño gráfico y adaptación de imágenes
29	Control y desplazamiento
30	Lógica de presentación en pantalla
31	Detección y tratamiento de colisiones
32	Diseño
33	Implementación
34	Evaluación
35	Reunión
36	**Fase 4 – Requisitos sobre Barra de control y órdenes**
37	**Requisitos Fase 4:**
38	Diseño gráfico y adaptación de imágenes
39	Mapa en miniatura
40	Implementación de órdenes
41	Presentación de información adicional
42	Diseño
43	Implementación
44	Evaluación
45	Reunión
46	**Fase 5 – Otros requisitos**
47	**Requisitos Fase 5:**
48	Adaptación de sonidos y música
49	Implementación de sonidos y música
50	Finalización de menús: opciones
51	Diseño
52	Implementación
53	Evaluación
54	Reunión
55	**Evaluación**
56	Evaluación y pruebas finales
57	Reunión final

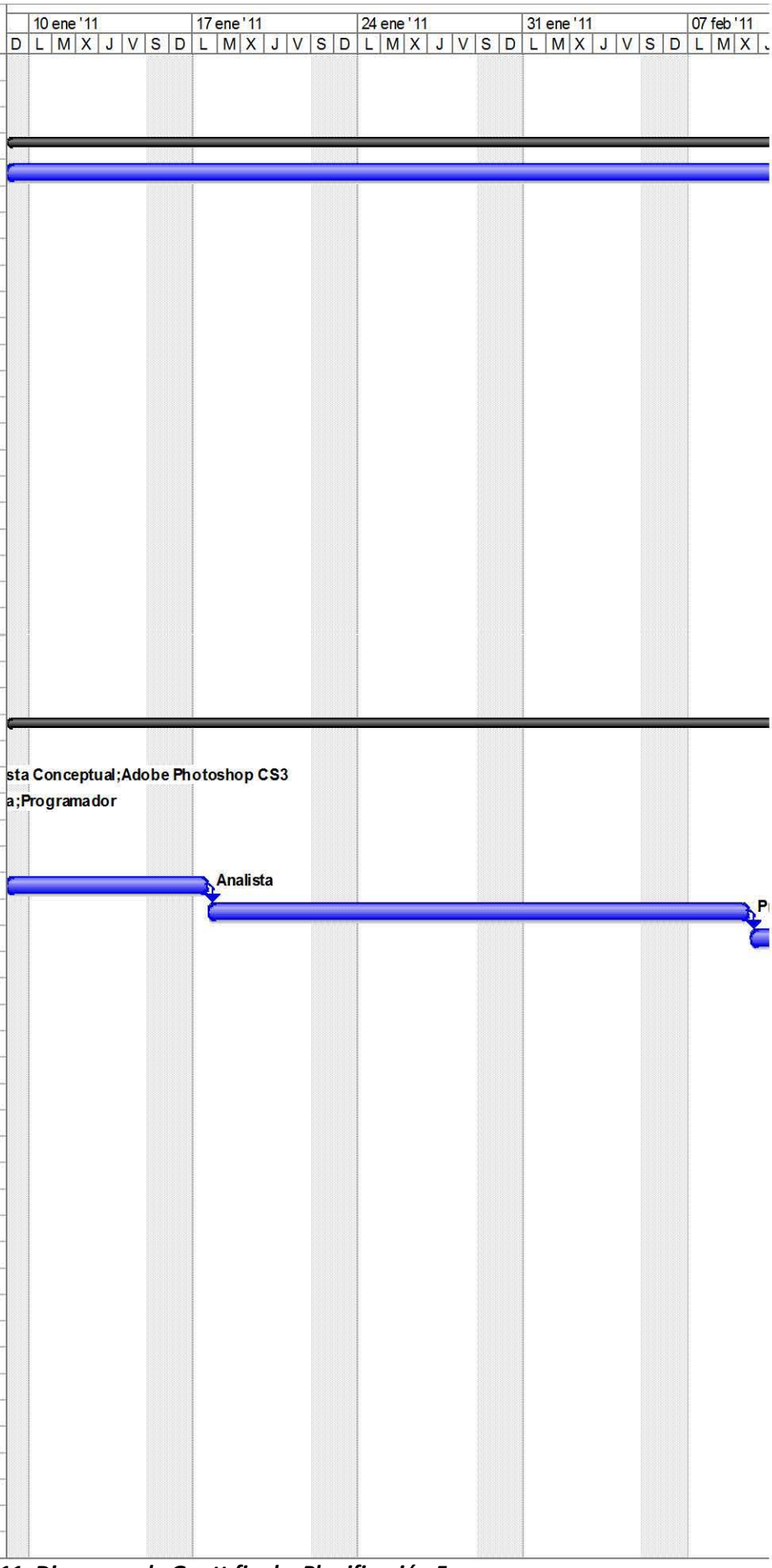

Figura 111. Diagrama de Gantt final – Planificación 5

Id	Nombre de tarea
1	**Formación preparatoria sobre la tecnología de desarrollo**
2	Estudio documentación Visual Studio.NET
3	Estudio documentación XNA
4	**Desarrollo de documentación y memoria**
5	Desarrollo de documentación y memoria
6	Desarrollo de presentación de proyecto
7	**Fase 1 – Requisitos Generales**
8	**Requisitos Fase 1:**
9	Aspecto y dimensiones de ventana
10	Utilización básica de imágenes
11	Realización de menús interconectados
12	Interrupción y salida del juego
13	Diseño
14	Implementación
15	Evaluación
16	Reunión
17	**Fase 2 – Requisitos sobre el Mapa**
18	**Requisitos Fase 2:**
19	Diseño gráfico y adaptación de imágenes
20	Lógica de niveles
21	Lógica de casillas
22	Diseño
23	Implementación
24	Evaluación
25	Reunión
26	**Fase 3 – Requisitos sobre Unidades y Edificios**
27	**Requisitos Fase 3:**
28	Diseño gráfico y adaptación de imágenes
29	Control y desplazamiento
30	Lógica de presentación en pantalla
31	Detección y tratamiento de colisiones
32	Diseño
33	Implementación
34	Evaluación
35	Reunión
36	**Fase 4 – Requisitos sobre Barra de control y órdenes**
37	**Requisitos Fase 4:**
38	Diseño gráfico y adaptación de imágenes
39	Mapa en miniatura
40	Implementación de órdenes
41	Presentación de información adicional
42	Diseño
43	Implementación
44	Evaluación
45	Reunión
46	**Fase 5 – Otros requisitos**
47	**Requisitos Fase 5:**
48	Adaptación de sonidos y música
49	Implementación de sonidos y música
50	Finalización de menús: opciones
51	Diseño
52	Implementación
53	Evaluación
54	Reunión
55	**Evaluación**
56	Evaluación y pruebas finales
57	Reunión final

