VIDEOJUEGO RPG PARA PC “MITOLOGÍA COLOMBIANA”

JUAN CARLOS RUIZ PACHECO

JAIRO ALEXANDER GÓMEZ MOLINA

WILLIAM ALEJANDRO GÓMEZ CASTILLO

UNIVERSIDAD CATÓLICA DE COLOMBIA

FACULTAD DE INGENIERÍA DE SISTEMAS

BOGOTA

2005

VIDEOJUEGO RPG PARA PC “MITOLOGÍA COLOMBIANA”

JUAN CARLOS RUIZ PACHECO

JAIRO ALEXANDER GÓMEZ MOLINA

WILLIAM ALEJANDRO GÓMEZ CASTILLO

Trabajo de Grado

Director

INGENIERO JAIME ÁLVAREZ

UNIVERSIDAD CATÓLICA DE COLOMBIA

FACULTAD DE INGENIERÍA DE SISTEMAS

BOGOTA

2005

Nota de aceptación: .

_____
_____
_____
_____
_____

_____
Firma Presidente del jurado .

_____
Firma del jurado .

_____
Firma del jurado .

Bogotá D.C. __ de _____ de 2005

A mi madre quien es y siempre será
la inspiración de todas mis metas y triunfos.

A mi hermano, por su apoyo y lealtad.

Juan Carlos Ruiz.

A mi familia por su respaldo incondicional
y sus sabios consejos.

A mi abuela, que en paz descanse,
Gracias por tu apoyo.

Alexander Gómez.

A mi madre quien es mi guía y mi meta
A quien le debo lo que soy y seré

A hermana mayor por haber compartido
Todos estos años de felicidad

A mi hermana menor eres la luz de mis ojos
y lo mas especial que me pudo pasar

William Gómez.

AGRADECIMIENTOS

Comunidad de desarrolladores de videojuegos de Teleportmedia S.A (CostaRica), por su guía y su ayuda para resolver algunas problemas en el desarrollo:

- Álvaro Tejada (Blag) de Perú
- Marco Alvarado (Cronodragón) de Costa Rica
- Yevhen Yakhnenko (Enko) de Rusia (actualmente Argentina)
- Rubén Moreno Montolíu (Ruben3D) de España
- Ibito
- Amilus
- Geo
- Middrel

Comunidad de desarrolladores SoloCódigo por su colaboración en la inspección de documentos guía y su ayuda para resolver algunas problemas en el desarrollo.

- Sergio Pacho Benedé (Solocodigo) de España
- Jonathan Enrique Kraft (© Jonathan ©) de Argentina
- Guillermo Alfonso Morales (RadicalEd) de Colombia

Comunidad de VB-Mundo por sus acertados y elogiantes comentarios acerca del proyecto

- Pablo Tilotta (Chiaravel) de Argentina

Diseño gráfico base de elementos complementarios escena uno:

- Andrius Gabriunas

CONTENIDO

pág.

pág.

pág.

pág.

pág.

pág.

pág.

pág.

pág.

pág.

pág.

pág.

pág.

pág.

pág.

LISTA DE FIGURAS

pág.

pág.

pág.

pág.

pág.

pág.

pág.

pág.

LISTA DE TABLAS

pág.

LISTA DE ANEXOS

pág.

GLOSARIO

2D, DOS DIMENSIONES: los planos son bidimensionales, y sólo pueden contener cuerpos unidimensionales o bidimensionales.

3D, TRES DIMENSIONES: las tres dimensiones son el largo, el ancho y la profundidad de una imagen. Hay casos en que se usa las tres dimensiones en computación para simular una realidad virtual. También se utilizan para hacer animaciones. Otro concepto conocido es el sonido 3D.

ACTIVEX: es una tecnología de Microsoft para el desarrollo de páginas dinámicas.
Tiene presencia en la programación del lado del servidor y del lado del cliente, aunque existan diferencias en el uso en cada uno de esos dos casos.

API, ADVANCED PROGRAMMING INTERFACE: es un conjunto de especificaciones de comunicación entre componentes software. Representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla. De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio.

ASF, ADVANCED STREAMING FORMAT: este formato de archivos almacena información de audio y video, y fue especialmente diseñado para trabajar en redes, como Internet. La información es descargada como un flujo continuo de datos, y por ende, no es necesario esperar la descarga completa del archivo para poder reproducirlo.

AVI, AUDIO VIDEO INTERLEAVED: sonido y vídeo entrelazados. Formato para archivos multimedia que puede contener tanto imagen como sonido. Para leer este tipo de archivos se necesita un lector como Windows Media Player.

BACKGROUND: son básicamente los elementos "de fondo" que hay en los videojuegos y que suelen dar una cierta atmósfera o ambiente al mismo.

BITMAP: formato basado en "mapa de puntos", es uno de los formatos posibles para la conservación de imágenes, usado para fotografías y gráfica analógica (como caricaturas y pinturas). Se opone al vectorial, que utiliza coordenadas geométricas y fórmulas trigonométricas.

BMP, ARCHIVO DE MAPA DE BITS: la extensión del nombre que se da a archivos de imágenes gráficas de mapa de bit.

BOXING: proceso de la Common Language Runtime que permite convertir los tipos de valor en tipos de referencia para acceder mediante una variable del tipo Object, que es el tipo base universal.

BIT, BINARY DIGIT: unidad mínima de información que puede ser transmitida o tratada y puede tener un valor de 0 (cero) ó 1 (uno).

BYTE: se describe como la unidad básica de almacenamiento de información, generalmente equivalente a ocho bits, pero el tamaño del byte depende del código de información en el que se defina.

CARTUCHO: en los videojuegos es un dispositivo para almacenar los juegos de una consola en una tarjeta de circuitos cubierta de una caja de plástico. Se suele aplicar este término incluso a un videojuego en formato de disco.

CLIPPING: este es el proceso que remueve porciones de una imagen que no están definidas por el campo visual de la cámara.

CONSOLA: es un sistema de hardware que se conecta a una pantalla de televisor para poder jugar videojuegos. Los videojuegos vienen almacenados en cartuchos, CD-ROMS o DVD, que para que sean ejecutados en la consola. En la actualidad las consolas poseen dispositivos para conexión a Internet, poseen capacidades gráficas y operativas similares a las de un PC permitiendo reproducir también música y video.

CROSS-FADING: es un efecto que permite hacer transiciones automáticas entre canciones, disminuyendo el volumen de la canción actual unos segundos antes de que se acabe e incrementando el volumen de la siguiente en lista al mismo tiempo.

CRT, CATHODE RAY TUBE: tubo de rayos catódicos. Es la forma en que trabajan muchos de las pantallas de TV o monitores de computadora. Existen otras tecnologías como el LCD, o pantalla de cristal líquido.

DISKETTE: es un tipo de dispositivo de almacenamiento de datos de una capacidad habitualmente de 1.44 MB, formado por una pieza circular de un material magnético que permite la grabación y lectura de datos, fino y flexible encerrado en una carcasa fina cuadrada o rectangular de plástico.

ENGINE: se usa en el lenguaje de los videojuegos para definir el tipo de entorno gráfico de un juego.

ESCENAS: conjunto de planos que forman parte de una misma acción: también, ambiente dentro de un espacio y un tiempo concretos

FOREGROUND: es el primer plano en los videojuegos, es decir, el que contiene al personaje, los enemigos y los objetos especiales.

FRAME: es el equivalente digital del fotograma físico, donde la imagen está codificada electrónicamente. Un segundo de vídeo está compuesto de diversos números de frames, según la norma de grabación. En la norma PAL, utilizada en la mayoría de países europeos, un segundo contiene 25 frames. En la norma NTSC, estándar norteamericano, un segundo contiene aproximadamente 30 frames.

FPS, FRAMES PER SECOND: mide la velocidad de una animación en cuadros por segundo.

FLICKER: parpadeo que se produce en la pantalla al dibujar cuando se esta efectuando el refresco vertical

GAME PAD: dispositivo para control de videojuegos equipado con una palanca y/o botnes con forma de cruz diseñado para mover con el dedo pulgar y con una zona de varios botones generalmente a la derecha.

GDI, GRAPHIC DEVICE INTERFACE: componente del núcleo de Windows para hacer graficas, con esto se dibujan las ventanas en el sistema

GRI, GOMEZ RUIZ IMAGEN: formato de imagen creado para el videojuego Fantasía Mitológica Colombiana, permite almacenar varios fotogramas en una sola imagen con su respectivas coordenadas en X e Y, las medidas en píxeles y sonidos asociados.

JOYSTICK: es un dispositivo de entrada que es utilizado, comúnmente en juego de consola o PC. Literalmente, palanca de juegos. Usado para mover uno o varios objetos en pantalla. Posee varios botones para acciones específicas y de una palanca para realizar los movimientos.

JUGABILIDAD: es la capacidad del videojuego para permitir la interacción para interactuar con el usuario y realizar distintas opciones.

MFC, MICROSOFT FOUNDATION CLASSES: una serie de funciones ya creadas para que se puedan utilizar con ventanas, botones, entre otros al crear programas para Windows, en el lenguaje C++.

MIDI, MUSICAL INSTRUMENT DIGITAL INTERFACE: estándar para la transmisión de información entre un instrumento y un ordenador. Un formato de datos de sonido.

MOUSE: es un periférico de computador, generalmente fabricado en material plástico, que se puede considerar, al mismo tiempo, como un dispositivo de entrada de datos y de control, dependiendo del software que se este utilizando en determinado momento.

MPG: desarrollado por el Motion Pictures Expert Group, Mpg es un sistema de compresión de vídeo que permite la codificación digital de imágenes en movimiento combinadas con texto, gráficos y sonidos.

MULTIJUGADOR: es un modo de juego para Computadores y Videojuegos en el cual varias personas pueden jugar el mismo juego al mismo tiempo. A diferencia de otros juegos de computador y videojuegos que son desarrollados para un único jugador debido a las limitaciones del sistema.

NINTENDO: es una empresa japonesa fundada en 1889 por Fusajiro Yamauchi para producir hanafuda (cartas de juego japonesas). Actualmente se dedica a la producción de software y hardware para videojuegos. Las oficinas centrales de la empresa se encuentran en Kyoto, Japón.

OFFSET: es la cantidad de bytes que hay desde el principio del stream, hasta donde empiezan este elemento.

PARALLAX SCROLLING: es una técnica especial del movimiento en sentido vertical y horizontal en los gráficos de escenarios de un videojuego. Esta técnica da a un videojuego 2D una mayor sensación de profundidad y de inmersión creando la ilusión de un ambiente 3D. Ser utiliza haciendo que los movimientos de la capa de fondo sean más lentos que los de la capa de primer plano, se necesitan varias capas para crear una ilusión de la profundidad.

POO, PROGRAMACIÓN ORIENTADA A OBJETOS: es una metodología de diseño de software y un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (es decir, datos) y comportamiento (esto es, procedimientos o métodos). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que se comunican entre ellos para realizar tareas, esto difiere del lenguaje en el cual se desarrolle.

PIXEL: abreviatura de la expresión inglesa "Picture Element", es la más pequeña unidad de medida de una imagen visualizada en la pantalla. La calidad de una imagen depende del número de píxeles por pulgada que la constituyen.

RAREWARE: más conocida como, Rareware, es una compañía dedicada al desarrollo de videojuegos. Fue fundada en 1985 por los hermanos Tim y Chris Stamper. Rare Ltd, fue una licenciataria de juegos para las plataformas de juego Nintendo durante varios años.
En el año 2002 fue adquirida por Microsoft.

RENDER: término que se refiere a la generación de un frame o imagen individual. El render consume potencia de cómputo debido a los cálculos necesarios para dibujar los frames según el número de objetos, luces y efectos en la escena y según el punto de vista de la cámara

RGB: es el acrónimo inglés Red, Green, Blue (Rojo, verde, Azul). Simboliza un sistema de colores, en el cual es posible representar 256 colores mediante una combinación de tres valores hexadecimales (uno por el rojo, otro por el verde y un último por el azul), en esta notación el valor 0x00 representa la ausencia del color en cuestión y el valor 0xff representa el mayor nivel del color actual posible. Viendo esto, el color negro (ausencia total de color) se representaría mediante 0x000000.

RUP, RATIONAL UNIFIED PROCESS: el proceso unificado racional es un método de diseño iterativo del software creado por la compañía Rational Software Corporation, ahora división de la IBM. Describe cómo desplegar el software que usa con eficacia técnicas comercialmente probadas. No es un proceso, es un marco o un meta-modelo de proceso. Abarca una gran cantidad de diversas actividades que se adaptan en el sentido de seleccionar solamente las características necesarias para un proyecto particular del software, en vista de su tamaño y tipo.

SPRITES: son una categoría de mapa de bits dibujados en la pantalla del computador. Normalmente son pequeños y parcialmente transparentes o con color de transparencia, dejándoles así asumir otras formas a la del rectángulo. Típicamente, los sprites son usados en videojuegos para crear los gráficos de los protagonistas. Generalmente son utilizados para producir una animación, como un personaje corriendo, alguna expresión facial o un movimiento corporal.

SQUARE SOFT: conocida internacionalmente como Squaresoft, es una compañía japonesa de videojuegos, creada en 1983 como parte de una firma de desarrollo de software, llamada Denyuusha.

STREAM: técnica usada para transmitir archivos de los multimedia por medio de paquetes, esto permite empezar a reproducir el archivo con los primeros paquetes, sin necesitar de esperar a que todos los datos se hayan descargado.

TILE, BALDOSA: es un “sprite” del escenario. La reunión de varios tiles conforman una imagen que se utiliza ya sea en las capas del foreground o del background.

WMA, WINDOWS MEDIA AUDIO: es un formato de compresión de audio con pérdida propiedad de Microsoft.

WMV, WINDOWS MEDIA VIDEO: es un formato que reúne varias tecnologías para video stream, desarrollado por Microsoft, forma parte de Windows Media Framework.

WAV, SIGLA DE WAVE: es un formato de audio digital normalmente sin compresión de datos, desarrollado y propiedad de Microsoft y de IBM que se utiliza para almacenar sonidos en el PC. El formato toma en cuenta algunas peculiaridades de la CPU Intel, y es el formato principal usado por Windows

XML, EXTENSIBLE MARKUP LANGUAGE: una versión redefinida de SGML. Permite personalizar las etiquetas que describen la presentación y el tipo de los elementos de datos. Es muy útil para los sitios web que mantienen grandes volúmenes de datos y para una intranet.

XSD, XML SCHEMA DEFINITION: un lenguaje utilizado para describir la estructura de un documento de XML. XSD se utiliza para definir clases que serán utilizadas para crear instancias de documentos XML que se conformen como un esquema.

VIDEOJUEGO: es un juego integrado por un universo virtual controlado por computador donde los jugadores pueden interactuar recíprocamente con dicho universo y también con otros jugadores para alcanzar una meta o un conjunto de metas.

UML, UNIFIED MODELLING LANGUAGE: lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software más conocido en la actualidad; aún cuando todavía no es un estándar oficial, está apoyado en gran manera por la OMG.

RESUMEN EJECUTIVO

El proyecto pretende llevar a cabo la creación de un videojuego en dos dimensiones donde se encuentran involucrados elementos de la mitología colombiana, sin que el objeto principal del videojuego sea la difusión de la misma, el objeto principal del proyecto es crear un producto de entretenimiento.

Llevar a cabo un proyecto de esta índole resulta un desafio desde el punto de vista académico y de ingeniería ya que la documentación existente al respecto es escasa en el entorno latinoamericano y son pocos los experimentos que se han realizado en este campo y aun mas escasos los que han sido realizados de manera exitosa. Por estas razones, entre otras tantas, definir con exactitud el curso de un proyecto de esta índole resulta complicado ya que no existe un marco de referencia para realizar las estimaciones respecto a los diversos procesos que son necesarios.

No existe una metodología pública creada expresamente para crear videojuegos, así que parte primordial e importante del desarrollo del proyecto consiste en definir una metodología de trabajo y un conjunto de diseños propios que permitan cubrir las necesidades inmediatas y que sea lo suficientemente flexible para cubrir las necesidades que se puedan generar durante el desarrollo del proyecto.