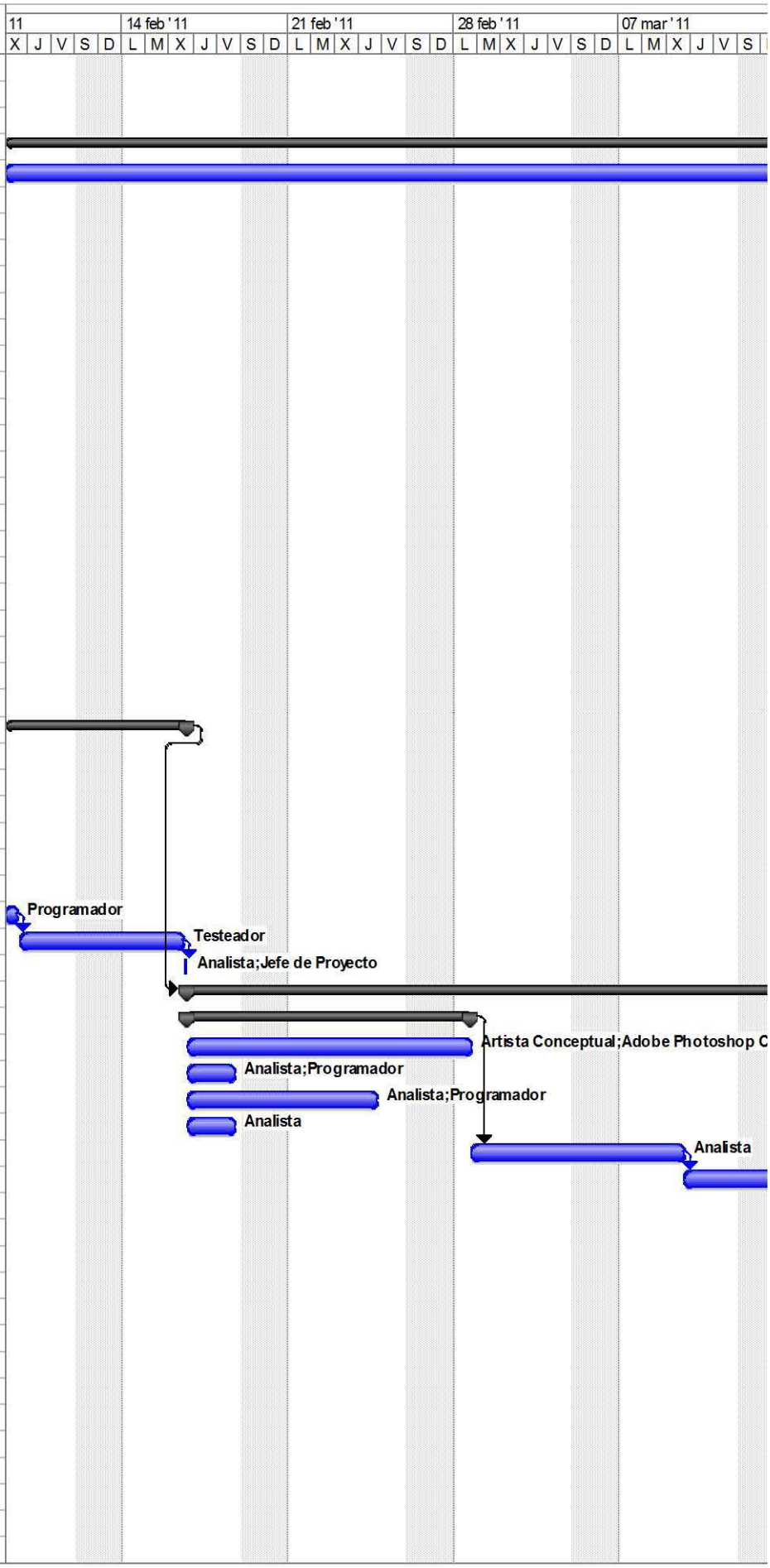

Figura 112. Diagrama de Gantt final – Planificación 6

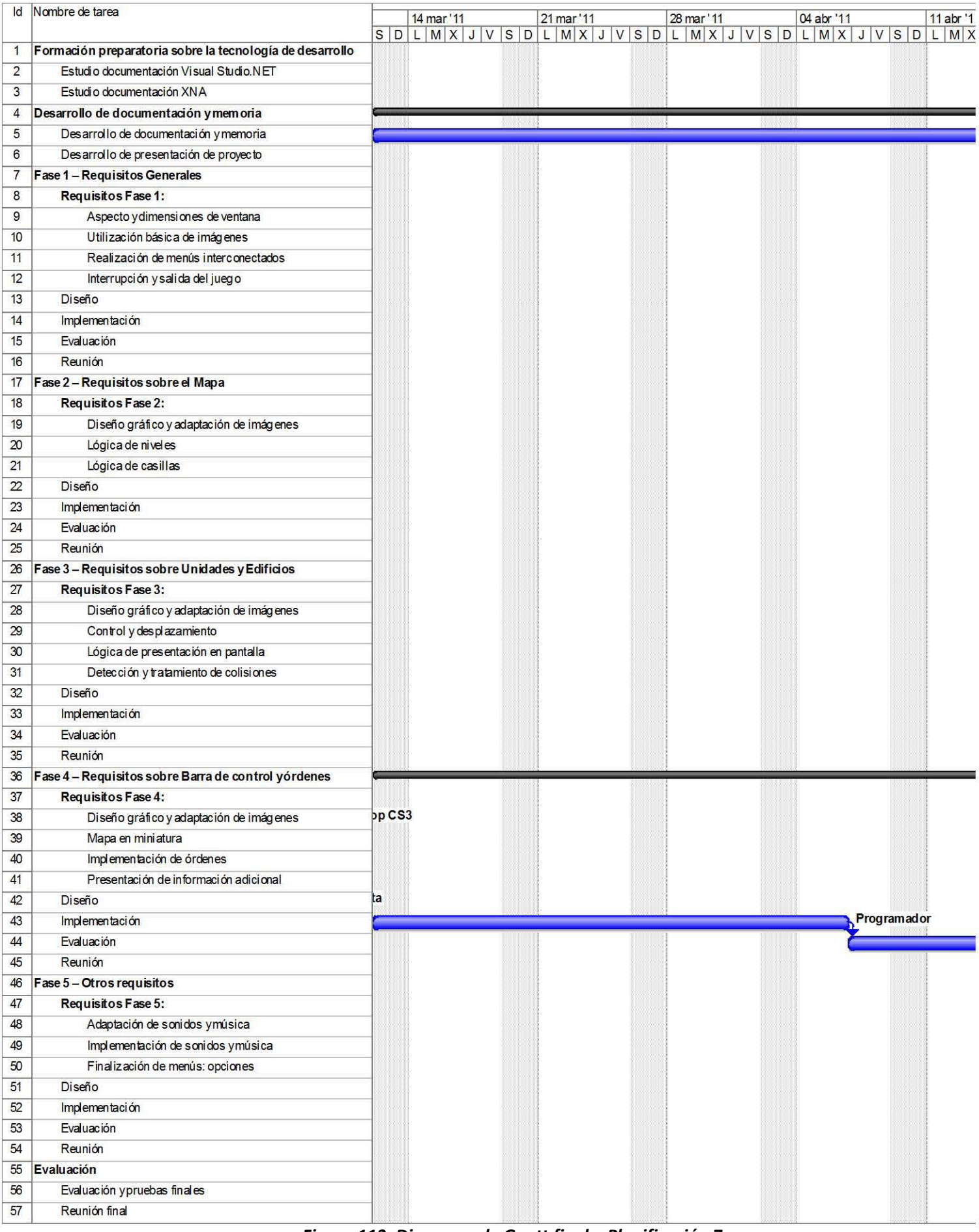

Figura 113. Diagrama de Gantt final – Planificación 7

Figura 114. Diagrama de Gantt final – Planificación 8

Id	Nombre de tarea
1	**Formación preparatoria sobre la tecnología de desarrollo**
2	Estudio documentación Visual Studio.NET
3	Estudio documentación XNA
4	**Desarrollo de documentación y memoria**
5	Desarrollo de documentación y memoria
6	Desarrollo de presentación de proyecto
7	**Fase 1 – Requisitos Generales**
8	**Requisitos Fase 1:**
9	Aspecto y dimensiones de ventana
10	Utilización básica de imágenes
11	Realización de menús interconectados
12	Interrupción y salida del juego
13	Diseño
14	Implementación
15	Evaluación
16	Reunión
17	**Fase 2 – Requisitos sobre el Mapa**
18	**Requisitos Fase 2:**
19	Diseño gráfico y adaptación de imágenes
20	Lógica de niveles
21	Lógica de casillas
22	Diseño
23	Implementación
24	Evaluación
25	Reunión
26	**Fase 3 – Requisitos sobre Unidades y Edificios**
27	**Requisitos Fase 3:**
28	Diseño gráfico y adaptación de imágenes
29	Control y desplazamiento
30	Lógica de presentación en pantalla
31	Detección y tratamiento de colisiones
32	Diseño
33	Implementación
34	Evaluación
35	Reunión
36	**Fase 4 – Requisitos sobre Barra de control y órdenes**
37	**Requisitos Fase 4:**
38	Diseño gráfico y adaptación de imágenes
39	Mapa en miniatura
40	Implementación de órdenes
41	Presentación de información adicional
42	Diseño
43	Implementación
44	Evaluación
45	Reunión
46	**Fase 5 – Otros requisitos**
47	**Requisitos Fase 5:**
48	Adaptación de sonidos y música
49	Implementación de sonidos y música
50	Finalización de menús: opciones
51	Diseño
52	Implementación
53	Evaluación
54	Reunión
55	**Evaluación**
56	Evaluación y pruebas finales
57	Reunión final