La investigación y la creatividad deben ser ampliamente explotadas, la gran cantidad de situaciones, de problemas complejos y de los niveles de dependencia que se presentan en todas las variables involucradas (hardware, guión, diseño artístico etc., desarrollo) demandan de un trabajo en equipo cohesionado y en permanente comunicación.

Hay varios ‘flancos de ataque’ que debe tener en cuenta el proyecto, primeramente desde el punto de vista de definición donde se ven involucradas tareas como:

- Visión del Juego
- Enfoque Técnico
- Objetivo Principal
- Etc.

Seguidamente se establecen parámetros relativos al guión de trabajo:

- Diseño de una historia
- Guión por escenas
- Descripción de cada uno de los personajes involucrados(caracterización)
- Etc.

Se establece la interfaz que se le dará al usuario y el modo de control que tendrá sobre el personaje u objeto que une al usuario con el juego entre muchos otros. Una vez atacado

este flanco pueden comenzar los demás, generalmente en un proceso paralelo que permite hacer afinamientos sobre la marcha.

Otro ‘flanco de ataque’ es el artístico donde se inicia una importante tarea que es la de transformar todos los conceptos del primer flanco en bocetos y diseños reales y medibles a través de dibujos artísticos de los personajes y escenarios, dibujos que irán madurando a lo largo del proyecto hasta convertirse en las animaciones y escenas el juego. En este flanco también se involucran los sonidos, menús, videos y música utilizada en el juego.

El ‘flanco de diseño’ es crucial para poder iniciar las etapas de desarrollo pues de la calidad y proyección dada a cada uno de los esquemas generados depende en gran parte que los desarrollos funcionen y se acoplen a las necesidades que puedan surgir sobre la marcha.

La investigación es uno de los flancos mas explotados, ya que gracias a esta se logro llevar la totalidad del proyecto a ser un 95% autoría propia, entre los principales logros se destacan la creación formatos propios de archivos de imágenes para personajes y para escenas, un ciclo de juego propio, y una serie de manejadores de objetos, codificadores y variedad de otros elementos que han dado pie a poder afirmar que se ha creado una metodología propia que ha llevado a la creación de un motor de desarrollo para videojuegos bastante robusto teniendo en cuenta las limitaciones de recursos económicos, de personal y de tiempo existentes.

Finalmente el flanco de los resultados, el de desarrollo, que no es más que la materialización de los resultados de los otros flancos, sobra mencionar que es la etapa mas difícil de todas ya que allí se desarrolló el motor y todas las capas que dependen del mismo y se tuvo que afrontar la complejidad de los dispositivos de hardware, el seudo paralelismo, la integración de todos los elementos multimedia (audio, video, dibujo etc.) los diferentes árboles de herencia y el diseño del sistema integrado de menús (ventanas) .

Todo esto para producir finalmente un videojuego, pero el resultado mas significativo es haber producido un motor de desarrollo que a la larga es capaz de soportar un sinnúmero de juegos de diferentes estilos, motor que se espera en un futuro sea la base sobre la cual se montara un producto mucho mas comercial y de mejor calidad.

PALABRAS CLAVE

Videojuego, multimedia, entretenimiento, C#, .NET, DirectX, UML, interacción, diversión, desarrollo, software, tecnología, gráficos, 2D, joystick, audio, video, sonido, mitología, animación, juego.

INTRODUCCIÓN

El desarrollo de las tecnologías de entretenimiento en Colombia es realmente pobre en especial en lo relacionado al tema de los videojuegos, este campo es uno de los más exigentes en el ámbito de la ingeniería en muchos niveles diferentes, en nuestro país son muy pocos los avances que se han hecho al respecto debido a varios aspectos:

- El mercado actual posee un alto nivel de competencia
- No existe espíritu investigativo ni entidades o instituciones dispuestas a financiarlo
- El conocimiento técnico y logístico y la disposición académica para llevar a cabo un proyecto de esta índole es realmente escaso.

Con la investigación y los avances realizados en este proyecto es posible comenzar a abrir el camino para el desarrollo de videojuegos en diferentes sectores académicos y de la industria Colombiana.

La gran mayoría de los proyectos para creación de videojuegos se han enfocado prácticamente al desarrollo de software educativo, el cual es mucho más sencillo en cuanto a manejo de periféricos, multimedia y su presentación por lo general es muy parecida a la de una enciclopedia electrónica, los temas tratados son muy diversos, comidas, bebidas, historia, geografía y matemáticas son los más comunes, la interactividad, con excepción del software desarrollado para discapacitados, posee un nivel muy bajo. Lejos de ejecutar un proyecto con estas características, la intención de este proyecto es involucrar al usuario en la temática del juego, motivarlo a descubrir nuevos niveles y a aprender sobre la mitología Colombiana, la intención primordial es brindar diversión.

El desarrollo de un videojuego demanda muchos recursos invertidos en tiempo, diseño del motor, diseño del guión, investigación, programación de motor y de juego, diseño de graficas, efectos especiales, escenarios, música etc. y mucho tiempo. A nivel comercial los videojuegos son elaborados por equipos que por lo general superan las 200 personas, como es el caso de los videojuegos desarrollados por grandes compañías como Nintendo, RareWare, Square Soft, entre otras. Estas razones han hecho del videojuego una empresa poco o nada productiva en países subdesarrollados, dada la alta inversión que se hace necesaria a nivel tecnológico, académico, artístico e investigativo no hay empresas dedicadas a este negocio en Colombia, los ingenieros desarrolladores han optado por reducirse en su gran mayoría al desarrollo de sistemas de información para empresas los cuales son proyectos comunes hoy en día y en el mas cercano de los casos se han dado desarrollos de juegos educativos o aplicativos multimedia los cuales siempre son por demás pobres comparados con un videojuego de entretenimiento.

Teniendo conocimiento de las restricciones y las exigencias de un proyecto con estas características se ha delimitado el alcance del proyecto a lo que seria el desarrollo de una primera fase de construcción del videojuego dejando el camino abierto a las mejoras consideradas convenientes por posteriores investigadores interesados.

1. OBJETIVOS

1.1. OBJETIVO GENERAL

Construir un videojuego RPG* basado en el desarrollo de una historia animada donde se den a conocer las características geográficas y míticas de Colombia.

1.2. OBJETIVOS ESPECÍFICOS

- Crear un guión para el desarrollo del juego basado en la mitología colombiana.
- Diseñar un entorno de desarrollo para historia es decir escenarios, música de fondo, diseño de objetos, espacio y tiempo de la acción.
- Diseñar cada uno de los personajes a nivel gráfico y de acción, entendiendo como esto, su caracterización gestual, sus movimientos característicos y su comportamiento general.
- Diseñar una interfaz de comando entre el usuario y el videojuego (jugabilidad).
- Programar la historia, personajes, escenarios, interfaz de comando, los elementos multimedia; juego completo.

* RPG es la sigla en ingles de Roll Playing Game.

2. METODOLOGÍA

Al tratarse de un proyecto que involucra diferentes aspectos como el artístico, el administrativo y el de desarrollo, es necesario plantear metodologías diferentes para cada uno de los grupos de procesos.

En primer lugar se describirá una metodología administrativa ya que sirve de apoyo para la realización de las otras dos.

2.1. METODOLOGÍA PARA LABORES ADMINISTRATIVAS

Pretendiendo dar una mayor continuidad al desarrollo del proyecto se han programado desde su inicio hasta su finalización una serie de reuniones semanales o quincenales, de las cuales se derivan las actas pertinentes como documentos de constancia y de compromiso para cada uno de los participantes.

Estas reuniones son el punto de partida para la realización frecuente de un seguimiento al desarrollo de las actividades plantadas en el cronograma de actividades, de este modo se pueden enunciar las principales guías para el desarrollo de esta metodología:

- Seguimiento permanente
- Comunicación de resultados
- Toma conjunta de decisiones
- Publicaciones Web (foros, comunidades de desarrollo, página Web del proyecto), para la búsqueda de soluciones o alternativas más eficientes.
- Orientar todas las fases iniciales del proyecto a la realización de un documento de diseño de videojuego[*].
- Fomentar y supervisar de manera conjunta la calidad en cada uno de los procesos involucrados en el desarrollo del videojuego.

2.2. METODOLOGÍA PARA ACTIVIDADES ARTÍSTICAS

Este conjunto de actividades exigen mucho a nivel artístico, muchos de los elementos involucrados en el videojuego no es posible que sean diseñados por ingenieros de sistemas, razón por la cual esta metodología esta orientada al trabajo interdisciplinario con profesionales[**] del área de publicidad y del diseño gráfico que poseen el talento y las destrezas necesarias para las labores mas exigentes.

---

[*] DOCUMENTO de diseño, comunidad de programadores de videojuegos http://www.gamedev.net/ 1 de septiembre de 2004.

[**] Profesionales se debe entender personas que se desempeñan en el área y no necesariamente deben tener un titulo profesional que los acredite.

La metodología parte de los siguientes aspectos:

- Elaborar a nivel conceptual todos los dibujos necesarios
- Diseñar todos los elementos que sean posibles dentro de las posibilidades de los miembros del grupo
- En caso de que algunos elementos demanden un nivel superior de diseño recurrir a la búsqueda de profesionales que quieran participar en el proyecto de manera voluntaria y sin ánimo de lucro o en su defecto proceder a buscar personas que realicen los diseños necesarios esperando una compensación de índole menor por ello.
- Utilizar sonidos y música de uso libre a través de la Internet buscando siempre que esta se ajuste a las necesidades del proyecto lo más posible.
- Editar los sonidos y la música utilizada por medio de herramientas avanzadas de definición, construcción, edición y optimización de sonidos.
- Realizar ajustes menores a los dibujos y sonidos durante el transcurso del proyecto con el fin de optimizar los procesos de desarrollo del software.

2.3. METODOLOGÍA PARA LA CONSTRUCCIÓN DEL SOFTWARE

Teniendo en cuenta que no existe una metodología (pública) previamente diseñada para la construcción de videojuegos se ha desarrollado una metodología propia que permite y procura la integración entre los diferentes módulos maximizando la cohesión entre todos los módulos del juego y minimizando el acoplamiento entre capas[*].

El desarrollo del software del proyecto sigue un proceso iterativo e incremental, el cual en un nivel de abstracción mayor se puede contemplar como un sistema basado en capas de funcionalidad donde las capas superiores son cliente de las capas inferiores del juego, razón por la cual el énfasis y diseño mas fuerte del videojuego se encuentra en las capas inferiores y medias.

Esta forma de trabajo permite fácilmente cambiar algunas implementaciones de los módulos por otras diferentes en el futuro sin demandar cambios mayores en los módulos ya creados, un ejemplo que clarifica esto es la siguiente situación:

Se debe cambiar el motor del videojuego para que funcione usando APIS de manejo en 3 dimensiones para así aprovechar las implementaciones existentes hoy en día a nivel de aceleración por hardware y así hacer mucho mas eficiente la parte gráfica del motor.

Recrear el módulo gráfico del motor, es decir convertirlo de manejo de un API para trabajo gráfico en dos (2) dimensiones por software a una arquitectura de trabajo en tres (3) dimensiones por hardware no implica nada diferente que reprogramar el únicamente módulo gráfico ya que los demás módulos de las capas superiores desconocen el funcionamiento de la API y son en general solo dependientes de la interfaz brindada por el motor, no deberían existir cambios en los demás módulo y de ser así estos deben ser mínimos.

---

[*] Ver este documento, Cáp. 18. MANEJADOR DE ESCENAS

La metodología procura el desarrollo de los módulos que cumplan con los siguientes aspectos:

- Todo el esquema (con excepción del sistema de menús por obvias razones) va direccionado a una implementación POO utilizando los recursos de las herramientas seleccionadas. (.NET Framework, DirectX 9.0 Managed)
- Todos los módulos deben estar 100% documentados siguiendo el estándar del .NET Framework 1.0
- Elaboración de los modelos fundamentales para la creación del motor, estos modelos son persistentes y no cambiarán en el transcurso del desarrollo.
- Elaboración superficial de los elementos de las capas superiores al motor debido a que en el momento del diseño es imposible determinar las maneras y el alcance propio de cada uno de los submódulos.
- Determinar antes de la construcción de cada módulo las entradas y salidas esperadas inicialmente por cada uno de ellos.
- Utilizar árboles complejos de herencia e implementación de interfaces (para simular la herencia múltiple) con el fin de minimizar el tiempo de desarrollo reutilizando la mayor cantidad posible de código.
- Implementaciones altamente genéricas en las primeras capas del motor.

Durante el desarrollo de este documento se profundiza más en estos contenidos según sea pertinente en cada capítulo.

3. MARCO REFERENCIAL

3.1. MARCO CONCEPTUAL

Un videojuego por ser un modelo hombre- maquina necesita de un serie de prerrequisitos para poder ser llevado a cabo.

A grandes rasgos los conceptos más relevantes para el desarrollo de videojuegos son:

- Realización de modelos de casos de uso en UML
- Conocimientos en lenguajes de programación y manipulación de librerías gráficas.
- Diseño, desarrollo e implementación de un Motor que soporte las peticiones básicas que se realicen a la maquina.
- Diseño, desarrollo e implementación de herramientas para administración de personajes, escenarios y multimedia.
- Diseño, desarrollo e implementación de una interfaz gráfica para la interacción del usuario con la aplicación.
- Diseño, Animación básica en 2D, Utilización de Herramientas de manipulación de Gráficos.

3.1.1. Situación Actual

3.1.1.1. Multinacionales. En el momento, multinacionales del entretenimiento interactivo, tales como Nintendo Square Soft, Microsoft, EA Games, Capcom, entre otras, tienen acaparado el mercado mundial de videojuegos. La mayoría de sus títulos son lanzados para sean compatibles con las distintas plataformas existentes en el mercado como lo son el Game Cube de Nintendo, Play Station 2 de Sony, Xbox de Microsoft, PC, y las plataformas portátiles etc. que disponen de los últimos avances tecnológicos para soportarlos.

3.1.1.2. Comunidades y Empresas Independientes. Por otro lado esta la comunidad de desarrolladores independientes, personas que se reúnen en grupos o que en forma individual realizan la investigación pertinente para empezar el desarrollo de un videojuego. Son muy pocas las personas que hacen públicos sus avances y más aún sus proyectos.

En estas comunidades se han desarrollado ya motores gratuitos para creación de juegos de estrategia tipo Age of Empires y Age of Mythology, el problema de dichos motores, son las limitantes en cuestión de interacción con la maquina, diseño de escenarios y personajes entre otros.

La información acerca de la metodología de desarrollo de videojuegos es prácticamente inaccesible debido a que solo ha sido realizada por las empresas desarrolladoras de videojuegos y no esta disponible al público. Mientras que la documentación que se

encuentran en los foros de las comunidades de desarrollo, solo enuncian aspectos básicos para representación de imágenes en pantalla y animaciones simples.

A continuación se hace mención a las comunidades de desarrollo de videojuegos más importantes.

3.1.1.2.1. TelePortMedia S.A. Es una empresa nacida a mediados del año 2002, ubicada en San José de Costa Rica en Centroamérica.

Está dedicada principalmente en los campos del desarrollo de videojuegos, presentaciones Multimedia, páginas web, desarrollo de software personalizado, etc.

TelePortMedia surgió principalmente por la necesidad de expandir la industria de los videojuegos en el área. Luego de haber comprobado que no existía ninguna empresa dedicada especialmente a estos campos en Costa Rica, nació la empresa.

3.1.1.2.2. Vjuegos. Es una comunidad virtual, que tiene como misión fomentar el interés, el aprendizaje y la práctica del desarrollo de videojuegos en los países de habla hispana; con una visión global de construir una comunidad experta en el área para el mejoramiento de la industria.

Este sitio cuenta con varios foros de discusión y una sala de Chat que te permitirá estar más en contacto con otros miembros de esta comunidad de desarrolladores de videojuegos, y así aprender de sus experiencias y conocimientos, además de poder analizar, discutir e intercambiar opiniones al respecto.