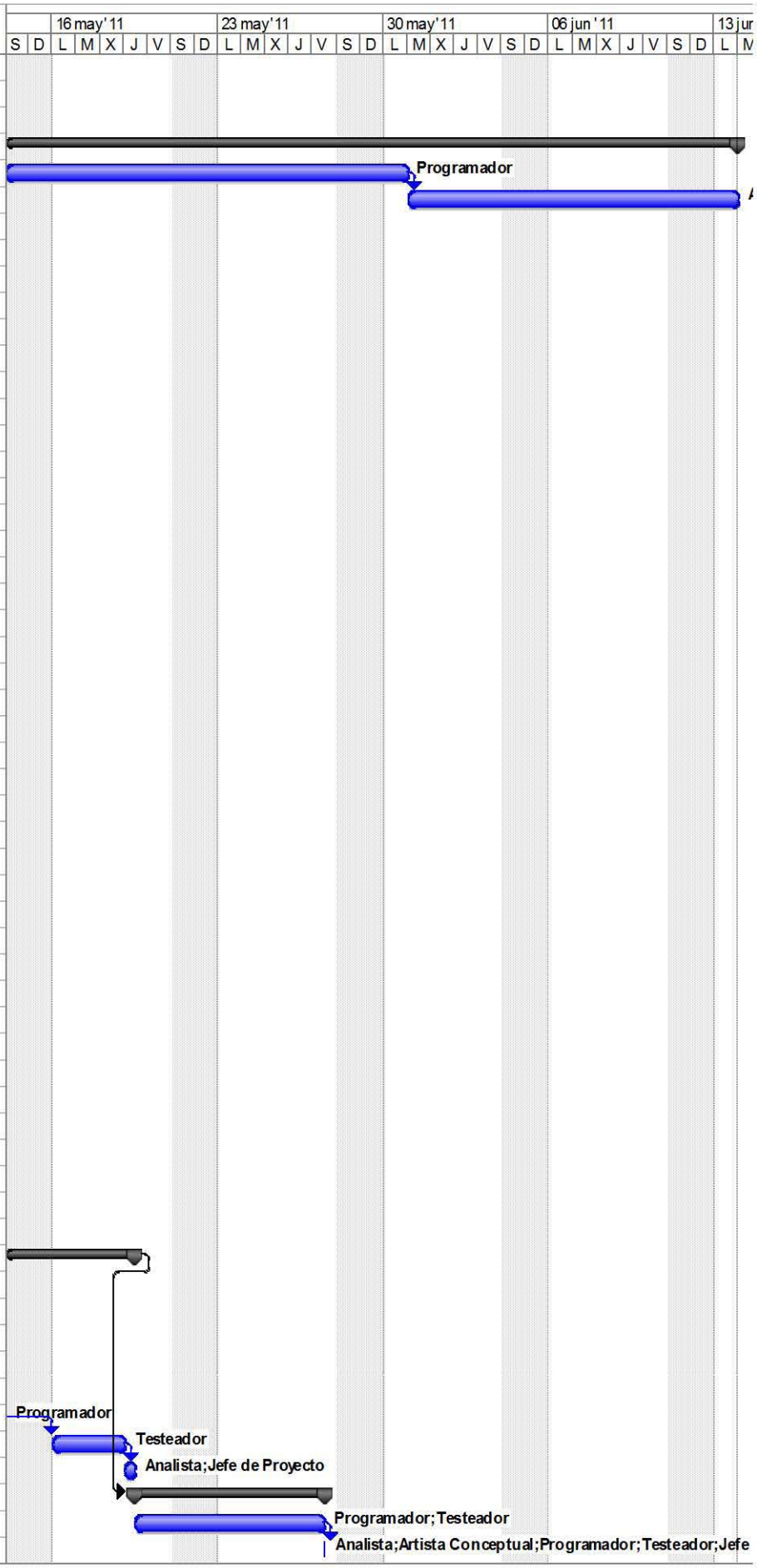

Figura 115. Diagrama de Gantt final – Planificación 9

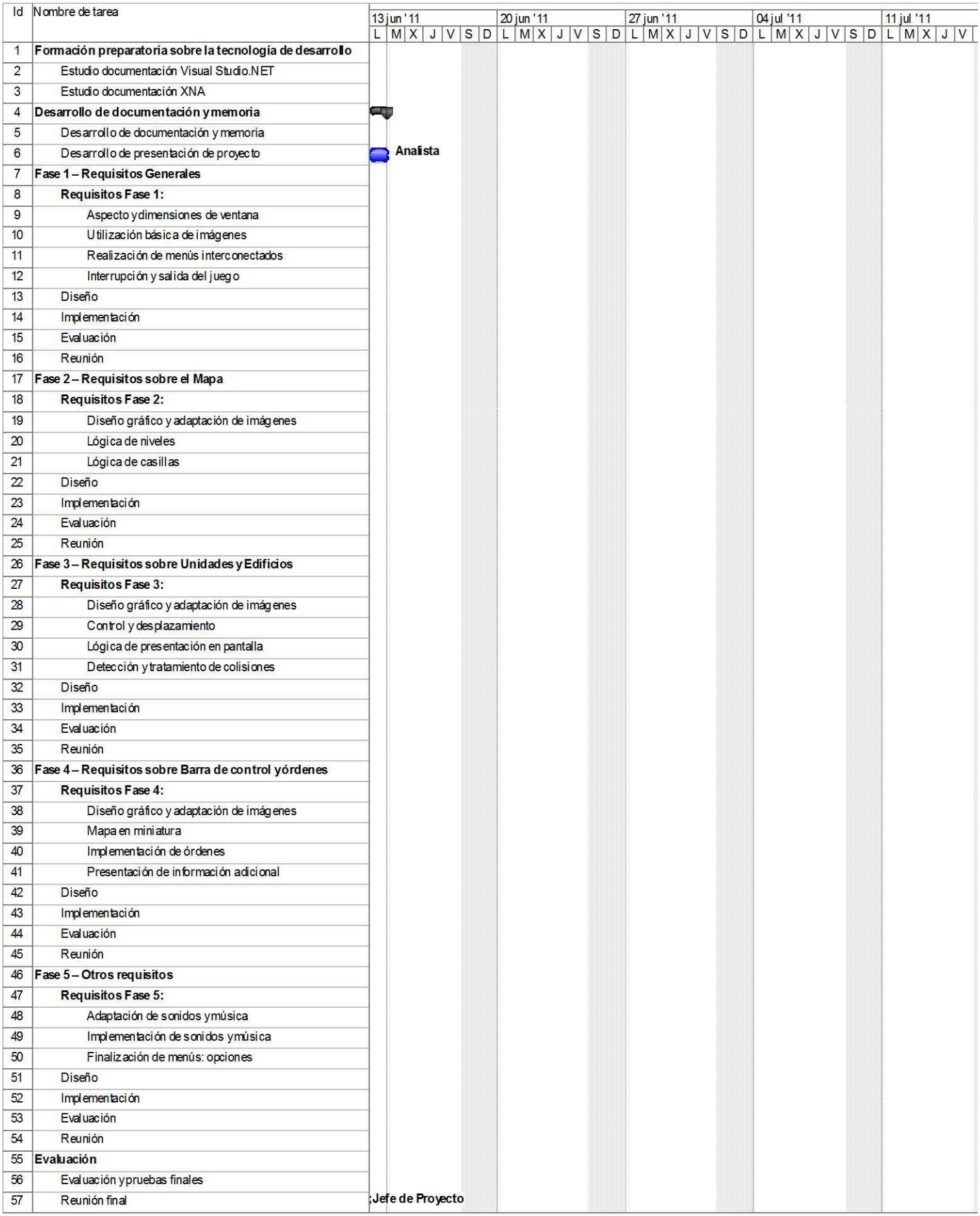

Figura 116. Diagrama de Gantt final – Planificación 10

A.4. Presupuesto del proyecto

Cualquier presupuesto es un factor clave para el estudio de viabilidad de un proyecto. Esta previsión económica se realiza en base a unas tareas, unos recursos y unas fechas determinadas. Cualquier variación de los datos de origen puede suponer una modificación importante del coste inicialmente previsto.

Por tanto, se ha creído óptimo recoger en esta sección los costes asociados al proyecto. Primero se hará de una manera estimada, previa a la realización, para luego, al terminar las actividades, ofrecer un presupuesto más preciso.

A.4.1. Presupuesto inicial

Dar por válida la idea inicial del proyecto supone una importante fase de análisis. Fruto de este trabajo se extrae un primer conjunto de requisitos. Estas necesidades reciben su asignación de recursos y son presupuestadas. Si todo va bien, la oferta se realizará y aceptará, y el proyecto dará comienzo.

La siguiente tabla muestra los recursos asociados al proyecto actual:

Recurso Humano	**Trabajo**
Jefe de proyecto	12 horas
Analista	716 horas
Artista conceptual	130 horas
Programador	2388 horas
Testeador	228 horas
Recurso Material	**Unidades**
PC con Windows XP	1
Suscripción XNA Creators Club	1
Adobe Photoshop CS3	1

Tabla 171: Recursos asociados al proyecto - inicial

Cada uno de estos datos tiene un coste asociado. La valoración de cada uno de ellos se puede observar en la siguiente tabla:

Recurso Humano	**€/Hora (bruto)**	**Coste directo € (bruto)**
Jefe de proyecto	23,08	276,96
Analista	16,71	11.964,36
Artista conceptual	9,31	1210,30
Programador	10,05	23.999,40
Testeador	10,02	2.284,56
Recurso Material	**€/Unidad**	**Coste directo € (bruto)**
PC con Windows XP	900,00	900,00
Suscripción XNA Creators Club	67,70	67,70
Adobe Photoshop CS3	1001,82	1001,82

Tabla 172: Coste asociado al proyecto - inicial

De lo expuesto se deduce que el coste directamente imputable al proyecto se estima en 41.705,10 €. La duración aproximada es de 157 días.

A.4.2. Presupuesto final

Al realizar el cierre del proyecto se realiza de nuevo la estimación presupuestaria asociada al mismo. Esto es doblemente beneficioso: por un lado, se tiene una valoración muy precisa del coste del proyecto; por otro lado, sirve para valorar el factor de corrección necesario para futuras estimaciones económicas.

La siguiente tabla muestra la dedicación de recursos que ha sido necesaria en el proyecto actual:

Recurso Humano	**Trabajo**
Jefe de proyecto	12 horas
Analista	684 horas
Artista conceptual	242 horas
Programador	2668 horas
Testeador	276 horas
Recurso Material	**Unidades**
PC con Windows XP	1
Suscripción XNA Creators Club	1
Adobe Photoshop CS3	1

Tabla 173: Recursos asociados al proyecto - final

Cada uno de los recursos tiene un coste asociado. En la siguiente tabla se puede apreciar dicho gasto:

Recurso Humano	**€/Hora (bruto)**	**Coste directo € (bruto)**
Jefe de proyecto	23,08	276,96
Analista	16,71	11.429,64
Artista conceptual	9,31	2253,02
Programador	10,05	26.813,40
Testeador	10,02	2.765,52
Recurso Material	**€/Unidad**	**Coste directo € (bruto)**
PC con Windows XP	900,00	900,00
Suscripción XNA Creators Club	67,70	67,70
Adobe Photoshop CS3	1001,82	1001,82

Tabla 174: Coste asociado al proyecto - final

Al finalizar el proyecto se puede afirmar que el coste directamente imputable al proyecto ha sido de 45.508,06 €. La duración es de 200 días.

A.5. Publicación del juego

Para concluir el anexo se ha considerado conveniente realizar un análisis de la potencial publicación del videojuego desarrollado. El medio más adecuado, dadas las tecnologías utilizadas y los medios al alcance del alumno, sería App Hub[r166] de Microsoft. Mediante la suscripción a esta plataforma se hace posible la publicación del videojuego, apareciendo en los catálogos de aplicaciones de XBOX 360 y de Windows Phone.

Aunque la aplicación se mantiene activa durante un año, el proceso de publicación no es inmediato. El juego es revisado por la comunidad de desarrolladores y por empleados dedicados de Microsoft. Se tendrá en cuenta la información que aporte el desarrollador, los países donde se desee publicar, el contenido presente en los ejecutables del juego…

Sobre el tamaño del videojuego se impone también cierto control. Hasta 50 MB de espacio en disco duro, el videojuego podrá ser adquirido por 80 Puntos. A partir de 50 MB, el valor mínimo será de 240 Puntos. Estos Puntos serán la moneda de cambio en la plataforma de distribución; su funcionamiento es similar al de las tarjetas prepago de las compañías telefónicas.

Tras finalizar el proceso de revisión (contenido funcional, no ofensivo, no limitado por derechos de autor…), el juego pasará a formar parte del catálogo de aplicaciones. A partir de este momento se comenzará a generar beneficios. La normativa de Microsoft al respecto es:

- El beneficio generado será compartido entre Microsoft (10-30%) y el desarrollador (70-90%). Este impuesto promocional se publica el primer trimestre de cada año.

- Cuando el desarrollador alcance un límite mínimo de pago podrá solicitar que le sean ingresados los beneficios generados. Éstos serán cambiados a su divisa local y serán ingresados en su cuenta. El límite mínimo mencionado es de 150 dólares estadounidenses. Si no se alcanza este límite, la cantidad pertenecerá íntegramente a Microsoft.

Teniendo en cuenta lo analizado, se estima adecuado un valor inicial de 240 Puntos para el presente proyecto, cantidad que podría mantenerse durante el primer semestre. Para la segunda mitad del año se recomienda analizar la evolución del

beneficio generado, pudiendo elevar o disminuir el precio del juego según las expectativas generadas en el mercado.

De manera orientativa, a la fecha de escritura de este documento 4200 Puntos cuestan 60 euros, por lo que 240 Puntos son 3,43 euros y 80 Puntos son 1,14 euros. El tipo de cambio de euro a dólar estadounidense es de 1 a 1,4357. Por tanto, sería necesario vender entre 31 (240 Puntos) y 92 (80 Puntos) copias para obtener la cantidad mínima exigida para reclamar el ingreso de la parte de los beneficios correspondiente al autor.

Anexo B

Manual de usuario

B.1. Instalación

La instalación del videojuego StarCraft es muy sencilla: simplemente se deben completar los siguientes pasos en el orden descrito:

- Comprobar que están instalados los siguientes paquetes:
  - Microsoft .NET Framework Redistribuible.
  - Microsoft XNA Framework Redistribuible.
  - DirectX 8.1 o superior.
- Instalar la fuente "Starcraft Normal (TrueType)".
- Opcionalmente, copiar la carpeta del juego a la carpeta deseada.

Tras concluir estos pasos, la aplicación estará lista para ser jugada. Se deberá iniciar mediante el fichero "StarCraft.exe".

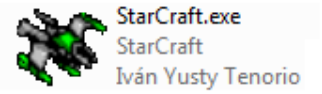

Figura 117. Ejecutable StarCraft

B.2. Menús

Para desplazarse por las opciones disponibles en los Menús se utilizarán las flechas de desplazamiento. Para aceptar se utilizará la tecla "Intro" o la tecla "Espacio". Para salir se utilizará la tecla "Esc".

Cuando sea ejecutado, StarCraft comenzará el proceso de carga de datos. Al concluir, lo primero que se verá es el Menú Principal de la aplicación.

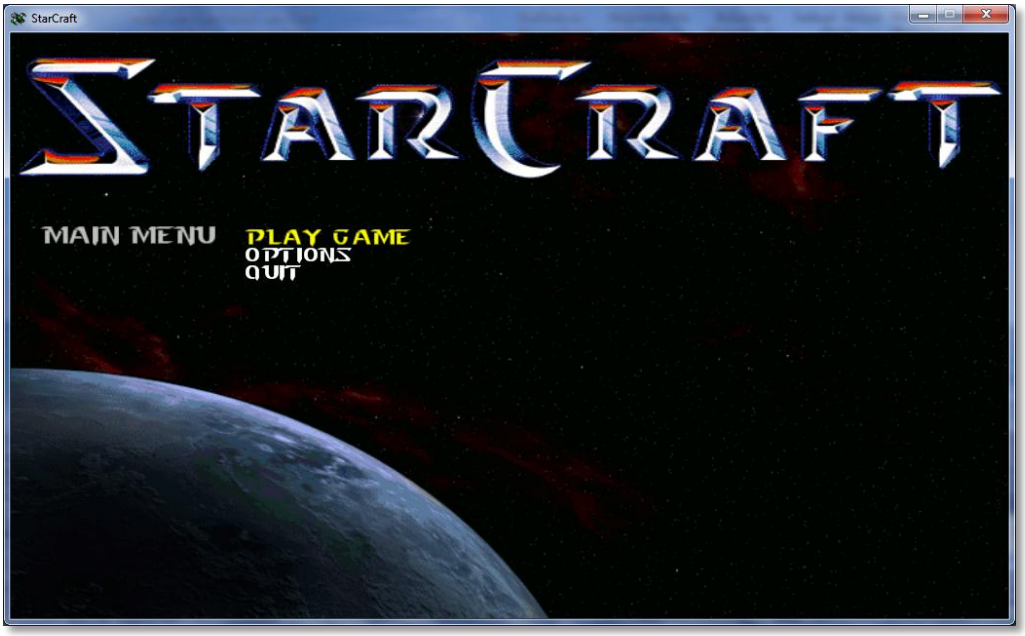

Figura 118. Menú Principal

Las opciones presentadas en el Menú Principal son:

- Play Game: lanza una Pantalla de Juego.
- Options: presenta el Menú de Opciones.
- Quit: se preguntará al usuario si desea cerrar la aplicación.

Figura 119. Menú de Opciones

El Menú de Opciones permite configurar los valores adecuados para las opciones implementadas, que son:

- Sound Effects: configura la existencia de efectos de sonido.
- Sound Effects Volume: configura el volumen de los efectos de sonido.
- Music: configura la existencia de música.
- Music Volume: configura el volumen de la música.
- Scroll Speed: configura la velocidad de desplazamiento del mapa.
- Screen Resolution: configura el tamaño de la ventana que utiliza la aplicación.
- Ally Color: configura el color de las unidades y edificios del bando aliado.
- Enemy Color: configura el color de las unidades y edificios del bando enemigo.

Además, el Menú de Opciones permite seleccionar cualquiera de las tres órdenes siguientes:

- Default Settings: devuelve los valores de todas las opciones a un conjunto de valores preestablecidos.
- Go Back: retrocede al Menú Principal sin guardar los cambios realizados. Se puede acceder a esta opción mediante la tecla “Esc”.

- Save Settings And Go Back: retrocede al Menú Principal guardando los cambios previamente.

Si se selecciona la opción “Quit” en el Menú Principal se ofrecerá la posibilidad de cerrar la aplicación. El aspecto de este Diálogo de Cierre de Aplicación es el siguiente:

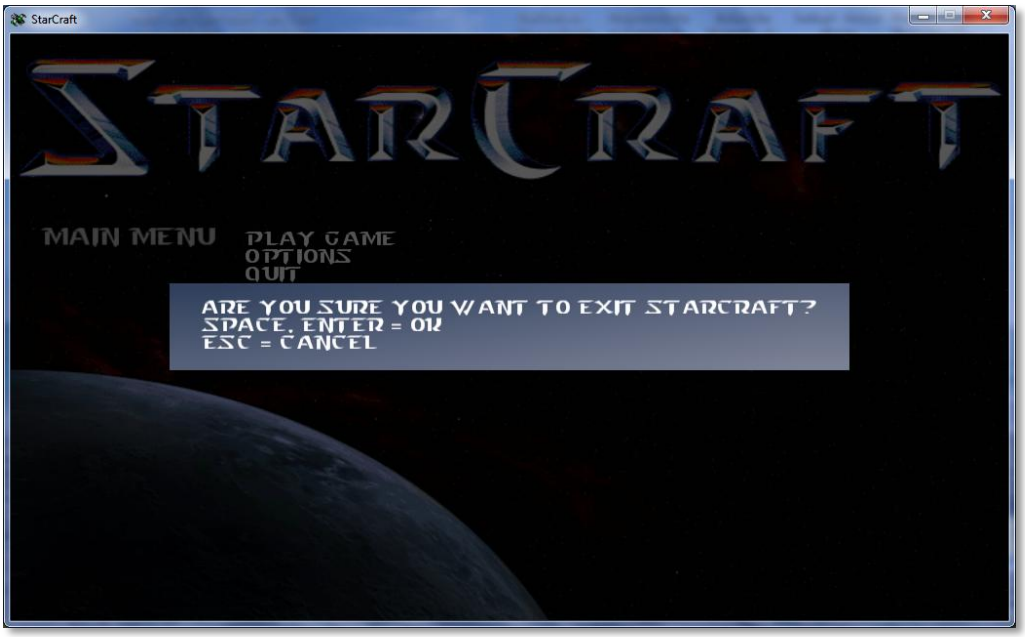

Figura 120. Diálogo de Cierre de Aplicación

Al seleccionar la opción “Play Game” del Menú Principal comenzará una nueva partida. Mientras se cargan los datos de juego se presentará una Pantalla de Carga, cuyo aspecto es el siguiente:

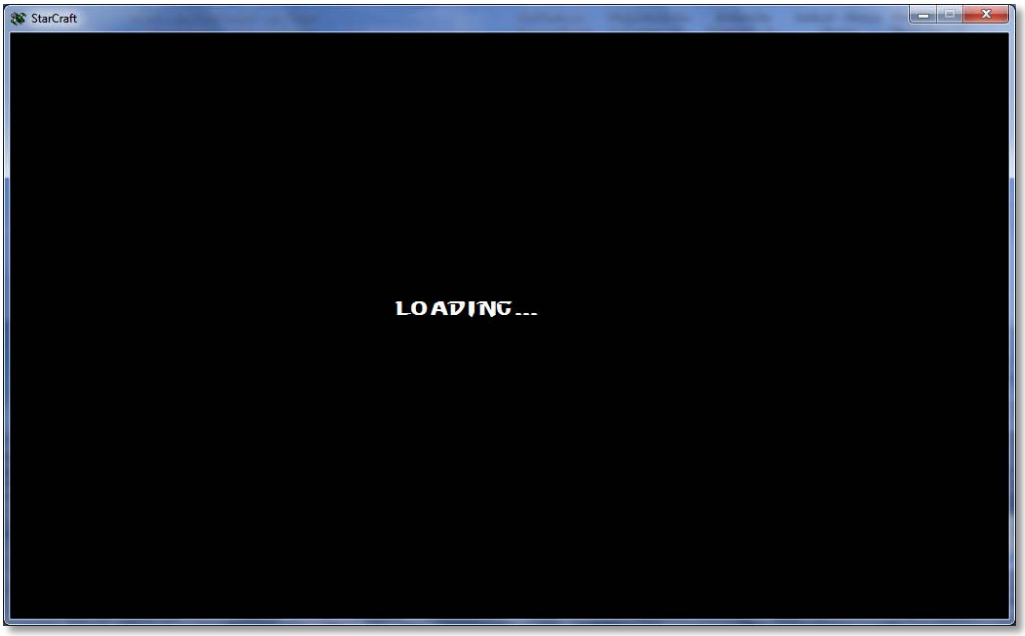

Figura 121. Pantalla de Carga

Cuando concluya el proceso se presentará la Pantalla de Juego. Para mayor información ver el apartado B.3, **Pantalla de Juego**. Un posible aspecto de la misma podría ser el siguiente:

Figura 122. Pantalla de Juego

Esta pantalla permite interactuar mediante el ratón. Desde esta pantalla se puede detener temporalmente la partida, mostrando el Menú de Pausa. El Menú de Pausa ofrece las siguientes opciones:

- Resume Game: sirve para continuar la partida donde se quedó pausada.
- Quit Game: sirve para volver al Menú Principal.

El aspecto del Menú de Pausa es el siguiente:

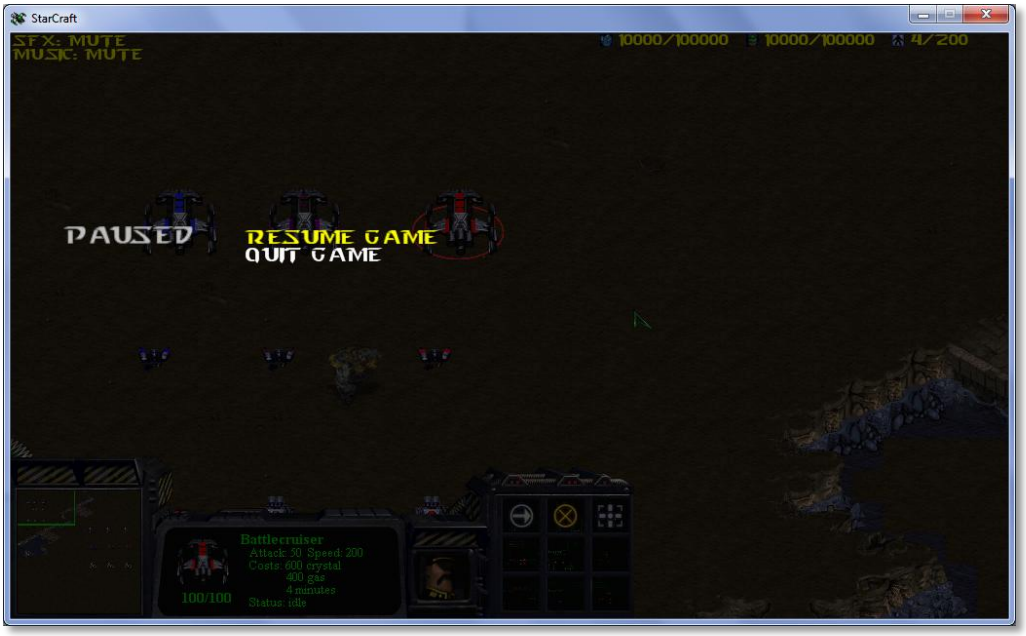

Figura 123. Menú de Pausa

Si se selecciona la opción “Quit Game”, el juego preguntará si se desea volver al Menú Principal. El aspecto de este Diálogo de Fin de Partida es el siguiente:

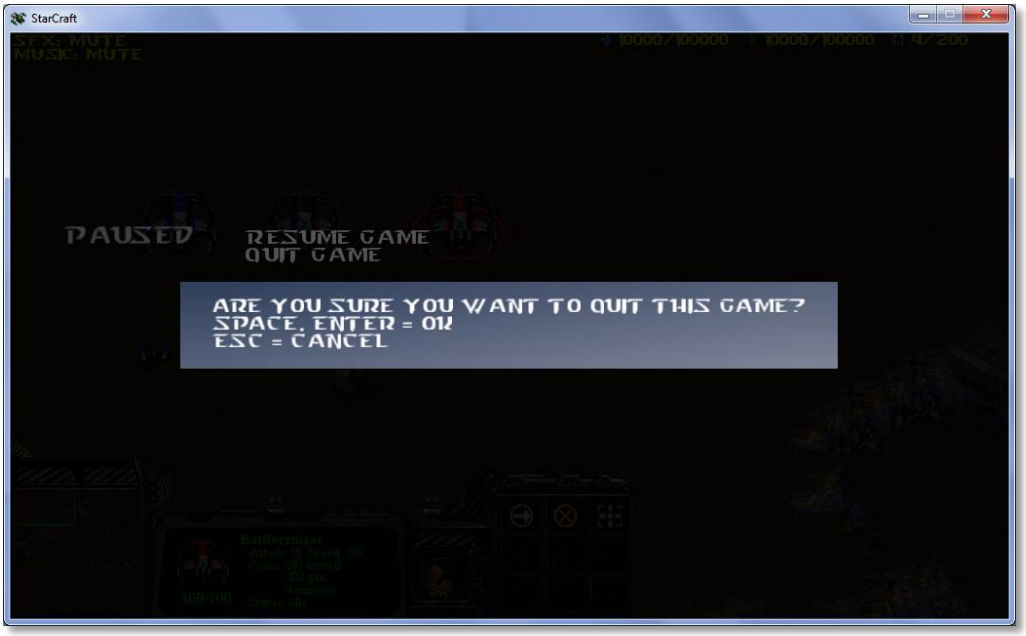

Figura 124. Diálogo de Fin de Partida

B.3. Pantalla de juego

Como se ha descrito en apartados anteriores, la Pantalla de Juego es accesible desde el Menú Principal. En este apartado se describen los elementos que aparecen en dicha Pantalla de Juego.

Figura 125. Pantalla de Juego en detalle

En la esquina superior izquierda, marcado con el número 1, se observa el volumen actual de música (“Music”) y de efectos de sonido (“SFX”). Los posibles valores están fijados entre 1 (mínimo) y 10 (máximo). También se pueden desactivar (“Mute”) o activar ambas opciones directamente.

En la esquina superior derecha, marcado con el número 2, el juego mostrará en todo momento los recursos del jugador. De izquierda a derecha son cristal, gas y población. Los dos primeros servirán para construir unidades y edificios. El último indicará el límite de unidades que es posible construir.

En la parte inferior izquierda estará situada la Barra de Control. Dicho elemento agrupa el mapa en miniatura (número 3), la información adicional (número 4), el retrato de la unidad (número 5) y las órdenes posibles (número 6).

El mapa en miniatura indica qué parte del mapa global se está viendo actualmente. También da una visión general de la acción, ya que muestra las unidades y los edificios a escala. Por último, también sirve para navegar a cualquier punto del mapa de manera instantánea.

La información adicional indica los datos de la unidad o edificio seleccionado actualmente. Estos datos son: nombre de la unidad, puntos de vida, cantidad de daño hecho al atacar, velocidad de movimiento, costes de producción y actividad que está realizando actualmente.

El retrato de la unidad ilustra la selección actual. También da una idea del rango en la escala militar: desde un trabajador de campo hasta el comandante de una poderosa nave.

La sección de órdenes posibles indica qué acciones puede realizar la unidad o edificio seleccionado. Algunas órdenes aparecerán en gris: esto se debe a que son las acciones realizadas por defecto (ver apartado B.7, **Órdenes**).

Finalmente, el resto de la superficie de pantalla será el mapa de juego. Este mapa podrá desplazarse en cualquier momento, bien acercando el cursor a uno de sus bordes o bien activando la opción por defecto sin ninguna unidad seleccionada. En el mapa aparecerán las unidades y edificios, y tendrán lugar todos los sucesos de la partida.

B.4. Bandos

Para facilitar la identificación del bando de cada unidad o edificio, el juego mostrará una imagen diferente del puntero del ratón para cada ocasión.

Figura 126. Punteros del ratón

De izquierda a derecha se puede observar el puntero normal, el puntero para aliados, el puntero para enemigos y el puntero para el bando neutro. El jugador controlará siempre el bando aliado y podrá tomar las acciones que considere adecuadas contra los otros bandos.

B.5. Unidades

Esta sección del manual de usuario incluye las unidades que dan vida al juego. Cada unidad se ilustra mediante imágenes reales y se detalla en la medida de lo posible.

Crucero de Batalla (“BattleCruiser”):

Esta nave de batalla es una de las unidades más poderosas del juego. Tiene 100 puntos de vida. Al atacar, quita 50 puntos de vida por cada disparo y tarda 4 segundos en recargar. Es capaz de desplazarse a una velocidad de 200 por el aire, con lo que evita la mayoría de obstáculos.

Figura 127. Unidad Crucero de Batalla

Vulture:

Esta rápida unidad es perfecta para realizar incursiones y luego replegarse. Tiene 50 puntos de vida. Al atacar, quita 20 puntos de vida por cada disparo y tarda 2 segundos en recargar. Es capaz de desplazarse a una velocidad de 600 por tierra, por lo que su movimiento está sujeto a ciertas restricciones.

Figura 128. Unidad Vulture

Tanque ("Tank"):

Esta unidad es lenta, pero sus disparos son muy poderosos. Tiene 80 puntos de vida. Al atacar, quita 30 puntos de vida por cada disparo y tarda 4 segundos en recargar. Es capaz de desplazarse a una velocidad de 200 por tierra, por lo que su movimiento está sujeto a ciertas restricciones.

Figura 129. Unidad Tanque

VCE ("SCV"):

Esta unidad es el Vehículo de Construcción Espacial ("Space Construction Vehicle"). Tiene un ataque muy débil pero, como indica su nombre, es capaz de construir cualquier tipo de edificio (la Base cuesta 1000 cristal, 100 gas y 10 minutos; las Barracas cuestan 500 cristal, 200 gas y 4 minutos; la Planta de Producción de Recursos cuesta 700 cristal, 500 gas y 7 minutos). También es capaz de recolectar recursos (16 cristal y 16 gas por viaje como máximo). Tiene 10 puntos de vida. Al atacar, quita 3 puntos de vida por cada disparo y tarda 2 segundos en recargar. Es capaz de desplazarse a una velocidad de 400 por tierra, por lo que su movimiento está sujeto a ciertas restricciones.

Figura 130. Unidad VCE

B.6. Edificios

En esta sección del manual se definen los edificios que forman la base del juego. Cada uno se ilustra mediante las imágenes utilizadas a lo largo de las partidas.

Base:

Este edificio actúa como centro logístico del jugador. Es capaz de producir unidades de tipo VCE (50 cristal, 0 gas, 1 minuto), así como de almacenar los recursos recolectados. Tiene 2000 puntos de vida.

Figura 131. Edificio Base

Barracas ("Barracks"):

Este edificio actúa como cuartel militar del jugador. Es capaz de producir unidades de tipo Crucero de Batalla (600 cristal, 400 gas, 4 minutos), Vulture (400 cristal, 50 gas, 2 minutos) y Tanque (300 cristal, 150 gas, 3 minutos). Tiene 1000 puntos de vida.

Figura 132. Edificio Barracas

Planta de Producción de Recursos ("Resource Production Plant"):

Este edificio actúa como centro de producción de recursos del jugador. Será capaz de producir una cantidad limitada de recursos que deberán ser extraídos mediante unidades de tipo VCE. Tiene 1000 puntos de vida.

Figura 133. Edificio Planta de Producción de Recursos

B.7. Órdenes

Para terminar de definir los aspectos que forman la jugabilidad de la aplicación se tiene esta sección, que recoge las órdenes que es posible dar a las unidades y edificios.

Atacar:

Indica a la unidad que debe entablar combate con otra unidad, cuestión que realizará en cuanto sea posible. La batalla continuará hasta que se especifique otra orden o alguna de las dos unidades muera. Esta acción es activada mediante el botón de acción por defecto sobre una unidad de otro bando.

Figura 134. Orden Atacar

Parar:

Indica a la unidad que se detenga, acción que realizará en cuanto sea posible (por ejemplo, evitará detenerse dentro del edificio de recolección de recursos).

Figura 135. Orden Parar

Mover:

Indica a la unidad que debe desplazarse a un punto del mapa concreto. La unidad acudirá a dicha posición siempre que exista un camino válido. Esta orden se especifica mediante el botón de acción por defecto sobre un punto vacío del mapa.

Figura 136. Orden Mover

Seguir a otra unidad:

Indica a la unidad que debe mantenerse lo más cerca posible de otra unidad del mismo bando. Esta orden se especifica mediante el botón de acción por defecto sobre una unidad del mismo bando.

Construir Edificio:

Indica a la unidad que debe comenzar la construcción del edificio señalado. La unidad comprobará si hay espacio (adyacente) en el mapa y recursos disponibles antes de comenzar a construir.

Figura 137. Orden Construir Edificio

Recolectar:

Indica a la unidad que debe comenzar un proceso de recolección de recursos. Esta acción se inicia mediante el botón de acción por defecto sobre un edificio recolector del mismo bando. Si la recolección es posible, la unidad comenzará un camino cíclico entre la Planta de Producción de Recursos señalada y la Base más cercana.

Construir Unidad:

Indica al edificio seleccionado que debe comenzar la construcción de una unidad concreta. Si es posible realizar la producción, el edificio mostrará el progreso de su actividad. Al completar la orden, la nueva unidad aparecerá al lado del edificio, pudiendo ser usada a voluntad del jugador.

Figura 138. Orden Construir Unidad

B.8. Controles

Para la navegación por los Menús, las teclas “Intro” y “Espacio” seleccionan la opción actual o dan a entender que se acepta lo que se expresa en pantalla. La tecla “Esc” realiza la operación inversa: da a entender que se descartan los cambios o que se rechaza lo que se expresa en pantalla.