Además, se busca abrir espacios con personas que estén trabajando profesionalmente en la industria para conocer y aprender de las experiencias y conocimientos que han adquirido desarrollando los proyectos en los que se encuentren trabajando.

3.2. MARCO TEÓRICO

Para la realización del proyecto se evaluaron aspectos como el tipo de juego, las librerías gráficas, herramientas de manipulación de gráficos entre otras.

Para poder aprovechar más los recursos de la mitología colombiana se optó por utilizar el concepto de videojuego del rol, con el cual se puede crear una historia y un guión en los cuales se puedan incorporar dichos recursos tales como personajes, sitios geográficos reales, leyendas y mitos. Con otros tipos de juego, existirían inconvenientes tales como el incremento en la cantidad de recursos necesarios (Juegos de Aventura, Simulación), poca o nula utilización de elementos mitológicos (Juegos de Pelea, juegos de disparos) o por razones obvias su concepto no aplica (Juegos de Deportes).
Debido a que las plataformas de trabajo disponibles para el desarrollo son PC con sistema operativo Microsoft Windows XP se eligió utilizar la librería de gráficos Microsoft DirectX para poder interactuar de una forma más amigable con el sistema operativo además de su integración con el lenguaje de programación

C Sharp (C#) perteneciente a la suite Microsoft Visual Studio.Net 2003.

3.2.1. Antecedentes. La Historia de los Videojuegos constituye el contexto histórico del proyecto, a continuación se muestra el origen de los videojuegos.

Muy lejos del concepto de un sistema de videojuegos comercial, un físico intenta en 1958 hacer la gira pública en su laboratorio un poco más excitante a sus aburridos visitantes, es lo que muchos consideran el precursor de los videojuegos. Trabajando en el Laboratorio Nacional de Brookhaven, un laboratorio de investigación nuclear americano en Upton, Nueva York, William A. Higinbotham nota que el público que asiste en otoño a las jornadas de puertas abiertas anuales (que se realizan para mostrarle al público lo seguro del trabajo que allí se desarrolla), está aburrido con la simple muestra de fotografías y equipo estático. Educado en la Universidad de Cornell y con un doctorado en Física, Higinbotham llega al BNL de Los Alamos y del Proyecto Manhattan, siendo testigo presencial de la primera detonación de una bomba atómica. Este fumador empedernido, chistoso y reconocido jugador de pinball, quiere desarrollar una exhibición que entretenga a los visitantes y al mismo tiempo les enseñe. Su idea fue entonces la de utilizar una pequeña computadora analógica del laboratorio para graficar y mostrar en un osciloscopio la trayectoria de una pequeña pelota en movimiento, con la que el público pudiera interactuar.

Trazar la trayectoria de proyectiles, fue una de las especialidades de las computadoras en este momento, la otra sería la criptografía. De hecho, la primera computadora electrónica fue desarrollada para trazar la trayectoria de las miles de bombas que se dejaron caer en la Segunda Guerra Mundial.

Como cabeza de la División de Instrumentación de Brookhaven, y pioneros en la construcción de complicados dispositivos electrónicos como los detectores de radiación, no fue ningún problema para Higinbotham, quién junto con Robert V. Dvorak (Especialista Técnico que realmente ensambla el dispositivo), crear en tres semanas el sistema de juego que bautizan como Tennis for Two, y que debuta junto a otras exhibiciones en el gimnasio de Brookhaven en la siguiente jornada de puertas abiertas en octubre de 1958. En este rudimentario juego de tenis visto de lado, la pelota rebota sobre una larga línea horizontal al fondo del osciloscopio, y hay una pequeña línea vertical pequeña en el centro que representa la red. Dos cajas cada una con un dial y un botón son los controles... los diales afectan el ángulo de la trayectoria de la pelota y los botones "golpean" la pelota al otro lado de la pantalla. Si el jugador no ajusta correctamente el ángulo de la pelota esta colisiona con la red. También dispone de un botón para resetear el juego, haciendo reaparecer la pelota delante de cualquier lateral de la pantalla listo para jugar de nuevo. No dispone de un contador de puntos, y se muestra en la gloriosa y endeble pantalla de fósforo monocromático de 5" de un osciloscopio, pero resulta un enorme éxito para todos quienes visitan la muestra. Las personas hacen cola durante horas para poder jugar.

El juego reaparece para las jornadas de 1959, y las modificaciones incluyen a un monitor más grande para desplegar la acción, y las condiciones de gravedad configurables para mostrar lo que sería jugar al tenis en otro planeta. Después de esta última aparición, el sistema es desmantelado y sus partes se dispusieron para otros usos. William no comercializa ni patenta su invención, pensando obviamente que no es de ningún valor.

Pero su testimonio es requerido años después durante la batalla legal por quebrar la patente del videojuego Magnavox obtenida a través del desarrollo del sistema de videojuego Odyssey.

El análisis del verdadero origen del "Tenis para Dos" es la intriga de muchos, por un lado; el Laboratorio Nacional de Brookhaven y David Ahl en ambos basó Higinbotham su experimento. Ahl jugó infinidad de veces durante las giras al laboratorio en su juventud y luego fundó Creative Computing unas de las primeras y más renombradas revistas de la industria. Por otro lado está Ralph Baer, quien obtiene la patente del primer videojuego hogareño (home videogame) que se convertiría en el Odyssey. Durante muchos años de litigio defiende su patente, aunque sin dudas el se basó en el trabajo de Higinbotham, aunque él lo describiera como una demostración simple de balística, basada en un osciloscopio. Desgraciadamente, el hombre al centro de esta controversia no puede defenderse: William Higinbotham, dueño de 20 patentes sobre circuitos electrónicos, fallece el 10 de noviembre de 1995, a la edad de 84 años.

Hacia finales de 1961 existe en el MIT un grupo que se llama así mismo "Tech Model Railroad Club", y las actividades del club incluyen encendidos debates sobre las novelas de E.E "Doc" Smith, considerado el pionero literario del género de SF (Ciencia Ficción). Ellos fantasean con una película que muestre los efectos especiales descritos en las novelas, que contienen detallados relatos de inmensas batallas de naves espaciales interestelares. Para esa época se está volviendo obsoleto el (gigante) mainframe de transistores TX-0 y el MIT busca una nueva computadora que le haga compañía: el relativamente esbelto PDP-1 de Digital Equipment Corporation PDP-1. En las anteriores jornadas de puertas abiertas (como las realizadas en Brookhaven, orientadas al público en general) para el TX-0 consistió en un laberinto mostrado en una pantalla de focos, representando los movimientos de un ratón y el venerable Tic-Tac-Toe.

Con motivo de la nueva adquisición la universidad decidió realizar en las próximas jornadas de puertas abiertas un programa que aprovechará al máximo las posibilidades del PDP-1. Se presentaron entonces Wayne Witanen y J. Martin Graetz, junto con Steve Russell de apenas 25 años, y especialmente conocido por sus compañeros como "Slug" debido a su tendencia a aplazar. Recordando las batallas espaciales narradas por E.E. Smith, desarrollan la idea de dos naves espaciales con combustible ilimitado, enfrentadas en un duelo de misiles frente a frente. El programa se convierte en el Space war, el primer videojuego interactivo del mundo, con Russell como principal programador.

Dos naves espaciales llamadas wedge y needle (aguja y cuña), debido a la representación gráficas que tienen en el monitor. Otros programadores le brindan ayuda a Russell, incluida la rutina de seno-coseno de Alan Kotok, y el realista campo estelar de fondo llamado Expensive Planetarium de Peter Samson. Dan Edwards desarrolla los efectos de gravedad del juego, centrados alrededor de un sol luminoso que atrae tanto a las naves como a los misiles. Graetz desarrolla el efecto de Hyperspace, para que cualquier jugador pudiera desaparecer de la pantalla y reaparecer de forma aleatoria en otro lugar, con lo que no siempre salvaba su situación.

El juego se termina en la primavera de 1962, y ocupa la enorme cantidad de 9K. Su exposición en la anual Science Open House del MIT tiene un éxito increíble, y para limitar el tiempo de juego, se le introduce un contador que se integra a los interruptores utilizados para el manejo del juego. Semejante suceso provoca un revuelo en la comunidad de usuarios de computadoras, y las copias del juego se distribuyen rápidamente en todos los establecimientos educativos de los Estados Unidos (atravesando de una punta a la otra la ARPAnet, la precursora de Internet).

DEC utiliza el programa para demostrar las posibilidades de su PDP-1 a los posibles clientes y lo incluye de forma gratuita en para cada sistema instalado. Una vez más, al igual que William Higinbotham, Russell no intenta patentar su trabajo, posiblemente porque el sistema del Space war se ejecutaba en un equipo del tamaño de un refrigerador y costaba US$120,000. Debido a su estatus de dominio público, el juego terminará siendo un de los concepto más copiados de la industria de los videojuegos, desde numerosas versiones arcade como Computer Space y Space Wars hasta consolas hogareñas como el Atari VCS y el Odyssey2. Pero el Space war hizo doblemente historia, sus adictos del MIT diseñaron el primer joystick para reemplazar los anteriores interruptores de control.

Mientras asiste a la universidad de Utah para obtener su licenciatura en de ciencias, el joven estudiante llamado Nolan Bushnell pasa la mayoría del tiempo jugando al Space war de Russell en la mainframe PDP-1 de la universidad, uno de los tres establecimientos educativos de Estados Unidos con el privilegio de poseer un monitor para mostrarlo. El interés en los videojuegos hace que en 1972, Nolan Bushnell funde una compañía llamada Atari, y comercialicé el juego con el nombre de Pong. Pong trataba de un juego sencillo de tenis de mesa, compuesto por dos barras que simulaban las raquetas y un cursor que, moviéndose, atravesaba la pantalla haciendo las veces de pelota, apareció en un principio en una máquina denominada Arcade.

A partir del éxito de Pong se desencadeno el interés de muchas compañías por desarrollar videojuegos tanto para Arcade, Consolas y para PC.

3.2.2. Tipos de videojuegos

3.2.2.1. Acción. En estos videojuegos existen uno o varios personajes centrales que pueden ser elegidos por el usuario, estos juegos están divididos en varias escenas o etapas en las cuales se debe cumplir distintas misiones u objetivos para así terminar el juego.

3.2.2.2. Deportes. El objetivo de este tipo de Videojuego es emular la forma en que se practican distintos deportes, el usuario debe controlar al (los) deportista(s). Estos videojuegos se caracterizan por ser multiusuario, permiten que varios usuarios interactúen entre ellos utilizando una sola aplicación.

3.2.2.3. Estrategia. La definición de este tipo de videojuego radica en la habilidad que debe tener el usuario para comandar varios personajes distintos, cada uno de ellos tiene distintas habilidades que serán útiles a lo largo de cada etapa del juego.

3.2.2.4. Roll o RPG (Roll Playing Games). Los videojuegos de rol tienen por objetivo hacer que el usuario personifique al protagonista de la historia, este tipo de videojuego tiene la misma base ya que los protagonistas secundarios están ligados de una u otra forma a la historia viajando en grupo de un lugar a otro, en los videojuegos de rol el protagonista puede interactuar con los habitantes del mundo en el que se desenvuelve la historia, son videojuegos cuya historia es épica, donde predominan la magia, los hechizos, los caballeros y monstruos.

3.2.2.5. Plataforma. Los videojuegos de plataforma deben su nombre a la percepción que tiene el usuario al jugarlos. Básicamente un videojuego de plataformas se desarrolla en un ambiente 2D donde la perspectiva del personaje principal es de lado o en una vista ¾, al avanzar por los escenarios el personaje debe saltar desfiladeros, escalar árboles o muros.

3.2.2.6. Aventura. En este tipo de videojuego el personaje generalmente tiene que avanzar a través de un gran mundo, en donde la historia es muy importante y juega un rol característico; e ir adquiriendo habilidades y resolviendo acertijos y problemas para cumplir el objetivo final del juego. Tiene elementos de videojuego RPG y generalmente es en una perspectiva de tercera persona.

3.2.2.7. Juegos de disparo. Son videojuegos que ofrecen la posibilidad de disparar a diversos blancos, muchos de estos videojuegos ofrecen compatibilidad con periféricos especializados, la perspectiva es en primera persona.

3.2.2.8. Simuladores. Son videojuegos que pretenden transmitir al usuario una sensación lo más cercana a la realidad. Las simulaciones permiten al usuario pilotear aviones, embarcaciones o autos de fórmula uno hasta desenvolver el papel de un alcalde, entre otros.

3.2.3. Especificación de librerías gráficas

3.2.3.1. Allegro (Allegro Low Level Game Routines). El videojuego puede definirse como un entorno informático que reproduce sobre una pantalla un juego cuyas reglas han sido previamente programadas. Como la literatura, el teatro o el cine, los videojuegos proponen la visita a mundos imaginarios, con el añadido de una interactividad que no puede ofrecer ningún otro espectáculo o arte.

En el desarrollo de videojuegos para incluir gráficos, sonido y sistemas de control efectivos se puede contar con librerías previamente programadas. Estas librerías evitan que los desarrolladores inviertan tiempo innecesario creando rutinas de frecuente uso.

Allegro es una librería para programadores de C/C++ orientada al desarrollo de videojuegos y programación multimedia, distribuida libremente, y que funciona en las plataformas: DOS, Unix (Linux, FreeBSD, Irix, Solaris), Windows, QNX y BeOS (la versión MacOS está en estado alpha). Es por lo tanto una librería portable, originalmente escrita por Shawn Hargreaves para el compilador DJGPP en una mezcla de C y ensamblador.

Según el suplemento de música del diccionario Oxford, Allegro es la palabra italiana para describir “rápido, vivo, brillante”. Además es un acrónimo recursivo de «Allegro Low Level Game Routines (rutinas de bajo nivel para videojuegos).

Tiene muchas funciones de gráficos, sonidos, entrada del usuario (teclado, ratón y joystick) y temporizadores. También tiene funciones matemáticas en punto fijo y coma flotante, funciones 3d, funciones para manejar archivos, archivos de datos comprimidos y una interfaz gráfica.

Detallando:

- Funciones gráficas
- Dibujo vectorial
- Polígonos.
- Sprites
- Soporte nativo de archivos
- Paletas de color
- Textos
- Drivers Gráficos
- Funciones de sonido
- Midi
- Wave
- Drivers de sonido

3.2.3.2. SDL (Simple Direct media Layer). También es considerada una librería no profesional a pesar de que con ella se han portado juegos comerciales desde Windows hacia Linux como el Civilización III. No obstante es muy recomendable utilizar esta librería para el que recién empieza porque además de ser simple de programar es rápida y tiene la posibilidad de trabajar con OpenGL para el manejo de aceleración gráfica. Algo importante a tener en cuenta es que es también multiplataforma y que se está trabajando para poder crear juegos para Playstation 2.

3.2.3.3. OpenGL. Es una interfaz software de hardware gráfico, es decir define las funciones que se pueden utilizar en una aplicación para acceder a las prestaciones de un dispositivo gráfico. Es un motor 3D cuyas rutinas están integradas en tarjetas gráficas 3D. Fue desarrollado por Silicon Graphics, Inc. (SGI) con el afán de hacer un estándar de

representación en 3D. Es compatible con prácticamente cualquier plataforma hardware así como con muchos lenguajes de programación (C, C++, Visual Basic, Fortran, Java).

3.2.3.4. Microsoft Directx. Microsoft DirectX es un conjunto de API multimedia incorporadas en los sistemas operativos Windows de Microsoft. Proporciona una plataforma de desarrollo estándar para los PC Windows desde la cual acceder a funciones especiales de hardware sin tener que escribir código específico para el hardware. Se introdujo por primera vez en 1995 y es un estándar aceptado de forma generalizada para el desarrollo de aplicaciones multimedia para Windows. DirectX desempeña un papel en muchas funciones como: el renderizado 3D, la reproducción de video, la captura de instantáneas o en movimiento, las aplicaciones de visualización de TV, interfaz con el joystick y el mouse, red para juegos con varios jugadores y muchos más.