Mientras que la Pantalla de Juego esté activa, los controles serán los siguientes:

- Botón izquierdo del ratón: selecciona algún elemento de pantalla (unidad, edificio, botón…).
- Botón derecho del ratón: indica que se desea realiza la acción por defecto (atacar, mover, recolectar…).
- Tecla “Control”: en conjunción con los botones del ratón, permite seleccionar un grupo de unidades (botón izquierdo) e impartir las órdenes que el grupo tiene en común (botón derecho). El funcionamiento de los grupos es idéntico al de las unidades utilizadas por separado.
- Acercar el puntero del ratón a un borde de la pantalla: indica que el mapa debe desplazarse en el sentido señalado por el borde.
- Tecla Fin: baja el volumen de la música.
- Tecla Inicio: sube el volumen de la música.
- Tecla AvPág: baja el volumen de los efectos de sonido.
- Tecla RePág: sube el volumen de los efectos de sonido.
- Tecla “M”: evita/permite la existencia de música y efectos de sonido.
- Tecla “Esc”: accede al Menú de Pausa.

Anexo C

Glosario, referencias y bibliografía

C.1. Glosario de abreviaturas, términos y acrónimos

[1] Videojuego comercial: Dentro de esta categoría se quiere englobar a todas aquellas producciones realizadas por empresas de manera profesional con el objetivo de obtener beneficios. Se agrupan los videojuegos producidos, desarrollados o respaldados por grandes firmas (Nintendo, SEGA, Sony, Microsoft, Konami…) y no tan grandes (Dinamic Multimedia, Alcachofa Soft, Hidden Station…). [↑]

[2] Videojuego casual: Este tipo de videojuegos también es conocido como "videojuego ocasional". Es común encontrar reglas simples y alta jugabilidad. La principal finalidad es dar una experiencia divertida y rápida al usuario, relegando a un segundo plano el desarrollo de una trama argumental o técnicas complejas de juego. Las temáticas más comunes son deportivas o de lógica mental. Los usuarios pueden tener cualquier edad y nivel de habilidad. [↑]

[3] Videojuego bajo demanda: En este caso, los productos se caracterizan por el sistema de publicación utilizado. Al usuario no se le proporciona ningún formato físico tradicional (CD, manual…), sino que tanto el pago como la recepción se realizan de manera electrónica. Tras la adquisición, el usuario guardará el juego en un dispositivo físico asociado a la plataforma de juego (generalmente, un disco duro dentro de la consola o PC). Este formato suele estar asociado a juegos con varios años de antigüedad o a aquellos originalmente diseñados para consolas de generaciones pasadas. [↑]

[4] Videojuego arcade: "Arcade" es el término genérico utilizado para designar a la mayoría de máquinas recreativas tanto antiguas como modernas. Con el tiempo, el término también hace referencia a los videojuegos que éstas contienen. Actualmente se ofrecen muchos de estos juegos según el sistema "bajo demanda" descrito previamente, por lo que aquellos productos que comparten plataforma de descarga suelen heredar, a su vez, el calificativo "arcade". [↑]

[5] Videojuego indie: Este tipo de creaciones suele promover aquellas temáticas menos conocidas por el público en general. El principal motivo es que se trata de creaciones independientes ("indie" es un americanismo con origen en la palabra "independent"; en castellano, "independiente"), entendiendo esto como el hecho de que todo el proceso de creación es realizado por parte de desarrolladores aficionados sin el apoyo de compañías comerciales. Estos individuos suelen desarrollar sus ideas en su tiempo libre, recurriendo más tarde a diversas formas de distribución digital. [↑]

[6] Videojuego social: Es el tipo de creaciones que se caracterizan por unir el ocio electrónico y la interacción entre amigos y/o desconocidos. Esta unión se materializa por dos vías principales: la primera, la situación de los juegos en páginas

web de gran afluencia; la segunda, la implantación de servidores de juego. En ambos casos se quiere motivar a los jugadores y darles una experiencia nueva, por lo que se les facilita el comparar puntuaciones entre amigos, se promueve el juego cooperativo… La temática de estos juegos es muy diversa, predominando estilos de juego sencillos y directos. [↑]

[7] Microsoft XNA Game Studio: Es el conjunto de código, documentación asociada y herramientas creado y mantenido por Microsoft para el desarrollo de videojuegos. Está basado en tecnología .NET[8]. Permite realizar desarrollos ejecutables para las plataformas de Microsoft: PC/Windows, XBOX, XBOX360, Zune, Windows Mobile y Windows Phone. [↑]

[8] Microsoft .NET Framework: Es el conjunto de conceptos, prácticas y criterios sobre el que se fundamenta el proyecto, así como uno de los productos más importantes de Microsoft. Está estructurado siguiendo un modelo de capas: la más alta ofrece soluciones comunes a necesidades de desarrollo; la más baja, sustituible, realiza la comunicación con el SO. Las capas intermedias gestionan hilos, excepciones, seguridad, recolección de basura… La finalidad de .NET es proporcionar un entorno para la creación rápida de aplicaciones, hacer transparente el acceso a redes y garantizar la independencia de la plataforma hardware. [↑]

[9] Microsoft Visual C#: Es un lenguaje de programación orientado a objetos creado y mantenido por Microsoft. Forma parte de la plataforma .NET, ya que comparten el mismo estándar internacional, pero se sitúa un nivel por encima: permite desarrollar programas independientes utilizando parte de .NET. Este lenguaje se ha visto influenciado por C, C++, Java y Delphi. Actualmente existen diversos compiladores: el proporcionado por Microsoft para PC/Windows, el proyecto Mono para UNIX/Linux… [↑]

[10] Videojuego de estrategia en tiempo real: Este tipo de videojuegos tiene dos características fundamentales: en primer lugar, los jugadores controlan una serie de recursos propios de manera estratégica (personajes, edificios, economía…); en segundo lugar, las acciones tienen lugar de manera proporcional al tiempo transcurrido en la vida real. Para ampliar esta información, consultar el capítulo 2, **Estado de la cuestión.** [↑]

[11] Software libre: Es aquel equipamiento lógico de un ordenador que respeta la libertad de los usuarios. De esta manera, tras ser adquirido, dicho equipamiento permite ser utilizado, estudiado, cambiado y redistribuido sin limitación alguna. Esto se hace posible mediante licencias de código suficientemente permisivas. Normalmente, las redistribuciones están obligadas a ser libres a su vez. [↑]

[12] Software propietario: En contraposición al software libre, este tipo de programas informáticos tiene limitaciones para ser utilizado, estudiado, cambiado o redistribuido con (o sin) modificaciones. Los derechos mencionados obran en poder de

la persona física o jurídica que creó el programa, quien a su vez podrá autorizar a quien considere adecuado. [↑]

[13] Wargame: Término que hace referencia a la temática de un juego cuando ésta tiene características bélicas: batallas, conquistas, ataques… [↑]

[14] Plataformas móviles: Dispositivos electrónicos ligeros y de pequeño tamaño que permiten la ejecución de videojuegos. Proporcionan una interfaz de control, una o más pantallas, un subsistema de sonido (cascos, altavoces), alimentación (pilas, baterías) y, opcionalmente, conectividad con otras plataformas móviles similares. [↑]

C.2. Referencias

[r1] Wikipedia: Cathode Ray Amusement Device [en línea] [en inglés]. Disponible en WorldWide Web: <http://en.wikipedia.org/wiki/Missile_simulator_game> [↑]

[r2] Brookhaven National Laboratory: The First Video Game? [en línea] [en inglés]. Disponible en WorldWide Web: <http://www.bnl.gov/bnlweb/history/higinbotham.asp> [↑]

[r3] Computer History Museum: Restoring the DEC PDP-1 Computer Exhibit [en línea] [en inglés]. Disponible en WorldWide Web: <http://pdp-1.computerhistory.org/pdp-1/?f=theme&s=4&ss=3> [↑]

[r4] Pong-Story: Atari home PONG systems [en línea] [en inglés]. Disponible en WorldWide Web: <http://www.pong-story.com/atpong2.htm> [↑]

[r5] Blizzard Entertainment: StarCraft [en línea]. Disponible en WorldWide Web: <http://eu.blizzard.com/es-es/games/sc/> [↑]

[r6] Dune II [en línea] [en inglés]. Disponible en WorldWide Web: <http://duneii.com/> [↑]

[r7] Command & Conquer: Home [en línea] [en inglés]. Disponible en WorldWide Web: <http://www.commandandconquer.com/> [↑]

[r8] Microsoft: Age of Empires (r) [en línea]. Disponible en WorldWide Web: <http://www.microsoft.com/spain/juegos/aoe/empires.aspx> [↑]

[r9] Blizzard Entertainment: Juegos clásicos [en línea] [en inglés]. Disponible en WorldWide Web: <http://eu.blizzard.com/es-es/games/legacy/> [↑]

[r10] Blizzard Entertainment: StarCraft II [en línea] [en inglés]. Disponible en WorldWide Web: <http://eu.blizzard.com/es-es/games/sc2/> [↑]

[r11] Wikipedia: Harry Potter y el Misterio del Príncipe (película) [en línea]. Disponible en WorldWide Web: <http://es.wikipedia.org/wiki/Harry_Potter_y_el_misterio_del_pr%C3%ADncipe_(pel%C3%ADcula)> [↑]

[r12] Los Angeles Times: Video game borrows page from Hollywood playbook [en línea] [en inglés]. Disponible en WorldWide Web: <http://articles.latimes.com/2009/nov/18/business/fi-ct-duty18> [↑]

[r13] Revista Privilege: Conoce las 21 películas más taquilleras de todos los tiempos [en línea]. Disponible en WordWide Web: <http://www.revistaprivilege.net/2010/01/23/conoce-las-21-peliculas-mas-taquilleras-de-todos-los-tiempos/> [↑]

[r14] ElGeek: Los juegos más vendidos de la historia [en línea]. Disponible en WordWide Web: <http://elgeek.com/los-videojuegos-mas-vendidos-de-la-historia> [↑]

[r15] Asociación Española de Distribuidores y Editores de Software de Entretenimiento (AdEsE) [en línea]. Disponible en WordWide Web: <http://www.adese.es/> [↑]

[r16] AdEsE: Recopilación de anuarios [en línea]. Disponible en WordWide Web: <http://www.adese.es/index.php?option=com_content&view=article&id=56&Itemid=4> [↑]

[r17] AdEsE: Nota de prensa “Adese presenta su anuario 2009 y las expectativas de la industria para la campaña 2010” [en línea]. Disponible en WordWide Web: <http://www.adese.es/notas-prensa/np_22-09-10.html> [↑]