3.2.3.4.1. DirectSound. La API de Microsoft DirectSound proporciona un vínculo entre los programas y las capacidades de mezcla y reproducción de sonido de un adaptador de audio. También permite la captura y reproducción de sonido de onda. DirectSound proporciona a las aplicaciones multimedia mezcla de baja latencia, aceleración de hardware y acceso directo al dispositivo de sonido. Proporciona estas características al tiempo que mantiene la compatibilidad con los controladores de dispositivo existentes.

3.2.3.4.2. Microsoft DirectMusic. La API de Microsoft DirectMusic es el componente musical de DirectX. A diferencia de la API de DirectSound, que captura y reproduce muestras de sonido digital, DirectMusic trabaja con datos musicales basados en mensajes que se convierten a audio digital por medio de la tarjeta de sonido o de su sintetizador de software integrado. Además de admitir entradas en formato Interfaz digital para instrumentos musicales (MIDI, Musical Instrument Digital Interface), DirectMusic proporciona a los programadores de aplicaciones la capacidad para crear pistas de sonido dinámicas y de inmersión que responden a la entrada del usuario.

3.2.3.4.3. Microsoft DirectInput. La API Microsoft DirectInput proporciona, para juegos y procesos, valores de entrada avanzados de joysticks y otros dispositivos relacionados, como el mouse, el teclado y otros dispositivos de juegos, como los dispositivos de juego con fuerza de respuesta. El nivel DirectX Media trabaja con el nivel DirectX Foundation para proporcionar servicios de alto nivel que permitan animaciones, transmisión de multimedia (transmisión y presentación de audio y vídeo a medida que se descarga de Internet) e interactividad. Igual que el nivel DirectX Foundation, el nivel DirectX Media constan de varios componentes integrados.

3.2.3.4.4. Microsoft Direct3D Retained Mode. La API Microsoft Direct3D Retained Mode permite el uso de gráficos avanzados, en tiempo real y tridimensional (3-D). Direct3D Retained Mode proporciona compatibilidad integrada con técnicas gráficas como jerarquías y animaciones. Direct3D Retained Mode se ha construido a partir de Direct3D Immediate Mode.

3.2.3.4.5. Microsoft DirectAnimation. La API Microsoft DirectAnimation proporciona integración y animación para diferentes tipos de multimedia, como imágenes bidimensionales, objetos tridimensionales, sonidos, películas, texto y gráficos vectoriales.

3.2.3.4.6. Microsoft DirectPlay. La API de Microsoft DirectPlay admite conexiones de juegos a través de un módem, de Internet o de una LAN. DirectPlay simplifica el acceso a los servicios de comunicación y proporciona una forma para que los juegos se comuniquen entre sí, independientemente del protocolo subyacente o del servicio en línea.

3.2.3.4.7. Microsoft DirectShow. La API Microsoft DirectShow reproduce archivos multimedia que se encuentran en archivos locales o en servidores de Internet, y captura flujos multimedia que provienen de dispositivos, como tarjetas capturadoras de vídeo. DirectShow reproduce audio y vídeo comprimido en diversos formatos, incluidos MPEG, audio-vídeo intercalado (AVI) y archivos WAV.

3.2.3.4.8. Microsoft DirectX Transform. La API Microsoft DirectX Transform permite a los programadores de aplicaciones crear, animar y modificar imágenes digitales. DirectX Transform trabaja tanto con imágenes bidimensionales (2-D) como tridimensionales (3-D) y se puede utilizar para crear programas independientes o complementos dinámicos para gráficos Web.

3.2.4. Matriz comparativa de librerías graficas

Tabla 1. Matriz comparativa de librerías graficas

	MULTIPLATAFORMA	EFICIENCIA	SONIDO	GRAFICOS	ENTRADA
Allegro	SI	3	3	SI	SI
SDL	SI	3	2	SI	SI
DirectX	NO	5	5	SI	SI

	2D	3D	CAPACIDAD 3D	EXTENSIBILIDAD	MAS USADO
Allegro	SI	SI	MEDIA	AMPLIA	2
SDL	SI	SI	MEDIA	MEDIA	1
DirectX	SI	SI	ALTA	AMPLIA	3

	LENGUAJES	COMPLEJIDAD	DOC	Estructrura	POPULARIDAD
Allegro	C, C++,	ALTA	BAJA	MEDIA	ALTA
SDL	C, C++,	MUY ALTA	BAJA	MEDIA	BAJA
DirectX	C, C++, Vbasic, C#, Delphi y otros más	ALTA	MEDIA	ALTA	ALTA

	OBSERVACIONES ADICIONALES
Allegro	Es muy popular por ser freeware, y opensource, pero solo soporta c y c++, su desempeño e nivel de sonido es un poco bajo, no es estructurado, difícil de entender.
SDL	Es freeware, y opensource, pero solo soporta c y c++, su no es estructurado, difícil de entender, no posee tan amplio soporte como las otras.
DirectX	Es la librería más completa que existe , soporta gran cantidad de lenguajes, y es extremadamente eficiente, es la más usada en el mercado, su única debilidad es que solo lo soportan sistemas operativos Microsoft.(Por el momento)

Fuente: Los Autores

3.2.5. Lenguajes De Programación.

3.2.5.1. Visual C# .Net 2003. Es una herramienta y un lenguaje de programación moderno e innovador que permiten generar software conectado a .NET para Microsoft Windows, Web y una amplia gama de servicios. Debido a su sintaxis familiar, similar a la de C++, a su entorno de desarrollo integrado de gran flexibilidad y a su capacidad para crear soluciones para una gran variedad de plataformas y dispositivos, Visual C# .NET 2003 facilita enormemente el desarrollo de software conectado a .NET.

Visual C# .NET está basado directamente en C++ lo cual hace que su entorno sea muy familiar con C++ y Java. Es un lenguaje de programación moderno e intuitivo orientado a objetos que ofrece mejoras significativas, incluido un sistema de tipos unificados, código "no seguro" que permite un control total al programador y nuevas construcciones de lenguaje muy eficaces y fáciles de entender para la mayoría de los programadores.
Visual C# .NET proporciona una funcionalidad óptima para racionalizar los procesos, incluidos:

- Compatibilidad con el diseño, el desarrollo y la implementación de servicios Web XML con rapidez.
- Diseñadores de formularios y controles visuales para crear aplicaciones basadas en Windows muy completas.
- Herramientas y servicios de diseño para crear eficaces soluciones de Microsoft .NET basadas en servidor.
- Herramientas de migración para convertir los proyectos basados en Java al entorno de desarrollo de Microsoft .NET.

3.2.5.1.1. Windows .Net Framework. Es el componente de Windows para crear y ejecutar la próxima generación de aplicaciones de software y servicios Web XML. Windows .NET Framework tiene las características siguientes:

- Es compatible con más de 20 lenguajes de programación diferentes.
- Se encarga de la mayor parte de la estructura necesaria para generar software, lo que permite a los programadores centrarse en el código lógico esencial para el negocio.
- Facilita más que nunca la creación, implementación y administración de aplicaciones seguras, sólidas y de gran rendimiento.

Windows .NET Framework se compone de Common Language Runtime y un conjunto unificado de bibliotecas de clases.

Common Language Runtime, Es responsable de los servicios en tiempo de ejecución, como por ejemplo, la integración de lenguajes, el cumplimiento de las normas de seguridad y la administración de la memoria, los procesos y los subprocesos. Además, CLR cumple una función en la fase de desarrollo, cuando ciertas características, como por ejemplo, la administración del ciclo de vida, la nomenclatura segura de tipos, la administración de excepciones entre lenguajes y los enlaces dinámicos, reducen la

cantidad de código que tiene que escribir el programador para convertir la lógica comercial en un componente reciclable.
Las bibliotecas de clases son funciones estándar, como las de entrada/salida, manipulación de cadenas, administración de seguridad, comunicaciones en red, administración de subprocesos, administración de textos y funciones de diseño de la interfaz de usuario.

Las clases de ADO.NET permiten a los programadores interactuar con los datos obtenidos en formato XML a través de las interfaces OLE DB, ODBC, Oracle y SQL Server. Las clases XML permiten la manipulación, búsqueda y conversión de objetos XML. Las clases ASP.NET son compatibles con el desarrollo de aplicaciones basadas en Web y de servicios Web XML. Las clases de Windows Forms son compatibles con la generación de aplicaciones cliente inteligentes basadas en escritorio.
En conjunto, las bibliotecas de clases ofrecen una interfaz de desarrollo común y coherente en todos los lenguajes compatibles con Windows .NET Framework.

3.2.5.2. Java. Se creó como parte de un proyecto de investigación para el desarrollo de software avanzado para una amplia variedad de dispositivos de red y sistemas embebidos. La meta era diseñar una plataforma operativa sencilla, fiable, portable, distribuida y de tiempo real. Cuando se inició el proyecto C++ era el lenguaje del momento, pero a lo largo del tiempo las dificultades encontradas con C++ crecieron hasta el punto en que se pensó que los problemas podrían resolverse mejor creando una plataforma de lenguaje completamente nueva. Se extrajeron decisiones de diseño y arquitectura de una amplia variedad de lenguajes como Eiffel, SmallTalk, Objetive C y Cedar/Mesa. El resultado es un lenguaje que se ha mostrado ideal para desarrollar aplicaciones de usuario final seguras, distribuidas y basadas en red en un amplio rango de entornos desde los dispositivos de red embebidos hasta los sistemas de sobremesa e Internet.

Java fue diseñado para ser:

- Sencillo, orientado a objetos y familiar: Sencillo, para que no requiera grandes esfuerzos de entrenamiento para los desarrolladores. Orientado a objetos, porque la tecnología de objetos se considera madura y es el enfoque más adecuado para las necesidades de los sistemas distribuidos y/o cliente/servidor. Familiar, porque aunque se rechazó C++, se mantuvo Java lo más parecido posible a C++, eliminando sus complejidades innecesarias, para facilitar la migración al nuevo lenguaje.
- Robusto y seguro: Robusto, simplificando la gestión de memoria y eliminando las complejidades de la gestión explicita de punteros y aritmética de punteros del C. Seguro para que pueda operar en un entorno de red.
- Independiente de la arquitectura y portable: Java está diseñado para soportar aplicaciones que serán instaladas en un entorno de red heterogéneo, con hardware y sistemas operativos diversos. Para hacer esto posible el compilador Java genera 'bytecodes', un formato de código independiente de la plataforma diseñado para transportar código eficientemente a través de múltiples plataformas de hardware y software. Es además portable en el sentido de que es rigurosamente el mismo lenguaje en todas las plataformas. El 'bytecode' es traducido a código máquina y ejecutado por la

Java Virtual Machine, que es la implementación Java para cada plataforma hardware-software concreta.

- Alto rendimiento: A pesar de ser interpretado, Java tiene en cuenta el rendimiento, y particularmente en las últimas versiones dispone de diversas herramientas para su optimización. Cuando se necesitan capacidades de proceso intensivas, pueden usarse llamadas a código nativo.
- Interpretado, multihilo y dinámico: El intérprete Java puede ejecutar bytecodes en cualquier máquina que disponga de una Máquina Virtual Java (JVM). Además Java incorpora capacidades avanzadas de ejecución multihilo (ejecución simultánea de más de un flujo de programa) y proporciona mecanismos de carga dinámica de clases en tiempo de ejecución.

3.2.5.2.1. Características de Java.

- Lenguaje de propósito general.
- Lenguaje Orientado a Objetos.
- Sintaxis inspirada en la de C/C++.
- Lenguaje multiplataforma: Los programas Java se ejecutan sin variación (sin recompilar) en cualquier plataforma soportada (Windows, UNIX, Mac...)
- Lenguaje interpretado: El intérprete a código máquina (dependiente de la plataforma) se llama Java Virtual Machine (JVM). El compilador produce un código intermedio independiente del sistema denominado bytecode.
- Lenguaje gratuito: Creado por SUN Microsystems, que distribuye gratuitamente el producto base, denominado JDK (Java Development Toolkit) o actualmente J2SE (Java 2 Standard Edition).
- API distribuida con el J2SE muy amplia. Código fuente de la API disponible.

3.2.5.3. C++. El comité para el estándar ANSI C fue formado en 1983 con el objetivo de crear un lenguaje uniforme a partir del C original, desarrollado por Kernighan y Ritchie en 1972, en la ATT. Hasta entonces el estándar lo marcaba el libro escrito en 1978 por estos dos autores[*]

El lenguaje C++ se comenzó a desarrollar en 1980. Su autor fue B. Stroustrup, también de la ATT. Al comienzo era una extensión del lenguaje C que fue denominada C with classes. Este nuevo lenguaje comenzó a ser utilizado fuera de la ATT en 1983.

El nombre C++ es también de ese año, y hace referencia al carácter del operador incremento de C (++). Ante la gran difusión y éxito que iba obteniendo en el mundo de los programadores, la ATT comenzó a estandarizarlo internamente en 1987. En 1989 se formó un comité ANSI (seguido algún tiempo después por un comité ISO) para estandarizarlo a nivel americano e internacional.

[*] LIBRO The C Programming Language, Kernighan and D. Ritchie, editorial Prenctice-Hall, 1978.

En la actualidad, el C++ es un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones. El C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original.

La evolución de C++ ha continuado con la aparición de Java, un lenguaje creado simplificando algunas cosas de C++ y añadiendo otras, que se utiliza para realizar aplicaciones en Internet.

Hay que señalar que el C++ ha influido en algunos puntos muy importantes del ANSI C, como por ejemplo en la forma de declarar las funciones, en los punteros a void, etc. En efecto, aunque el C++ es posterior al C, sus primeras versiones son anteriores al ANSI C, y algunas de las mejoras de éste fueron tomadas del C++.

3.3. HERRAMIENTAS DE DISEÑO GRÁFICO

3.3.1. Adobe Photoshop CS. Es el software estándar de edición de imágenes. Las herramientas presentes dentro de la aplicación ayudan a conseguir resultados excelentes en muy poco tiempo debido a los procesos de edición, aplicación de efectos, tratamiento y gestión de archivos de diversas extenciones, haciéndolo de una forma más eficaz.

3.3.2. Corel PhotoPaint. Esta aplicación ofrece herramientas para edición digital de imágenes, aplicación de efectos especiales, retoques fotográficos y creación de gráficos en movimiento. Permite modificar y crear imágenes rápidamente.

3.3.3. Macromedia Fireworks MX 2004. Es una aplicación que permite realizar el diseño y la optimización de gráficos, además de la integración de gráficos para sitios y aplicativos Web.

Permite el desarrollo de gráficos complejos pero con un rendimiento óptimo y una excelente integración con otras herramientas de la suite Macromedia como lo son Dreamweaver MX 2004 y Flash MX Professional 2004.

3.3.4. Maya. Integra el mundo de la animación y efectos visuales, entregando herramientas de última tecnología para una solución completa de trabajo en 3D.

Maya tiene una interfaz de usuario muy intuitiva, un conjunto de herramientas extenso de animación, herramientas de modelaje y una arquitectura abierta si se desea ir más allá de la animación convencional.

3.4. TEORÍA GENERAL DE VIDEOJUEGOS

3.4.1. Videojuego. La frase “videojuego” (o juego de video) es un conjunto de palabras que definen una idea. La siguiente es la definición de cada una de estas palabras:

Juego: (*del latín locus)* podemos decir que el juego es un acto en el cual una o varias personas se entretienen, es decir el acto de jugar.

Video: Se dice en televisión del procedimiento de grabado en una cinta magnetofónica las imágenes y los sonidos captados por una cámara para luego ser reproducidos inmediatamente.

Desde el comienzo se han entendido a los videojuegos como "un juego", pero un videojuego se entiende también como un medio de almacenamiento en el cual se graba, de manera digital y computarizada, un tipo de juego especial que se le ha bautizado con el mismo nombre que el medio en el que se guarda.

Así que un videojuego es entendible tanto como un medio de almacenamiento y como un tipo de juego.