[r18] Gradius Wiki [en línea] [en inglés]. Disponible en WordWide Web: <http://gradius.wikia.com/wiki/Gradius_Wiki> [↑]

[r19] Wikipedia: Galaxian [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Galaxian> [↑]

[r20] Xbox.com: Halo Official Site [en línea] [en inglés]. Disponible en WordWide Web: <http://halo.xbox.com/en-us> [↑]

[r21] Rockstar Games: Max Payne Official Site [en línea] [en inglés]. Disponible en WordWide Web: <http://www.rockstargames.com/maxpayne/index.html> [↑]

[r22] Xbox.com: Gears of War Official Site [en línea] [en inglés]. Disponible en WordWide Web: <http://gearsofwar.xbox.com/default.htm> [↑]

[r23] Wikipedia: R-Type [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/R-Type> [↑]

[r24] Sega.com: The House of The Dead Overkill Official Site [en línea] [en inglés]. Disponible en WordWide Web: <http://www.sega.com/hodoverkill/index3.html> [↑]

[r25] Wikipedia: Time Crisis Series [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Time_Crisis_(series)> [↑]

[r26] Wikipedia: Tekken [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Tekken> [↑]

[r27] Wikipedia: Mortal Kombat (serie) [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Mortal_Kombat_(serie)> [↑]

[r28] Wikipedia: Anexo Videojuegos de Street Fighter [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Anexo:Videojuegos_de_Street_Fighter> [↑]

[r29] Wikipedia: The King of Fighters [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/The_King_of_Fighters> [↑]

[r30] Smash Bros. DOJO!! [en línea]. Disponible en WordWide Web: <http://www.smashbros.com/es/index.html> [↑]

[r31] Wikipedia: Double Dragon [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Double_Dragon> [↑]

[r32] Wikipedia: Final Fight [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Final_Fight> [↑]

[r33] Wikipedia: Battletoads [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Battletoads> [↑]

[r34] Wikipedia: Devil May Cry (serie) [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Devil_May_Cry_(serie)> [↑]

[r35] God of War Home [en línea]. Disponible en WordWide Web: <http://www.godofwar.com/es_ES/index.htm#Inicio> [↑]

[r36] WarHammer 40000 Dawn of War España [en línea]. Disponible en WordWide Web: <http://www.dawnofwargame.com/es/home/index> [↑]

[r37] Wikipedia: Heroes of Might and Magic [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Heroes_of_Might_and_Magic> [↑]

[r38] Wikipedia: SimCity [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/SimCity> [↑]

[r39] Sega.es: Football Manager 2011 [en línea]. Disponible en WordWide Web: <http://www.sega.es/games/football-manager-2011/> [↑]

[r40] Wikipedia: Theme Park World [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Theme_Park_World> [↑]

[r41] EA.com: Los Sims [en línea]. Disponible en WordWide Web: <http://www.lossims.ea.com/pages.view_frontpage.asp> [↑]

[r42] Gran-turismo.com [en línea]. Disponible en WordWide Web: <http://eu.gran-turismo.com/es/> [↑]

[r43] Microsoft.com: Flight Simulator X [en línea]. Disponible en WordWide Web: <http://www.microsoft.com/latam/juegos/fsx/default.aspx> [↑]

[r44] Wikipedia: Mario Kart (serie) [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Super_Mario_Kart> [↑]

[r45] Need for Speed Racing Game [en línea]. Disponible en WordWide Web: <http://www.needforspeed.com/es_ES/> [↑]

[r46] Rcokstar Games: Midnight Club Los Angeles [en línea] [en inglés]. Disponible en WordWide Web: <http://www.rockstargames.com/midnightclubLA/> [↑]

[r47] Konami: Pro Evolution Soccer 2011 [en línea]. Disponible en WordWide Web: <http://www.konami-pes2011.com/> [↑]

[r48] EA Sports: FIFA 11 [en línea]. Disponible en WordWide Web: <http://www.ea.com/es/futbol/fifa> [↑]

[r49] EA Sports: NBA Live 10 [en línea]. Disponible en WordWide Web: <http://www.ea.com/es/juegos/nba-live> [↑]

[r50] EA Sports: NBA Jam [en línea] [en inglés]. Disponible en WordWide Web: <http://nba-jam.easports.com/home.action> [↑]

[r51] EA Sports: FIFA Street [en línea] [en inglés]. Disponible en WordWide Web: <http://www.fifastreet3.com/flash/wk/player.asp> [↑]

[r52] Wikipedia: Blitz The League II [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Blitz:_The_League_II> [↑]

[r53] BioWare: Baldur's Gate [en línea] [en inglés]. Disponible en WordWide Web: <http://www.bioware.com/games/baldurs_gate/> [↑]

[r54] BioWare: Dragon Age II [en línea] [en inglés]. Disponible en WordWide Web: <http://dragonage.bioware.com/> [↑]

[r55] Wikipedia: Icewind Dale [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Icewind_Dale> [↑]

[r56] BioWare: Neverwinter Nights [en línea]. Disponible en WordWide Web: <http://nwn.bioware.com/about/description.html> [↑]

[r57] Wikipedia: Fallout 2 [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Fallout_2> [↑]

[r58] BioWare: Mass Effect 2 [en línea] [en inglés]. Disponible en WordWide Web: <http://masseffect.bioware.com/> [↑]

[r59] Bethesda Game Studios: The Elder Scrolls III Morrowind [en línea]. Disponible en WordWide Web: <http://www.elderscrolls.com/morrowind/> [↑]

[r60] Bethesda Game Studios: The Elder Scrolls IV Oblivion [en línea]. Disponible en WordWide Web: <http://www.elderscrolls.com/oblivion/> [↑]

[r61] Nintendo Zelda Universe: The Legend of Zelda [en línea] [en inglés]. Disponible en WordWide Web: <http://www.zelda.com/universe/game/zelda/> [↑]

[r62] Bethesda Game Studios: Fallout 3 [en línea]. Disponible en WordWide Web: <http://fallout.bethsoft.com/spa/home/home.php?country=es> [↑]

[r63] Wikipedia: Final Fantasy [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Final_Fantasy> [↑]

[r64] Square Enix: Star Ocean [en línea] [en inglés]. Disponible en WordWide Web: <http://na.square-enix.com/starocean/> [↑]

[r65] Wikipedia: Lost Odyssey [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Lost_Odyssey> [↑]

[r66] XSeed Games: Wild Arms 5 [en línea] [en inglés]. Disponible en WordWide Web: <http://www.wildarms5.com/> [↑]

[r67] Square Enix: Front Mission Evolved [en línea]. Disponible en WordWide Web: <http://www.frontmissionevolved.es/> [↑]

[r68] Square Enix: Final Fantasy Tactics [en línea] [en inglés]. Disponible en WordWide Web: <http://na.square-enix.com/fftactics/> [↑]

[r69] Wikipedia: Fire Emblem [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Fire_Emblem> [↑]

[r70] Wikipedia: Colossal Cave Adventure [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Colossal_Cave_Adventure> [↑]

[r71] Wikipedia: Monkey Island series [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Monkey_Island_(series)> [↑]

[r72] Wikipedia: Simon The Sorcerer [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Simon_the_Sorcerer> [↑]

[r73] Funcom: The Longest Journey [en línea] [en inglés]. Disponible en WordWide Web: <http://www.longestjourney.com/> [↑]

[r74] Wikipedia: Grim Fandango [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Grim_Fandango> [↑]

[r75] Sega: Shenmue [en línea] [en inglés]. Disponible en WordWide Web: <http://www.shenmue.com/eng/index_egn.html> [↑]

[r76] Wikipedia: Fahrenheit [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Fahrenheit_(video_game)> [↑]

[r77] Quantic Dream: Heavy Rain [en línea] [en inglés]. Disponible en WordWide Web: <http://www.quanticdream.com/en/game/heavy-rain> [↑]

[r78] Wikipedia: Arkanoid [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Arkanoid> [↑]

[r79] Tetris Official Web Site [en línea] [en inglés]. Disponible en WordWide Web: <http://www.tetris.com/> [↑]

[r80] Wikipedia: Zuma (video game) [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Zuma_(video_game)> [↑]

[r81] Wikipedia: Super Pang! [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Super_Pang> [↑]

[r82] Wikipedia: Bust-A-Move (Puzzle Bobble) [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Puzzle_Bobble Referencia saga Bust-A-Move> [↑]

[r83] Sony Computer Entertainment: Lemmings [en línea]. Disponible en WordWide Web: < http://www.lemmings.tv/> [↑]

[r84] EA: RockBand [en línea] [en inglés]. Disponible en WordWide Web: <http://www.rockband3.co.uk/> [↑]

[r85] Activision: Guitar Hero [en línea] [en inglés]. Disponible en WordWide Web: <http://hub.guitarhero.com/> [↑]

[r86] Activision: DJ Hero [en línea] [en inglés]. Disponible en WordWide Web: <http://www.djhero.com/es-es/home/index> [↑]

[r87] Sony: Singstar [en línea]. Disponible en WordWide Web: <http://www.singstar.com/es_ES/singstar.html> [↑]

[r88] Wikipedia: Beatmania [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Beatmania> [↑]

[r89] Nintendo: Wii Fit [en línea]. Disponible en WordWide Web: <http://www.nintendo.es/NOE/es_ES/games/wii/wii_fit_2841.html> [↑]

[r90] Nintendo: Wii Sports [en línea]. Disponible en WordWide Web: <http://www.nintendo.es/NOE/es_ES/games/wii/wii_sports_2781.html> [↑]

[r91] DDRgame.com: Dance Dance Revolution [en línea] [en inglés]. Disponible en WordWide Web: <http://www.ddrgame.com/> [↑]

[r92] Ubisoft: Just Dance [en línea]. Disponible en WordWide Web: <http://justdancegame.es.ubi.com/> [↑]

[r93] Wikipedia: Eye Toy Play [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/EyeToy:_Play_3> [↑]

[r94] Nintendo: Mario Party [en línea] [en inglés]. Disponible en WordWide Web: <http://www.nintendo.com/sites/mp8/> [↑]

[r95] Wikipedia: Sonic Shuffle [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Sonic_Shuffle> [↑]

[r96] Wikipedia Fuzion Frenzy [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Fuzion_Frenzy> [↑]

[r97] Sony Computer Entertainment: Buzz The Game [en línea]. Disponible en WordWide Web: <http://www.buzzthegame.com/es-es/> [↑]

[r98] Disney: Scene It? [en línea] [en inglés]. Disponible en WordWide Web: <http://www.sceneit.com/> [↑]

[r99] Disney: Disney´s Think Fast! [en línea] [en inglés]. Disponible en WordWide Web: <http://disney.go.com/disneyinteractivestudios/thinkfast/> [↑]