3.4.2. Creación de un videojuego. Para la creación de un video se necesitan como mínimo las siguientes herramientas:

- Computadoras
- Un lenguaje de programación
- Un programa de diseño gráfico
- Un programa para crear sonidos y música
- Creatividad
- Diseñador de arte
- Diseñadores gráficos y de 3D
- Programadores
- Compositor de música
- Sala para los sonidos
- Algunos especialistas y técnicos en efectos de sonido
- Recursos económicos
- Otros específicos de cada tipo de juego

Lo anterior solo para construcción del videojuego, después vendrá el proceso de mercadeo para poder comercializar el producto.

3.4.3. Diseño de juegos y videojuegos. Un juego esta compuesto por reglas, ambiente del juego, obstáculos, objetivos, virtudes del jugador y el nombre que es lo que identifica a un juego.

Las reglas serían las acciones que se tienen que realizar para que se cumplan otras acciones, es decir serian las condiciones para permanecer jugando. Si no se cumplen las reglas, no se juega.

El ambiente sería el espacio en el cual el jugador esta, los obstáculos son problemas puestos en ambiente en los cuales el jugador debe esquivar. A la vez pueden existir obstáculos que si los pasaste dan un premio o lo que preferimos llamar bonus en el lenguaje de los videojuegos. Así que la solución del problema de determinado obstáculo nos puede dar determinado beneficio, pero si no solucionamos ese problema perdemos ese beneficio y/o ganamos consecuencias graves.

El objetivo se puede confundir con una regla pero en realidad el objetivo seria el obstáculo final para concluir al juego. Las reglas con los obstáculos generan un perfecto diseño de juego en el cual el jugador crea que no puede realizar el objetivo.

Y el nombre, ¿por qué el nombre es tan importante para un juego? El nombre del juego es la primera impresión que causa en la persona y crea un buen ambiente del juego. El nombre habla por toda la arquitectura del juego. Por ejemplo, “Age of empires” es un buen nombre. ¿Jugarías a este juego si se llamara “La edad de los imperios” o “Edad de imperios” o “Los imperios de la antigua edad”?. Los nombres de los juegos son claves exactas para la elección de uno, como una estrategia de publicidad o marketing.

Hemos visto el diseño de los juegos, veamos el diseño en distintos modelos de videojuegos:

3.4.3.1. En el caso de un jugador. Uno de los modelos más antiguos es solo un jugador. La idea es que el jugador se comunique con la consola, de tal manera que el jugador le brinde la información a la consola de lo que el quiere hacer y la consola le dice los resultados de las acciones que el jugador realizo. Ahora en realidad el que da la información al jugador es el programa que esta en un medio de almacenamiento [casette, CD, disco duro, etc.] que lleva al juego programado. Ahora hay que diferenciar dos cosas: lo que es el juego y lo que es el programa. El juego es el conjunto de definiciones mencionadas anteriormente y el programa simplemente son instrucciones metidas en un medio de almacenamiento que le dan órdenes al equipo o consola. A su vez esta consola puede recibir información a través del joystick y enviar información a través de la pantalla. Es decir que en realidad el jugador se comunica con el programa en forma indirecta a través del joystick, la pantalla y la consola.

Aquí podemos exactamente diferenciar al programa del juego. El juego es el problema, el programa la solución. El juego es lo que le da el gusto al programa, ese gusto que es la diversión.

3.4.3.2. En el caso de dos jugadores. Los dos jugadores se comunican a la vez con el programa. Los dos ven en una misma pantalla y enfrentan sus capacidades uno contra el otro, aquí nace un gran cambio que es cuando se enfrenta al jugador contra un jugador controlado por la computadora.

3.4.3.3. En el caso de Múltiples jugadores. Todos los jugadores se comunican a la vez con el programa y el videojuego controla la manipulación de los eventos generados por los jugadores entregando un resultado para todos, aquí entra el concepto de manejo de redes para que los jugadores puedan jugar en pantallas distintas y hasta en sistemas distintos.

3.4.4. Arte en el diseño de videojuegos

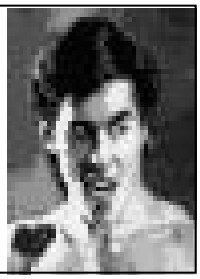
3.4.4.1. Construcción del guión. El guión es el alma del proyecto, es donde están escondidas todas las cartas puesto que la forma en que se narra la historia y la descripción de cada personaje que intervenga son parte fundamental para el desarrollo del resto del juego.

Un buen guión para videojuegos es elaborado a parte de las exigencias de los jugadores expertos, puesto que ellos tienen una visión y un gusto mas elaborado que el de un novato, el experto aporta elementos muy importantes en la elaboración de la historia, desde ideas tan sencillas como el lenguaje a utilizar, hasta llegar a la complejidad de las caracterizaciones de los personajes, su diseño y su historia particular.

3.4.4.2. Historia. Es un elemento que se debe tener siempre en cuenta, al jugador no le gustan los personajes que salen de la nada, siempre como complemento del guión es muy importante anexar un documento donde de manera breve se describe cada uno de los personajes y lugares importantes de la historia de donde vienen, su edad, su profesión y los aspectos mas relevantes de su vida que los han llevado hasta el punto de su aparición en la historia por ejemplo este documento del videojuego mundialmente conocido Mortal Kombat.

Figura 1. Mortal Kombat

- Liu Kang, un antiguo monje Shaolin, fue un miembro de la secta supersecreta White Lotus Society (Sociedad del Loto Blanco). Fuerte en su creencia, el despreciaba a Shang Tsung. Después recibió una invitación para ir al torneo, el dejó el White Lotus y obtuvo el permiso de representar al Templo Shaolin en el Mortal Kombat. El abordó el barco con la esperanza de regresar el torneo al Templo Shaolin.



Fuente: Los Autores

El guión describirá una historia tipo aventura es decir un personaje principal que se desplaza por diferentes escenarios y que tiene la particularidad de héroe, la descripción del mundo que rodea al personaje central es muy descriptiva puesto que tratará en lo posible de representar lugares que existen en el mundo real.

3.4.4.3. Caracterización de personajes y escenarios. En la medida en que un juego logre cautivar la atención del jugador más efectivos serán sus resultados, así pues un juego puede tener una excelente historia, un excelente control y ser impecable a nivel de programación, pero si el ambiente, el cual esta comprendido por gráficos, animación, música, ambientación, escenografía, acción!!, no es lo suficientemente fuerte, se tiene como resultado un pésimo juego, el arte del juego es importante así que si no se cuenta con un buen equipo, se requiere de mucha dedicación y ganas de hacer las cosas bien, hay que ser dibujante, modelador, animador, músico, editor de video, guionista, scripter, programador, debugger y tester.

La idea de esto es principalmente cautivar, deslumbrar e impactar al jugador, el hecho de encontrarse con sorpresas a nivel gráfico o de acción acompañados de un excelente fondo musical... es lo que muchos jugadores sueñan con encontrar o lograr una meta en el juego, estos elementos son el trampolín que necesitan los jugadores para continuar con una nueva aventura.

3.4.5. Programación. Acometida final, sería el nombre de que bien podría merecer la programación del videojuego, aunque tras de este enunciado simplista se esconde la mayor dificultad o reto a vencer, para la programación de un videojuego se requiere conocer mas de un lenguaje de programación, lo que implica como mínimo el lenguaje C y el ensamblador, y se requiere una alta pericia en la resolución de problemas de hardware, y sobre todo tener la habilidad de plasmar las ideas del guionista y de los diseñadores, tal cual como fueron concebidas, la programación del juego es especialmente estructurada, se debe tener iniciativa, entender de procesos de programación y poder optimizar el código. Es también útil (y necesario para 3D) la comprensión de trigonometría [bastante útil en la creación de algoritmos de colisión]. Lo interesante de un programador de videojuegos es que pueda proponer nuevas formas para hacer más eficientes los procesos.

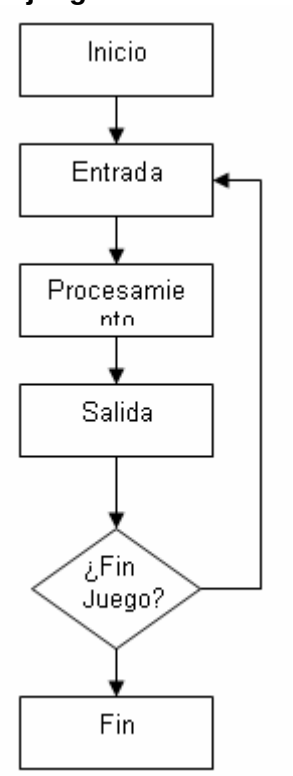
3.4.5.1. Interfaz del control. La interfaz de control consta de dos partes fundamentales, la primera es la jugabilidad, la cual se refiere a que en general, sea fácil entender como jugar y que sea agradable el jugar, y la segunda es la manejabilidad, es decir, qué tan complicado es hacer que el personaje haga un movimiento determinado, esta mas ligada a la parte del control del hardware para realizar una acción que al hecho de saber como se juega.

Hasta hace algunos años los movimientos en el Hardware para lograr una acción especifica, eran algo complicados, hoy las cosas han cambiado y hemos pasado a un paradigma de simplicidad.

3.4.6. Ciclo básico en un videojuego. La mayoría de los juegos están construidos bajo una misma estructura básica sobre la cual corre el programa, por supuesto puede haber juegos que lo manejan con algunas variaciones, pero en esencia un juego trabaja así:

- 1: Inicialización
- 2: Ciclo de Juego
- + Entrada
- +Procesamiento
- +Salida
- 3: Finalización

Figura 2. Ciclo básico en un videojuego



Fuente: Los Autores

En la inicialización se pone al juego en un estado ya predefinido, para que siempre esté de ese modo cuando empiece el juego, como serían los valores iniciales del jugador, su energía, sus armas, la posición dentro del mapa, en qué mundo se encuentra, etc. además también en esta parte es donde se cargán las imágenes, sonidos, tabla de records y demás cosas que son necesarias antes de iniciar el juego.

El ciclo de juego es la parte donde está la acción, es un ciclo que se va a estar repitiendo una, otra y otra vez hasta que el jugador pierda, gane o se salga del juego, en general se puede dividir esta sección en tres partes:

- Entrada:

Aquí es donde se captura todo lo que hace el jugador, como presionar los botones del control, mover el ratón, presionar las flechas del teclado y toda la demás información que recibe el juego.

- Procesamiento:

Esta es la parte donde se procesa toda la información que se recibió de entrada, quizás se podría decir que es el mismo núcleo del juego, ya que en esta parte es donde se lleva acabo la lógica del programa, los cálculos de la física del juego, la inteligencia artificial y en sí la forma como va a responder el juego a la entrada.

- Salida:

En esta parte es donde se le envía al jugador toda la información que se procesó en el paso anterior, es decir, se le muestra la respuesta a lo que hizo; por lo general poniendo en pantalla los cambios, tocando sonidos, etc.

La finalización se ejecuta justamente cuando el juego ha terminado, y todas las instrucciones que se ejecutan aquí tendrán que ver precisamente con dejar al juego en un estado óptimo, como guardar los records que se lograron durante el juego, los valores al finalizar, liberar recursos almacenados durante el juego, etc., etc.

Más o menos el código de un juego se debería de ver así:

```
bool fin_juego;
int main(void){
   inicializacion();
   fin_juego = false;
   do{
Entrada ();
Procesamiento ();
Salida ();
   }while(!fin_juego) ;
   finalizacion();
   return 0;
}
```

3.4.7. Sincronización del juego. Un videojuego siempre está en un constante ciclo, desde que se ejecuta el programa hasta que se termina, y es precisamente en este ciclo del juego donde se tendrán que realizar absolutamente todos los cálculos del juego, desde las animaciones de cada personaje que se encuentre en pantalla, los cálculos físicos, etc. Es por esto que es muy importante siempre estar buscando la forma como la computadora haría más rápido todos los cálculos.

Aquí se presenta un problema, en el caso de una computadora con procesador a 200 Mhz, en donde se corre un juego sencillo donde lo único que hay es un personaje que se mueve de izquierda a derecha en la pantalla y cuenta con el siguiente código dentro del ciclo principal del juego, el cual tarda 0.04 segundos en realizar todo el trabajo que tiene que hacer:

```
int x;  // posicion del personaje
int vx; // velocidad del personaje
x  = 0;
vx = 2;
do{
  // t = 0 seg.
  ...
  x = x + vx;  // aumenta en 2 pixeles (vx) la posición del personaje
  dibuja_personaje(x);
  ...
```

```
  // t = 0.041 seg.
}while(!fin_juego);
```

Como se observa, se esta dibujando el personaje en pantalla aproximadamente 24 veces por segundo, o en términos correctos a 24 Fps (Frames Por Segundo), que es el máximo número de imágenes que puede percibir el ojo humano en un segundo, por lo tanto, la imagen del personaje se vera muy fluida, con una calidad bastante aceptable, y como solo va aumentando su posición en 2 píxeles por cada ciclo entonces cada segundo se estará desplazando 48 píxeles más a la derecha (2 x 24 = 48), si se quiere que se mueva más rápido simplemente se aumentaría la velocidad en "vx". Aquí no hay ningún problema, el personaje se mueve a 48 píxeles por segundo, y el juego corre a 24 Fps, el problema se presenta cuando se intentara correr ese mismo juego en otra computadora con un procesador de distinta velocidad.
En una computadora a 100 Mhz el ciclo principal del juego en lugar de tardarse 0.041 segundos como en la anterior se tardaría 0.082 segundos, lo que haría que en un segundo solo se pudieran ejecutar 12 ciclos, y por lo tanto se dibujaría en pantalla a una velocidad de 12 Fps, y el personaje se movería solamente 24 píxeles cada segundo.
Y en una computadora de 800 Mhz el ciclo principal de juego tardaría solamente 0.01 segundos, que haría que tu juego corriera 100 ciclos cada segundo, por lo tanto se dibujará en pantalla a una velocidad de 100 Fps, y el personaje se movería a 200 píxeles por segundo.
Así el mismo juego, con exactamente el mismo código, incluso usando el mismo ejecutable, puede correr mucho más lento o más rápido dependiendo de la computadora en que corra, lo cual es necesario evitar al máximo.
Es por eso que, para evitar que el juego corra a diferentes velocidades en diferentes computadoras, es necesario sincronizar el juego, "forzar" a que el juego corra siempre de una forma al menos deseable si no es que óptima.
Existen dos formas de sincronizar un juego, con base en el framerate, y con base en el tiempo.

3.4.7.1. Sincronización por framerate. Cuando se sincroniza un juego en base a su framerate (número de frames por segundo), lo que se hace es detener el ciclo principal del juego hasta que se ejecute el siguiente ciclo, es decir si el juego corre a 24 fps lo que se hace es fijar el framerate, o sea forzar a que cada ciclo dure 0.041 segundos, esto se hace de la siguiente manera dejando que se ejecute todo el ciclo del juego, y cuando termine de ejecutarse se revisa si pasaron 0.041 segundos desde que se inició, si no, se espera a que se cumplan los 0.041 segundos, y cuando se hayan cumplido se ejecuta el siguiente ciclo.

Así no importa si el juego se desarrollo en una computadora con un procesador a 200 Mhz y lo están corriendo en otra con un procesador a 1.8 Ghz, siempre se va a ejecutar el ciclo principal del juego 24 veces cada segundo, ejemplo:

```
int x;  // posición del personaje
int vx; // velocidad del personaje
int t1; // tiempo con el que inicia el ciclo
int t2; // tiempo con el que finalizó el ciclo
x  = 0;
```

```
vx = 2;
t1 = 0;
t2 = 0;
do{
  t1 = obtener_tiempo();  // en milisegundos
  ...
  x = x + vx;  // aumenta en 2 píxeles [vx] la posición del personaje
  dibuja_personaje(x);
  ...
  do{
              // se mantiene en un ciclo haciendo nada
    t2 = obtener _ tiempo();  // hasta que la diferencia entre t1 y t2
  }while( (t2-t1) <= 41 );  // sea mayor a los 41 milisegundos (0.041 seg)
}while( !fin_juego );
```

De esta forma, como "x" se ha aumenta a un ritmo constante el personaje siempre se moverá 48 píxeles cada segundo.
Pero existe un problema, no se podrá implementar en computadoras mas lentas, ya que no se puede hacer que se trabaje mas rápido, para esto se desarrolla el juego para un equipo límite, es decir, se desarrolla para computadoras con un mínimo de velocidad.

3.4.7.2. Sincronización por tiempo. Cuando se sincroniza un juego en base al tiempo no importa tanto el framerate que tenga el juego, es una de las grandes ventajas de sincronizarlo de esta forma, puede ser que el juego esté corriendo con un framerate de 8 fps o de 100 fps, ¡no importa!, los objetos que se encuentren dentro del juego siempre se moverán a la misma velocidad.

Si se quiere que el personaje se mueva a 48 píxeles por segundo, una de las características principales es que el framerate no está fijo, puede variar tranquilamente en cada computadora sin afectar la velocidad de los objetos, incluso algunas veces puede llegar a variar el framerate dentro de la misma computadora mientras se está ejecutando el juego y no le afectaría en lo más mínimo, ejemplo:

```
int x;  // posición del personaje
int t1; // tiempo con el que inicia el ciclo
int t2; // tiempo transcurrido hasta antes de calcular la nueva "x"
float vx;   // velocidad del personaje
x  = 0;
vx = 0.048; // velocidad a la cual se mueve el personaje
t1 = 0;
t2 = 0;
do{
  t1 = obtener_tiempo();  // en milisegundos
  ...
  t2 = obtener_tiempo();  // justo antes de calcular la nueva posición de "x"
  x = x + vx*[t2-t1];   // la diferencia de tiempos la multiplicamos por "vx"
  dibuja_personaje[ x ];
  ...
```

```
}while( !fin_juego );
```

De esta forma si el juego se esta ejecutando en una computadora en verdad extremadamente lenta y que se tardara 1 segundo en ejecutar cada ciclo el nuevo valor de "x" se calcularía:

x = x + 0.048*[1000]
x = x + 48

Que si se tardara un segundo el ciclo del juego el personaje avanzara 48 píxeles, ahora que si se corre el juego en una supercomputadora que lo ejecutara en 0.021 segundos (21 milisegundos) el nuevo valor de "x" se calcularía:

x = x + 0.048*[21]
x = x + 1

Ya que para que el personaje se mueva 48 píxeles se tardaría... 21 milisegundos x 48 = 1008 milisegundos, ¿bien no?, como se puede ver no importa qué tan rápida o lenta sea la computadora donde se esté ejecutando el juego, el personaje siempre se moverá a la misma velocidad.
Pero tiene dos pequeños inconvenientes, primero la utilización de variables flotantes y variables enteras afectará la precisión del juego y hay que manejar el error redundante, segundo entre la computadora sea más lenta se tendrá un movimiento menos fluido.

4. DESCRIPCIÓN DEL VIDEOJUEGO

Figura 3. Logotipo del Grupo

Fuente: Los Autores

4.1. FILOSOFÍA

4.1.1. Visión del Juego. Realizar un juego que utilice aspectos de la mitología colombiana desde un punto de vista de acción y aventuras. En el juego se emplearán elementos propios del folclor Colombiano, así como la fauna la flora y otras características autóctonas de cada región.

En el se pretende colaborar con la conservación de la cultura colombiana.

4.2. ENFOQUE TÉCNICO

El Juego esta destinado a ser utilizado en PC de escritorio con las siguientes características mínimas:

- 128 MB en RAM
- Procesador Pentium 3 450Mhz
- Espacio 500Mb
- Tarjeta Video 32Mb
- (opcional) control Genius G-08
- Sistema Operativo Windows 2000 o superior
- DirectX 9.0
- .Net Framework 1.1.

4.3. PREGUNTAS COMUNES

4.3.1. ¿Qué es Fantasía Mitológica? Fantasía Mitológica es un videojuego Acción RPG 2D desarrollado para plataforma Windows, utilizando el lenguaje de programación C# y la Librería DirectX 9.0 basado en la mitología colombiana en el cual se cuenta la historia de Zue el personaje principal que lucha por salvar a su pueblo de las fuerzas mitológicas en las cuales aparecen los grandes mitos colombianos como lo son El Mohan, La Patasola, El Sombrerón, el Hombre caimán entres otros.

4.3.2. ¿Por qué se Creo este juego? El videojuego se esta creando como parte del proyecto de grado de Ingeniería de sistemas por parte de los integrantes del proyecto, adicionalmente se ha creado con el ánimo de impulsar el desarrollo de la industria de software en Latinoamérica centrándose en el área del entretenimiento y mas específicamente en el desarrollo de videojuegos en donde las compañías extranjeras ganan mucho dinero.

4.3.3. ¿Dónde el juego toma lugar? La historia del juego transcurre en las diferentes regiones de Colombia destacando sus características culturales, sociales, geográficas etc., pero principalmente los mitos más característicos de cada región.

4.3.4. ¿Qué se va a Controlar? El jugador controlará el personaje principal Zue quien poseerá diferentes tipos de poderes y magias con los cuales enfrentará sus enemigos, se controlará el desplazamiento a través de las diferentes escenas y las situaciones que se puedan presentar, así como las diferentes magias y diálogos con los demás personajes que participen en el juego.

4.3.5. ¿Cuál es el Objetivo Principal? El objetivo del juego es avanzar a través de cinco (5) escenas, derrotando a un jefe principal en cada una de ellas y una escena final donde se pretende encontrar a la madre de Zue el personaje principal.

4.3.6. ¿Qué es lo innovador? A nivel Colombia no existen proyectos formales relevantes al desarrollo de videojuegos, a nivel de la Universidad Católica de Colombia tampoco se han realizado ni existen en este momento proyectos relacionados con la creación de uno, razón por la cual nuestro proyecto resulta innovador en un marco local.

Adicionalmente los proyectos que se han generado con un perfil similar al de los videojuegos siempre han estado enfocados al entorno puramente educativo, este proyecto aunque pretende dar difusión de muchos aspectos de la mitología Colombiana está lejos de convertirse en un proyecto de esa índole ya que el objetivo principal es el entretenimiento.

4.4. CONJUNTO DE CARACTERÍSTICAS

4.4.1. Características Generales

- Tipo: RPG + Acción
- Resolución de 640 x 480 píxeles
- Intuitivo, rejugable.
- Fácil de Manejar
- El juego es expandible.
- Pantalla Completa y en ventana de Windows.

4.4.2. Características De Juego

- Soporte de teclado y joystick
- Múltiples pantallas
- Menú de opciones
- Permite guardar avances
- Personaje Principal Multifacético
- Cambios en medio ambiente
- Trucos y sorpresas ocultas (cheats)
- Efectos visuales
- Efectos de Sonido
- Múltiples modos de juego
- Resolución de Acertijos
- Divertido
- Descripción Básica del Motor del Juego

4.5. DESCRIPCIÓN DEL ENGINE

Se desarrollara un engine de videojuego 2d, en el cual se cubrirá el flujo del programa y se pondrán todas las funciones necesarias para el juego. Se realizará su diseño en UML (Lenguaje Unificado de Modelamiento) y será orientado a objetos.

4.5.1. Características del Engine. Soportará el manejo de ventanas en el juego, al iniciar el juego será el encargado de configurar el hardware para que el juego pueda iniciar. También soportará la administración de los avances y el formato de los archivos de avance. Tendrá una arquitectura abierta para que el juego sea actualizable, y puedan cargarse misiones extras.

4.5.2. Detección de Colisiones. El motor de juego no es el encargado del control de colisiones, esta característica se maneja de manera individual entre personajes, escena, personajes alternos y personaje principal. Se ha ideado una la técnica de colisión por puntos de contacto los objetos que interactúen entre si. Se desarrollará un algoritmo de estados para que se pueda tomar la decisión más correcta.

4.6. HISTORIA

4.6.1. Aspectos a tener en cuenta en el desarrollo de la historia. Se han tenido en cuenta tres aspectos fundamentales crear aspectos de la historia, cada uno de estos aspectos ha sido analizado de manera independiente respecto a los intereses personales de los miembros del grupo procurando una selección y desarrollo puramente profesional.

4.6.2. Aspecto geográfico. El juego, en busca del objetivo general, debe procurar hacer alusión constantemente a lugares de la geografía colombiana que correspondan con personajes mitológicos determinados, involucrar estos lugares en la historia y en la cotidianidad del personaje principal es un requerimiento desde la definición del proyecto.

4.6.3. Aspecto de autenticidad. La historia debe involucrar características autóctonas de las diferentes regiones del país, como lo son la fauna y la flora, adicionalmente a favor del aspecto gráfico se debe procurar que la descripción de los lugares sea bastante subjetiva, con el fin deque en la etapa de diseño exista una mayor libertad ala hora de diseñar los escenarios ligados a las características de cada región.

4.6.4. Aspecto de profundidad. Aunque ya era un aspecto cubierto este aún se encontraba en desarrollo en el momento de realizar este análisis, por lo cual es conveniente hacer alusión al mismo, las escenas deben contar con una descripción detallada de los lugares clave en cada una de ellas y adicionalmente deben tener alusión al aspecto geográfico correspondiente, cada diálogo de la escena debe ser descrito con el detalle necesario de tal manera que los diálogos que sea necesario crear en el proceso de desarrollo sean mínimos.

4.7. GUIÓN DE LA HISTORIA

Hace diez años en la región central del Magdalena, desde Hilarco, en Purificación, hasta Guataquicito en Coello se realizaba una expedición científica con el fin de descubrir nuevas especies de fauna y flora en la cual participaban dos reconocidos biólogos, el Dr. Manuel Elkin Pies del Rió y la Dra. Maria Peña quienes además eran esposos.

Todo marchaba muy bien, pero en la noche del tercer día de la exploración la Dra. Peña se alejo un poco del grupo para apreciar la belleza de la noche desde la rivera del rió, mientras tanto su esposo conversaba con los lugareños quienes le comentaban con cierto

temor que en el transcurso del día muchas personas aseguraban haber visto misteriosos personajes en las cercanías del rió Magdalena, un personaje al cual ellos llamaban el Mohan, mientras el Dr. prestaba atención a los relatos de estas personas se pudo dar cuenta que las mujeres de los caseríos se encontraban rezando temerosas.

Hablaron bastante acerca de este personaje, uno de ellos, el más viejo del pueblo decía:

“Se les presentaba como un hombre gigantesco, de ojos vivaces tendiendo a rojizos, boca grande, de donde asomaban unos dientes de oro desiguales; cabellera abundante de color candela y barba larga del mismo color.”

Rafael Pérez decía:

“Con las muchachas era enamoradizo, juguetón, bastante sociable, muy obsequioso y serénatelo. Perseguía mucho a las lavanderas de aquellos puertos, como en la Jabonera, la Rumbosa, el Cachimbo.”

De repente el Dr. se percató de la ausencia de su esposa, dirigió su mirada al cielo y logro apreciar un abrupto cambio en el color de la luna... en ese momento el cielo fue cubierto por las nubes y se inicio una fuerte tormenta, desesperado empezó a buscar a buscar a su mujer sin éxito alguno…

Mientras tanto para la Dra. Pérez todo parecía transcurrir de manera normal, el cielo era despejado y las estrellas se reflejaban sobre el rió, de repente vio llegar una boga a la orilla del rió en la cual viajaba un hombre mestizo de cabellera larga quien se acercó lentamente hacia donde ella se encontraba, comenzaron a sostener una conversación hasta que sin razón alguna la Dra. cayó en sueño… al siguiente día ella despertó en un lugar muy extraño, no existía norte ni sur, ni día ni noche, solo una pequeña caverna donde vivir, con muchos objetos de oro y piedras preciosas. Nunca se volvió a saber nada de la Dra. Pérez, el Dr. Pies del Rió sufrió por varios años la pérdida de su esposa, su hijo Zue es el único aliciente que le quedo para seguir adelante, y el único recuerdo que posee de su amada.

-- Quince años después… Zue se ha convertido en un adolescente, gracias al trabajo de su padre se ha criado en un ambiente de expediciones arqueológicas a través de Colombia, ha visitado lugares muy importantes como lo son la laguna de Guatavita, las selvas del Catatumbo, la ciudad perdida de los Tayrona, San Agustín entre otras…

Transcurría época de navidad, Zue y su padre se encontraban en una expedición en la amazonía investigando a los delfines rosados, su padre entro en mal estado de salud tras haber sido picado por un insecto, por varios días su padre agonizó y mientras deliraba nombraba innumerables historias de leyendas populares, raptos y tesoros ocultos, un día en el lecho de su muerte mando llamar a su hijo Zue y le dijo:

--Dr. Pies del Rió:

“Hijo mió, te diré la verdad, tu madre no murió ahogada como siempre te había dicho… a tu madre se la llevo el Mohan... tienes que buscarla por mí”

El padre de Zue tomó una pequeña libreta de notas y se la entregó a su hijo, en ese momento... exhalo y murió.

4.8. ESCENAS DEL JUEGO

4.8.1. Descripción Global. El juego transcurre en diferentes lugares del país abarcando las características geográficas, folclóricas, climáticas y míticas más importantes de cada una.

Cada escena esta conformada de cuatro (4) partes de las cuales la última parte es el enfrentamiento con el jefe de la escena.

Dentro de las características del paisaje se puede encontrar ríos, lagunas, cascadas, selvas, habitantes y animales más característicos de cada región.

Para navegar a través de estos mundos se hará a través de un mapa de navegación el cual será más versátil conforme se avanza en el juego, en el mapa podrá observarse el estado del clima en cada región, la hora global (dentro del juego) en un momento determinado.

Dentro del mapa se puede encontrar una descripción de la región y el enemigo principal.

4.8.2. Zonas Comunes

a) Tienda: Una tienda consta de los siguientes elementos:

- Tendero: Siempre es el mismo personaje pero cambia su atuendo de acuerdo a la región a la que pertenezca.
- Vitrina: Estante cambia de acuerdo a la región.
- Decoración: Un letrero de tienda, seguido del tipo de tienda y un cuadro característico.
- Menú de productos venta: Todos los ítems que están a la venta y su precio.
- Menú de productos compra: Todos los ítems que se compran en la tienda y su precio.
- Diálogos: Los diálogos se adecuan al dialecto de cada región

“Buenos días Señor, le puedo ofrecer los siguientes productos:” + Menú
“Déme X cantidad de aquello.”
“Estoy comprando estos ítems”
Le vendo C Ítems.

b) Casa de habitantes: Una casa consta de los siguientes elementos:

- Ocupantes: Serán entre uno y tres personajes los cuales brindarán información de acuerdo a la escena en que se encuentren y también podrán solicitar algo a cambio de

ella. Siempre serán los mismos personajes pero vestirán de manera diferente de acuerdo a la región en que se encuentren, no siempre saldrá la misma cantidad de personajes.

- Un mueble: Una cama o una mesa, o una vitrina.
- Decoración: Tres elementos de decoración, floreros, cuadros, una foto.
- Una mascota: Perro, gato, ave… etc.

c) Santuario: Es un lugar de sanación y una buena fuente de información siempre y cuando se entreguen a cambio reliquias y objetos religiosos.
Un santuario consta de los siguientes elementos:

- Sacerdote (o equivalente): Será uno por santuario.
- Un ídolo: Cruz, Tótem, estatua etc.
- Elementos de luz: Velas, candelabros, hoguera etc.

Todos los elementos cambian de acuerdo a la región en la que se encuentre el santuario.

d) Hotel o posada: Es un lugar de descanso, sirve para guardar avances y para guardar elementos también se puede conversar con personas para obtener información extra:

- Anfitrión: Administrador del hotel, hay que pedirle permiso para tomar un cuarto.
-
- Una empleada: Quien da las pistas a manera de chismes
-
- Ocupantes: Pueden ser de uno a tres por hotel pueden brindar información.
-
- Decoración: Recepción, Bocadillos, 2 o 3 cuadros, una mesa, un sofá una silla y unas escaleras al segundos piso.