[r100] Wikipedia: Super Mario Bros. [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Super_Mario_Bros.> [↑]

[r101] Ubisoft: Prince of Persia [en línea]. Disponible en WordWide Web: <http://prince-of-persia.es.ubi.com/> [↑]

[r102] Wikipedia: Mirror's Edge [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Mirror's_Edge> [↑]

[r103] Nintendo: Brain Training [en línea]. Disponible en WordWide Web: <http://www.nintendo.es/NOE/es_ES/games/nds/brain_training_del_dr_kawashima_cuntos_aos_tiene_tu_cerebro_3234.html> [↑]

[r104] Ubisoft: Mind Quiz [en línea]. Disponible en WordWide Web: <http://www.ubi.com/ES/Games/Info.aspx?pId=5406> [↑]

[r105] Nintendo: English Training [en línea]. Disponible en WordWide Web: <http://www.nintendo.es/NOE/es_ES/games/nds/english_training_disfruta_y_mejora_tu_ingls_3125.html> [↑]

[r106] Nintendo: Training For Your Eyes [en línea]. Disponible en WordWide Web: <http://www.nintendo.es/NOE/es_ES/games/nds/training_for_your_eyes_entrena_y_relaja_la_vista_6386.html> [↑]

[r107] Nintendo: Big Brain Academy [en línea]. Disponible en WordWide Web: <http://www.nintendo.es/NOE/es_ES/games/wii/big_brain_academy_para_wii_2774.html> [↑]

[r108] Ubisoft: Far Cry 2 [en línea]. Disponible en WordWide Web: <http://farcry.es.ubi.com/> [↑]

[r109] 2K Games: Bioshock [en línea] [en inglés]. Disponible en WordWide Web: <http://www.2kgames.com/bioshock/enter.html> [↑]

[r110] Nintendo Entertainment System – Nintendo [en línea]. Disponible en WordWide Web: <http://www.nintendo.es/NOE/es_ES/systems/nintendo_entertainment_system_1165.html> [↑]

[r111] Wikipedia: Tetris [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Tetris> [↑]

[r112] Wikipedia: PacMan [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Pac-Man> [↑]

[r113] 3D Realms: Wolfenstein 3D [en línea] [en inglés]. Disponible en WordWide Web: <http://www.3drealms.com/wolf3d/> [↑]

[r114] Electronic Arts: Classic Live SimCity [en línea] [en inglés]. Disponible en WordWide Web: <http://simcity.ea.com/play/simcity_classic.php> [↑]

[r115] HASBRO: Risk [en línea] [en inglés]. Disponible en WordWide Web: <http://www.hasbro.com/risk/default.cfm> [↑]

[r116] Wikipedia: Herzog Zwei [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Herzog_Zwei> [↑]

[r117] Wikipedia: Mega-Lo-Mania [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Mega_Lo_Mania> [↑]

[r118] Wikipedia: Stonkers [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Stonkers> [↑]

[r119] Wikipedia: Herzog [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Herzog_(video_game)> [↑]

[r120] Wikipedia: WarCraft [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Warcraft:_Orcs_%26_Humans> [↑]

[r121] The official Dune website [en línea] [en inglés]. Disponible en WordWide Web: <http://www.dunenovels.com/> [↑]

[r122] Wikipedia: Frank Herbert [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Frank_Herbert> [↑]

[r123] Blizzard Entertainment: Battle.net [en línea]. Disponible en WordWide Web: <http://eu.battle.net/es/> [↑]

[r124] Wikipedia: Sports in South Korea [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/South_Korea#Sports> [↑]

[r125] Wikipedia: StarCraft professional competition [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/StarCraft_professional_competition> [↑]

[r126] Wikipedia: Total Annihilation [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Total_Annihilation> [↑]

[r127] Relic.com: Homeworld [en línea] [en inglés]. Disponible en WordWide Web: <http://www.relic.com/games/homeworld/> [↑]

[r128] Relic.com: Homeworld 2 [en línea] [en inglés]. Disponible en WordWide Web: <http://www.relic.com/games/homeworld-2/> [↑]

[r129] Blizzard Entertainment: WarCraft III Reign of Chaos [en línea] [en inglés]. Disponible en WordWide Web: <http://us.blizzard.com/en-us/games/war3/> [↑]

[r130] Wikipedia: Emperor: Battle for Dune [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Emperor:_Battle_for_Dune> [↑]

[r131] Electronic Arts: El Señor de los Anillos: La Batalla por la Tierra Media [en línea]. Disponible en WordWide Web: <http://www.ea.com/es/juegos/senor-anillos-batalla-tierra-media-i> [↑]

[r132] TotalWar.com: Rome Total War [en línea]. Disponible en WordWide Web: <http://www.totalwar.com/rome> [↑]

[r133] Electronic Arts: Command and Conquer Generals [en línea]. Disponible en WordWide Web: <http://www.ea.com/es/juegos/command-and-conquer-generals> [↑]

[r134] Microsoft: Age of Empires III [en línea]. Disponible en WordWide Web: <http://www.microsoft.com/spain/juegos/ageIII/ageIII.aspx> [↑]

[r135] Havok.com: Home [en línea] [en inglés]. Disponible en WordWide Web: <http://www.havok.com/> [↑]

[r136] Wikipedia: El Señor de los Anillos [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/El_Se%C3%B1or_de_los_Anillos> [↑]

[r137] TotalWar.com: Shogun Total War [en línea]. Disponible en WordWide Web: <http://www.totalwar.com/shogun> [↑]

[r138] Electronic Arts: Command and Conquer 3 [en línea]. Disponible en WordWide Web: <http://www.ea.com/es/juegos/command-and-conquer-tiberium-wars> [↑]

[r139] Official Halo Wars Community Site: Home [en línea] [en inglés]. Disponible en WordWide Web: <http://www.halowars.com/> [↑]

[r140] Nintendo: página oficial de Pikmin [en línea] [en inglés]. Disponible en WordWide Web: <http://www.pikmin.com/> [↑]

[r141] Havas Media: Home [en línea] [en inglés]. Disponible en WordWide Web: <http://www.havasmedia.com/> [↑]

[r142] Vivendi: Home [en línea] [en inglés]. Disponible en WordWide Web: <http://www.vivendi.com/vivendi/-accueil-en-> [↑]

[r143] Wikipedia: WarCraft Adventures Lord of The Clans [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Warcraft_Adventures:_Lord_of_the_Clans> [↑]

[r144] Wikipedia: StarCraft Ghost [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/StarCraft:_Ghost> [↑]

[r145] Wikipedia: StarCraft Brood War [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/StarCraft:_Brood_War> [↑]

[r146] CreepColony – StarCraft StarEdit Utilities [en línea] [en inglés]. Disponible en WordWide Web: <http://www.creepcolony.com/staredit.shtml> [↑]

[r147] Guinness World Records [en línea]. Disponible en WordWide Web: <http://www.guinnessworldrecords.com/es/> [↑]

[r148] World News.com: UC Berkeley StarCraft Class Lecture 1 [en línea] [en inglés]. Disponible en WordWide Web: <http://wn.com/UC_Berkeley_Starcraft_Class_Lecture_1> [↑]

[r149] Blender.org: Home [en línea] [en inglés]. Disponible en WordWide Web: <http://www.blender.org/> [↑]

[r150] Crystal Space: Main Page [en línea] [en inglés]. Disponible en WordWide Web: <http://www.crystalspace3d.org/main/Main_Page> [↑]

[r151] Wikipedia: DIV Games Studio [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/DIV_Games_Studio> [↑]

[r152] Wikipedia: Gamebryo [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/Gamebryo> [↑]

[r153] YoYoGames: Game Maker [en línea] [en inglés]. Disponible en WordWide Web: <http://www.yoyogames.com/gamemaker> [↑]

[r154] Game Studio Game Development System [en línea] [en inglés]. Disponible en WordWide Web: <http://www.3dgamestudio.com/> [↑]

[r155] Qeradiant.com: GtkRadiant [en línea] [en inglés]. Disponible en WordWide Web: <http://www.qeradiant.com/> [↑]

[r156] id Software: Quake III Arena [en línea] [en inglés]. Disponible en WordWide Web: <http://www.quake3arena.com/> [↑]

[r157] Wikipedia: Havok (software) [en línea]. Disponible en WordWide Web: <http://es.wikipedia.org/wiki/Havok_(software)#Juegos_que_utilizan_Havok> [↑]

[r158] lwjgl.org: Home of the Lightweight Java Game Library [en línea] [en inglés]. Disponible en WordWide Web: <http://lwjgl.org/> [↑]

[r159] elecbyte: MUGEN [en línea] [en inglés]. Disponible en WordWide Web: <http://www.elecbyte.com/mugen> [↑]

[r160] ogre3d.org: Home [en línea] [en inglés]. Disponible en WordWide Web: <http://www.ogre3d.org/> [↑]

[r161] SOURCE: Home [en línea] [en inglés]. Disponible en WordWide Web: <http://source.valvesoftware.com/> [↑]

[r162] Unreal Technology: Unreal Engine [en línea] [en inglés]. Disponible en WordWide Web: <http://www.unrealengine.com/> [↑]

[r163] Wikipedia: List of Unreal Engine games [en línea] [en inglés]. Disponible en WordWide Web: <http://en.wikipedia.org/wiki/List_of_Unreal_Engine_games> [↑]

[r164] Microsoft.com: XNA Game Studio 4.0 [en línea] [en inglés]. Disponible en WordWide Web: <http://msdn.microsoft.com/en-us/library/bb200104.aspx> [↑]

[r165] App Hub: Game State Management [en línea] [en inglés]. Disponible en WordWide Web: <http://create.msdn.com/en-US/education/catalog/sample/game_state_management> [↑]

[r166] MSDN.com: App Hub [en línea] [en inglés]. Disponible en WordWide Web: <https://users.create.msdn.com/Register> [↑]

C.3. Bibliografía

- ARCHER, T. *A fondo C#*. McGraw-Hill, 2001, 374 páginas.
- FERGUSON, J. *La Biblia de C#*. Anaya Multimedia, 2003, 861 páginas.
- LOBAO, A. S., EVANGELISTA, B. P., DE FARIAS, J.A.L., GROOTJANS, R. *Beginning XNA 3.0 Game Programming: From Novice to Professional*. APress, 2009, 422 páginas.
- CAWOOD, S., McGEE, P. *Microsoft XNA Game Studio Creator's Guide: An Introduction to XNA Game Programming*. McGraw-Hill, 2009, 482 páginas.
- NITSCHKE, B., *Professional XNA Game Programming: For XBOX 360 and Windows*. John Wiley & Sons, 2007, 476 páginas.