Todos los elementos cambian de acuerdo a la región donde se encuentre el hotel.

4.9. ESCENAS DEL VIDEOJUEGO

El juego consta de cinco (5) escenas cada una de las cuales se llevará a cabo en una región específica del país donde a su vez deberá enfrentar con un ser mitológico determinado.

4.9.1. Escena Uno

- Ubicación : Amazonía Colombiana
- Jefe : Bufeo (delfín rosado)
- Fondo : Selvas, río, caseríos.
- Descripción : Inicia con la lectura de la libreta de notas del padre donde se encuentra el siguiente mensaje:

"Estaba en la región amazónica, conversé con el Sr. Yurupari de la tribu Los witoto y me contó de un personaje que enamoraba a las muchachas bonitas y nunca volvían a aparecer…"

Flujo y detalles de la escena:

PARTE UNO: Tierra de Santa Sofía: Zue se dirige hacia donde se encuentra la tribu witoto, se enfrenta con un jaguar y varios mosquitos, al llegar al pueblo se observan seis chozas en 3 de las cuales la puerta parece de un color mas resaltado indicando que se puede ingresar en ellas, una de las chozas es una tienda naturista, la otra es de una familia indígena y la última es la del indígena mas viejo, en ellas se deberá obtener la información necesaria acerca de la indígena "Solpeta", quien es una mujer que sobrevivió al los encantos del bufeo.

Lugares clave:

Tienda naturista: En este lugar se consiguen:

- Polvos de tripas de boa
- Botas pantaneras
- Plantas medicinales
- Polvos del sueño
- Manual de supervivencia en la selva

En esta tienda el tendero compra pescado y pieles de animales. El único pescado que se compra es el pirarucú o pez rojo gigante.

Casa Indígena: En ella vive una familia de tres indígenas quienes cuentan la historia de su hija, la cual desapareció en las aguas del río amazonas víctima del bufeo.

Padre: Mi hija conoció un hombre en las riberas del río amazonas y comenzó a visitarla todos los días en la noche.

Madre: el siempre traía un sombrero y ropas elegantes, era fuerte y alto.
Hijo: una vez vi que la piel de espalda era de un rosado como el de los delfines del río.

Casa Antigua: En ella habita un viejo anciano el cual narra una leyenda de un delfín que se transforma en hombre y se lleva a las mujeres.

Viejo: El Bufeo es un acuático que tiene grandes poderes, se transforma en atractivo hombre que seduce a las mujeres jóvenes. Tiene una gran fuerza; pero no es humano es un delfín rosado.

Solo una joven se pudo escapar de él es la india solpeta.

Zue: como escapo. Dígame por favor.

Viejo: en realidad no se, solo te puedo decir que ella vive en la profundidad del parque nacional Amacayacu.

PARTE 2: Parque Nacional Amacayacu

Zue ingresa a la selva, después de enfrentar algunos animales nativos; llega al encuentro con una tribu conocida como los Tucanos, ellos no le han una cordial bienvenida y lo obligan a superar el reto de la cascada, en la cual debe recuperar una antigua reliquia. En pueblo esta conformado por una tienda, un santuario y varias casas.

Lugares clave:

Tienda de Herramientas: En este lugar se consiguen artículos y armas:

- Machete largo
- Soga
- Impermeable
- Repelente de insectos
- Fósforos
- Hacha
- Costal

En esta tienda el tendero compra caucho y madera.

Santuario: Es el santuario de la tribu se encuentra el brujo de la aldea, el cual le entrega la misión y le promete a cambio revelarle el lugar donde se encuentra solpeta y enseñarle como protegerse de las boas.

Brujo: Tu misión es rescatar el tótem del jaguar el cual está detrás de la cascada de la boa. Ten cuidado con el laberinto de canales del río es traicionero y lleno de peligros

PARTE 3: Laberinto De La Selva

Los aldeanos le proveen de una canoa, comienza su viaje hacia la cascada de la boa, se encuentra en un laberinto en cual debe tomar un serie de decisiones sobre el rumbo que tomará (izq, arr, der, arr, aba, der, arr, izq), se enfrentará en el camino a insectos y arañas

colgantes, al final encontrará una roca con un pintura de tres micos cada uno con una fruta (Figura de micos tapándose los ojos, orejas y boca).
Al llegar a la cascada una gran boa la estará custodiando, tendrá que matarla utilizando las armas y magias que posea, adicionalmente en la fosa que queda en la cascada esta poblada de peces de varias especies, regularmente cambian las especies, según las horas del día.

Reto: Al matar la boa se abre la cascada y aparecen tres vasijas en diferentes niveles de altura; un pequeño mico se acercará y comenzará a saltar sobre las vasijas podrá conversar con él o ahuyentarlo de un golpe. Para conversar con él deberá darle 3 tipos de frutos diferentes en el orden indicado por la roca, Cuando el mico hable le dirá que deberá compensar las vasijas en peso utilizando arena. La medida correcta será (5,5,5) cuando logre esto las vasijas bajarán y se abrirá un pequeña caja la cual contiene el tótem del jaguar.

Zue lo tomará y lo llevará al santuario de los tucanos ahora no deberá tomar la ruta del río sino con una soga ir de árbol en árbol. Al entregar el tótem el brujo le dirá:
Brujo: Solpeta se encuentra en la rivera occidental del rió, casi nadie se acerca a esos lugares porque son muy peligrosos.
Toma estas plantas las cuales te ayudarán a dormir a las anacondas.
Brujo: La ruta a seguir en la canoa deberá ser: izq, izq, arr, arr y dig.

Parte 4: Solpeta Y Bufeo

Avanzará entre la selva hasta hallar el rancho del solpeta, ella estará dentro de la choza:

Solpeta: Extraño, tu deberías salir de estas tierras es muy peligroso.

Zue: Que sabes tú del Bufeo?

Solpeta: Es un Delfín con grandes poderes, se convierte en un hombre corpulento y atractivo. Solo la suerte me salvo de él.

Zue: Como te le escapaste?

Solpeta: Fue gracias al hechizo de escamas del pez pirarucú el cual me enseñó mi abuela.

Solpeta: No he podido matarlo el polvo solo logra que se aleje de la choza.

Zue: Debo interrogarlo haber si sabe algo sobre mi madre

Solpeta: creo que con una dosis más fuerte podremos dormirlo y obtener la información

Si tiene pez podrá preparar el hechizo, si no deberá ir a conseguirlo.

En ese momento se escucha un ruido en las afueras. Es el Bufeo, aparece un menú en donde zue tiene la opción de enfrentarlo o retirarse.

En este momento se debe regresar a la tierra de santa Sofía para obtener información acerca del gran pez rojo donde el viejo de la casa antigua dará las indicaciones pertinentes.

Zue: Conoces algo sobre el pez rojo?
Viejo: Si, que sabe delicioso con algo de salsa de ajo y un delicioso guiso de tomate.
Zue: ummm…????

Viejo: Ya veo, no te gusta el ajo… de todas maneras puedes traer y uno y yo te preparare algo especial.

Zue: ¿Dónde lo puedo encontrar?

Viejo: En el fondo de una cascada!!! En las horas que el Sol te da sueño.

Una vez se tenga el pez se puede ir donde el anciano a comer un plato que él mismo prepara, lo cual aumentará un poco la energía, o bien se puede ir donde Solpeta para preparar más poción y así adormecer al Buféo para poder averiguar sobre la madre de Zue.

Zue: Solpeta, Solpeta!!!, he traído al pez!!!

Solpeta: Muy bien!!! Prepararemos la poción

Solpeta prepara la poción.

En ese momento nuevamente llega el Buféo pero al enfrentarlo, este se duerme debido al efecto de los polvos tras arrojarle 5 dosis de estos.
Una vez dormido…

Buféo: ¿Qué quieres de mí?

Zue: Quiero que me digas si tu tienes a mi madre…

Buféo: jajajajajajajajajaja

Zue: ¿¡Qué sabes de mi madre!? Grrrrrrrrrr…

Buféo: Niño estupido, no me pude haber llevado a tu madre, puesto que a mi solo me gusta llevarme a las jovencitas inocentes.

En ese momento el Buféo escapa.

Zue, y solpeta quedan sorprendidos.

FIN DE LA ESCENA.

4.9.2. Escena Dos

Ubicación: Orinoquía
Jefe: Madre de Monte
Fondo: Llanos, selvas, caseríos.

Descripción:

En la libreta de notas del papá de Zue encontró un párrafo que mencionaba lo siguiente “Los indígenas Guahibo en el Guaviare mencionan la existencia de un espíritu de la naturaleza que la protege de la destrucción, los colonizadores llamaron a este espíritu, la madre monte”

Flujo y detalles de la escena:

Parte Uno: Caserío La Juga
Zue se dirige desde la amazonía hasta la zona media del departamento del Guaviare, a una población llamada La Juga. Antes de llegar al caserío, Zue debe enfrentarse a varios espantos y duendes. En este caserío se encuentra uno de los líderes de la tribu los Guahibos. La ubicación de este líder de nombre Wamone, es en un centro médico donde esta con otros miembros de la tribu, su padre fue una de las 3 personas que fueron desaparecidas por desafiar al espíritu del Monte.

Wamone: ¿Quien eres muchacho?

Zue: Mi nombre es Zue, mi papá era arqueólogo y me contó historias acerca de seres misteriosos, de hecho uno de esos monstruos rapto a mi madre hace mucho tiempo.
Wamone: Mi padre también fue víctima de uno de los que tú llamas monstruos, nosotros los descendientes de indígenas lo conocemos como el espíritu de la naturaleza, pero los colonizadores lo conocen como Madre Monte.

Zue: Madre Monte?

Wamone: Ella protege los seres de la selva, las plantas, los árboles, los ríos, los animales, además de mantener el equilibrio de la naturaleza, tus intenciones son nobles muchacho, pero tendrías que modificar el equilibrio de la naturaleza para que este ser apareciera.

Zue: Alterar el equilibrio de la naturaleza?

Wamone: Existen cazadores furtivos cerca de la Macarena, solo tienes que ubicarlos y de seguro allí también llegará la Madre Monte.

Zue: Muy bien, me dirigiré hacia allá.

Wamone: Te entregaré esta pulsera, con un colmillo de Jaguar, este te ayudará a detectar las extrañas energías de la naturaleza, además de la roca del sol, que te será de gran ayuda para hacer conjuros de fuego.
Tienda: El lugar donde se consiguen

- Pociones
- Textos con hechizos de fuego.
- Escamas de Caimán Negro

PARTE 2: San José Del Guaviare

Inspección de Policía
En ella se encuentran 2 policías.

Policía 1: Con que tú eres el joven que esta preguntando acerca de los cazadores furtivos?.

Zue: Si señor, necesito sabes dónde se encuentran.
Policía 1: Hay informes de haberlos visto en la Sierra de la Macarena.
Zue: Esta muy lejos de aquí?

Policía 1 y 2:Jajá ja já, estás loco? Piensas ir hasta allá? Ni quisiera nosotros nos atreveríamos a realizar tal viaje para reclamarles por sus acciones.

Policía 2: Muy bien, joven tonto, la sierra de la macarena esta aproximadamente a 150 Km. de aquí, puedes ir en lancha río arriba, pero nosotros no nos haremos responsables de lo que te suceda.

Puerto en el Río: En el se encuentra un señor muy obeso y de barba.

Zue: Buenas Tardes señor, cuánto costaría un viaje desde aquí hasta la sierra de la macarena?

Lanchero: Un viaje río arriba hasta la sierra de la macarena? Ni borracho que estuviera, Y por qué quieres hacer ese viaje?
Zue: (…)

Lanchero: Muy bien chico listo, si eres hábil, te entregaré esta vieja canoa para que hagas ese viaje. Tendrás que adivinar en donde queda ubicada esta esfera después de que yo la tape y mueva estas tapas.

Zue por más que intente nunca le ganará, hasta que después de 5 turnos el lanchero le da pesar de verlo perder y le regalará la vieja chalupa.

Tienda: El vendedor le ofrecerá productos que Zue podrá utilizar para realizar su viaje en lancha.
Combustible para Lancha

Comida
Impermeable.

Parte 3: Sierra de la Macarena: Aquí Zue probará sus nuevas habilidades con los hechizos de fuego, al tener que enfrentar a varios enemigos.
Arañas, Águilas, tigrillos y ánimas en Pena.

Habilidad

El camino se encuentra bloqueado debido a que hay un un hueco que no se puede saltar ; Para poder avanzar Zue tendrá que utilizar la Roca del Sol para quemar un árbol,

Al final de esta escena Zue encuentra a 3 cazadores furtivos, tratando de cazar un caimán llanero.

Zue: Con que ustedes son los cazadores furtivos.
Cazadores: Lárgate de aquí niño, sino quieres meterte en problemas.
Zue: Yo en problemas, los de los problemas serán ustedes cuando llegue la Madre Monte.

La pulsera con el colmillo de Jaguar empieza a iluminarse. Cuando de repente aparece un jaguar enorme gruñe y se transforma en la Madre Monte.

Madre Monte: Ahhhh Mortales ingenuos, como se atreven a atacar a un animal inocente, pagarán muy caro su osadía.

Cazador 1 y 2: Virgen del agarradero agarrame a mi primero.

Cazador 3: Cálmate animal feroz, que primero fue el niño Jesús que vos.

Madre Monte: Ni su virgen ni sus rezos podrán ayudarlos ahora.

Cazadores: Señora, cálmese, la idea de matar al caimán es del muchacho para tener una chaqueta de cuero. Vámonos, que esto se esta poniendo como peligroso.

Madre Monte: Es eso cierto?

Zue: No Madre Monte eso no es cierto la culpa es de esos cazadores, yo solo venía a.

Madre Monte: Cazadores, ya no veo a ninguno.

Zue: Oh Oh.

Parte 4: La Furia de la Naturaleza, La Madre Monte.

Zue: Madre Monte, yo no fui el que quería atacar a ese caimán.

Madre Monte: Todos los humanos son iguales, quieren progresar sin importarles el daño que sufre la naturaleza, me aseguraré que no vuelvas a dañarla.

Enfrentamiento: La madre monte desaparece entre la espesura y empieza a atacar con el látigo venenoso, Zue tendrá que evadirlo, y atacarla cuando aparezca utilizando La Roca del Sol, pero también tendrá que evadir su ataque de cerca, que es los pétalos de Orquídea.

FIN DE LA ESCENA.

4.9.3. Escena tres

Ubicación :
Eje Cafetero
Jefe :
La Patasola
Fondo :
Montañas, cafetales, Caminos

Descripción: Inicia con la lectura de la libreta de notas del padre donde se encuentra el siguiente mensaje:

“En los hermosos cafetales entre los límites de Antioquía y Caldas; por los lados de la pintada, los recolectores cuentan que aparece un extraño ser con forma de mujer que ha sido condenado, que encanta a los hombres para devorarlos según Pedro López”

Zue debe viajar hasta la zona cafetera colombiana en la cual encuentra hermosos paisajes y con personas que lo atienden de manera cordial.

Flujo y detalles de la escena

PARTE UNO: BAJANDO LA MONTAÑA

Zue debe atravesar un gran camino rodeado por árboles, la noche es brumosa la luna solo aparece por momentos. Por el camino se escuchan extrañas voces algunas veces aparecen unos espantos que lanzan rocas contra zue. Son espantos de media noche; Zue no los puede atacar debe tratar de avanzar con el menor número de heridas posibles.
Más adelante en el camino se encuentra con perros endemoniados que lo atacan, para librarse de ello podrá enfrentarlos o lanzarles comida para entretenerlos y que no lo ataquen también podrá utilizar el hechizo del sueño para dormirlos y escapar.

Al final del camino aparece el Sombrerón, un personaje malévolo el cual condena a la persona a cargar un sombrero y pasa la condena de persona en persona.

Enfrentamiento:

Dialogo: Sombrerón: “si te alcanzo te lo pongo”

Zue: Conozco tu secreto hoy serás derrotado, sombrero maldito
Sombrerón: prepárate para tu muerte.

Zue deberá evitar que el sombrero lo atrape, en especial cuando la luna este oculta por que si esto pasa será condenado. Para derrotarlo deberá quemar el sombrero con el poder de la roca del fuego.

Cuando el sombrero es derrotado cae y zue se da cuenta que es pedro López.

Zue piensa: “Otra pista equivocada”.

El personaje entra a un pueblo cafetero.

Lugar Clave:

Antes del enfrentamiento, al final del camino un pequeño punto de luz aparece, esta encima de una tumba, la lápida de la tumba tiene una serie de letras, que el personaje debe organizar hasta formar el nombre “Pedro López”, cuando lo consiga la lápida caerá y aparecerá un caja la cual contiene una bolsa con monedas. Un escrito con el título hechizo del café, y un papel con la siguiente leyenda:
“He visto un personaje que me persigue todas las noches me grita “si te alcanzo te lo pongo” cuando por fin llego a la casa solo escucho risas, mi mujer me ha abandonado, la próxima vez que lo escuche lo esperaré”

PARTE DOS: LA HACIENDA LA CONSENTIDA

Zue debe investigar su próximo destino; en el pueblo hay todas las partes típicas:
Tiendas de Café, Santuario, Posada, Casa de Familia. El deberá encontrar la Información necesaria para continuar su viaje.

Lugares clave:

Tienda de Café: Aquí el podrá comerciar, con frutas y madera recogida por el camino de llegada o con el oro encontrado en la guaca. Productos
Ungüento para heridas
Botella de tinto con hierbas

Casa de Familia: Aquí aparece un familia cafetera la cual le habla sobre un ser misterioso al cual llama la patasola.

Zue: Hola, Saben ustedes de algún ser extraño en estas tierras

Mujer: he escuchado de un espanto o ser horripilante que se lleva a las personas.

Zue: ¿En qué Lugares?, ¿Cómo es?

Viejo: Por los lados de los cafetales de la hacienda “La Consentida”, sal de aquí no queremos que molesten a los espíritus de las santas ánimas.

Zue caminará hasta la hacienda, a través de café en los cuales pájaros y gusanos lo atacan.

Al final del camino encuentra una mujer

Mujer: Mi esposo, mi esposo se lo ha comido un ser extraño con una sola pata.

Zue: Dónde, dónde se encuentra

Mujer: Huí de ella, se encuentra merodeando por el cementerio atravesando la hacienda.

Cuando Zue por fin llega al cementerio es atacado por varios grupos de espantos, murciélagos y esqueletos vivientes. Después de eliminar a sus enemigos avanza hasta un claro donde ve un extraño animal, devorando a un hombre aun con vida.

De un momento al otro solo parece ser una bella mujer:

Dialogo.
Mujer: por favor ayúdame un extraño ser me esta intentando hacer daño.

Zue: ¿por dónde se fue?

Mujer: Acompáñame esta dentro del bosque.

Zue: Eso no es cierto dime que haces con las personas que secuestras

La mujer sufre una terrible transformación se convierte Patasola

Patasola: Me lo como, tu serás mi cena
Zue: Pagarás por todo lo que has hecho

Enfrentamiento

FIN DE LA ESCENA.

4.9.4. Escena Cuatro

Ubicación :
Pacífico
Jefe :
Madreagua
Fondo :
Selvas, rió, caseríos, manglares.

Descripción: Inicia con la lectura de la libreta de notas del padre donde se encuentra el siguiente mensaje:

“Una noche de lluvia en lo profundo del pacifico en las riveras del río Atrato, escuche un suave sonido, el arrullo de una madre para su hijo perturbó el ambiente… era el sonido que los pobladores le atribuían a un misterioso ser que viene de la profundidades del agua... la madreagua”.

He tenido un pensamiento, tal ves la madreagua, quien conoce el dolor de perder a sus seres queridos, pueda ayudarme a buscar mi esposa a lo largo de la nación… ¡Noo de que estoy hablando!!!, me estoy volviendo loco!.”

Zue ha emprendido el viaje a la región pacifica, no sabemos que nuevas aventuras le esperan. Durante el viaje en avión Zue puede divisar las nubes oscuras y densas que la rodean, largos ríos y una extraña sensación de soledad en el ambiente.

Tras un aterrizaje un poco turbulento emprende su rumbo hacia Serranía del Darién, viajando en una chalupa a lo largo del rió atrato, para finalmente se dirige a la población de Arquía a doce quilómetros del río en donde comenzará esta nueva aventura, en busca de la madreagua.

a) Flujo y detalles de la escena:

PARTE UNO: RIO ATRATO, RUMBO A ARQUÍA:

Zue desciende de la avioneta en un aeropuerto improvisado, toma un jeep hasta puerto libre, donde toma una chalupa para iniciar su viaje hacia el tapón del Darién.

A lo largo del camino deberá esquivar algunos obstáculos, principalmente derivados por el clima, podrá hacer paradas en algunas casas con el fin de conseguir energía y algunos consejos para las próximas partes de la escena.

Lugares clave:

Puerto (Tienda): En este lugar se debe comprar un tiquete para poder viajar, adicionalmente algunas herramientas importantes

- Ancla.
- Impermeable.
- Mapa.
- Termo
- Destilador
- Fósforos
-
-

En este puerto se compran semillas de manglar.

Casa de Familia: En ella vive una familia de 4 personas quienes no tienen mucho que decir, sin embargo es un buen lugar para pasar tiempo mientras no sea de noche ya que en la noche no atienden a nadie, no hay energía ni nada similar… por el momento.

Miembro de la familia: Puedes esperar acá mientras pasa alguna tormenta.
Zue: Gracias.

Posada de paso: Solo hay una mujer que no dice nada, lo único que hace es barrer y barrer, al parecer no hay nadie más en la casa, solo tres perros y un corral de gallinas, ropa tendida en una cuerda, recipientes, vidrios etc.

Si se ingresa de día no hay nada en la casa pero al ingresar entre las 6:00 PM y las 03:00 AM aparecerá el duende burlón que hace travesuras en las casas, destiende la ropa, ahuyenta a los perros y a las gallinas y le gusta romper cosas en general.

Zue: ¿¡ oh Dios mió que es esto!?

Duende: ñejejejejeñeeenmejeje, (risa chillona)

El duende sigue destruyendo la casa, espantando los animales y haciendo maldades.

Tras unos segundos se detiene en frente de Zue y le dice:

Duende: Que animal tan extraño, nunca había visto un animal tan feo como tu.
Zue: Que queeeeeee!!!!

Duende: ahhhhhhhhhhhhhhhhhhhhhhhhhhhhhh Un animal que habla auxilió!!!

Zue: Todos las personas hablamos amigo…

Duende: ¿Personas? Creí que eran un mito de los duendes antiguos…

Zue: ¿Quién eres Tú?

Duende: Soy alguien, los duendes me llaman Edad.
Zue: ¿Y porque estas haciendo tantas maldades?

Duende: Por que esto se ve horrible, siempre lo arreglo todas las noches pero al día siguiente vuelve a quedar todo en su sitio… ¡que horrible! Es una pesadilla.

Zue: Ya veo… jajaja

Duende: Me caes bien, si me traes el hechizo con el que desordenan la casa te daré algo muy especial…

Zue: Esta bien… pero llámame Zue.

Duende: De acuerdo. –umm Nunca había visto un Zue... por un momento llegue a creer que era un humano--

En ese momento amanece.

Al volver a hablar con la dueña de la casa esta vez si responde.

Dueña de la casa: Jovencito ayúdame a limpiar.

Si Zue no acepta no pasa nada, pero si acepta:

Zue: Te ayudare.

-- La dueña de la casa le entrega la escoba a Zue, si se conserva la escoba hasta que caiga la noche, el duende la aceptara como el hechizo para limpiar.

Zue: Acá te traigo el hechizo para ordenar la casa.
Duende: ¿Ese aparato tan feo es un hechizo?, muéstrame como funciona.
En ese momento Zue comienza a limpiar toda la casa.

Zue (Cansado): Así funciona…

Duende: Muy bien zue, como premio te entregare estos polvos de mangle antiguo hechizados, te servirán para mantenerte seco, además te entregaré esta bolsita con una sorpresa, claro que tiene un cerrojo especial que solo te permite abrirla en las noches de luna llena.

Zue se puede retirar de la casa y continúa su rumbo por el río, se inicia una fuerte tormenta, para que no le haga daño la tormenta, Zue deberá usar un impermeable, en un momento determinado del viaje al cesar la tormenta, aparece un enjambre de mosquitos gigantesco, el polvo del duende solo mata algunos, y algunas cosas como el machete etc. no sirven para nada, si se conserva la escoba y se compraron fósforos en el puerto se puede hacer una hoguera para facilitar el paso, otra opción es haber comprado el ancla, para estacionar la balsa justo antes de llegar con los mosquitos, quienes al cabo de 2 horas (en tiempo real serían solo unos minutos) se irían.

Después se continúa el tramo suavemente hasta llegar a Arquía.

PARTE 2: LLEGADA A ARQUIA

Al inicio es el tramo final del rió, justo a doce kilómetros de Arquía, el recorrido es una zona de gruesa vegetación con altos niveles de humedad, atravesar esta espesa selva es la misión de esta escena, no será fácil pues en el camino se encontrarán varios obstáculos como animales, insectos obstáculos geográficos etc.

Lugares clave:

Casa pobre: Este lugar es verdaderamente pobre, allí encuentra una familia hambrienta puesto que su padre era quien trabajaba y les brindaba el sustento, desafortunadamente se encuentra muy enfermo y hace varios días que no tienen nada para comer.

Penélope:por favor Señor ayude a mi esposo, esta muy mal de salud y no tenemos dinero para pagar un medico ni para comparar medicina, mucho menos para alimentarlo.

Zue: claro pero… ¿que puedo hacer?

Penélope: Hay una hongo muy especial en lo profundo de la selva, es de color azul, puedo preparar un medicamento para mi padre con eso, pero necesito más o menos 10 hongos.

Zue: ¡Iré a buscarlo de inmediato!

Penélope: ¡Espera!, puede ser muy peligroso, lleva esto para defenderte.. recuerda que el hongo es muy venenoso al tacto.

Penélope le entrega una garra metálica a Zue.

Ya en el bosque Zue debe buscar en los árboles que estén en descomposición el extraño hongo, la mejor forma de buscarlos es dándole varios machetazos a los árboles par que quede visible su interior, si se utiliza la garra esta se queda enterrada en el árbol y no hace nada, Zue solo puede cargar 1 hongo a la vez, ya que son muy venenosos, pero si los rocía con polvo de mangle el hongo se reseca y pierde su propiedad venenosa, de esta manera puede cargar todos los hongos que quiera, de paso en el bosque también se enfrentara con mosquitos y jaguares.

Al regresar a la casa Zue le entrega los Hongos a Penélope.

Zue: Ya tengo los hongos, tómalos.

Penélope: Gracias.

Prepara la Poción y se la entrega a su esposo quien se cura de inmediato.
Carlos (esposo):¡Gracias forastero!

Zue: Para nada, lo hago con gusto.

Carlos: Que haces por estas tierras?

Zue: He venido en busca de la madreagua.

Carlos: Es muy peligroso realizar ese viajes solo, como agradecimiento te acompañaré en tu travesía, estaré siempre cuando lo necesites.

5. ELABORACIÓN DE BOCETOS

Con base en el documento de la historia se empiezan a generar los bocetos de los personajes, las características específicas tales como edad, altura, contextura, ropa, accesorios, etc. se moldean en esta fase.

Los primeros bocetos, tenían muy marcado el estilo Manga*, esto se modifico para dar paso a un estilo de dibujo más original, sin desmeritar la calidad del dibujo en el Manga.
Ver Anexo H. Primeros Bocetos

En la elaboración de los conceptos ilustrados de los personajes se tuvieron en cuenta todas las opciones propuestas incluyendo algunas no muy detalladas. Ver Anexo E. Evolución Ilustraciones de Zue.

Figura 4. Boceto de Zue

Fuente: Los Autores

* Manga es el arte japonés de dibujo empleado en tiras cómicas.

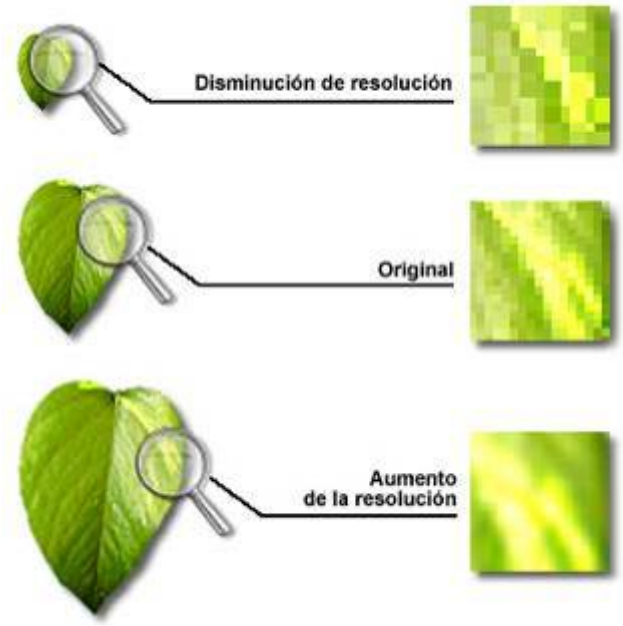
5.1. TÉCNICAS DE MUESTREO DE IMÁGENES

Estas técnicas se utilizan para cambiar el tamaño en píxeles y por lo tanto la resolución de la imagen. Si se disminuye o aumenta el tamaño de la imagen, se modifica la cantidad de píxeles utilizados en la misma.

Estas son las técnicas estándar:

- Por Aproximación: Es la técnica más rápida debido a que no utiliza ningún método para suavizar la imagen. Es empleada para conservar bordes de imagen duros o irregulares. Las imágenes muestreadas con esta técnica tienen un tamaño de archivo menor.
- Bilineal: Se utiliza para obtener una calidad media de imagen.
- Bicúbica: Esta técnica permite obtener los mejores resultados de calidad de imagen, debido a que maneja degradado de tonos y suavizado de bordes con colores de canal alpha. La modificación de los parámetros de esta técnica permite hacer más precisos los cambios de tamaño de la imagen, bicúbica más suavizada para aumentar la imagen y bicúbica más enfocada para reducir la imagen.

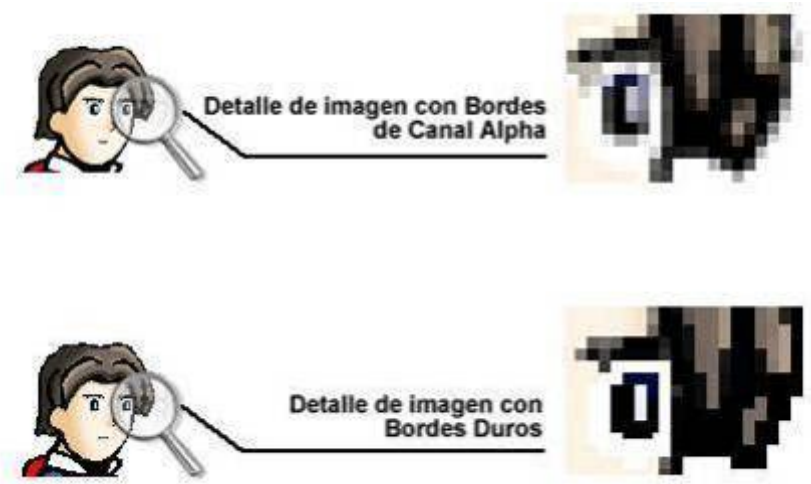
Figura 5. Imágenes redimensionadas



Fuente: Los Autores

Debido a que haber implementado soporte para el canal alfa en el engine habría tomado demasiado tiempo, se optó por utilizar la técnica de aproximación para muestrear las imágenes a la escala apropiada, para ser utilizados en el juego y evitar que dichas imágenes tuvieran bordes suavizados.

Figura 6. Bordes de Imagen



Fuente: Los Autores

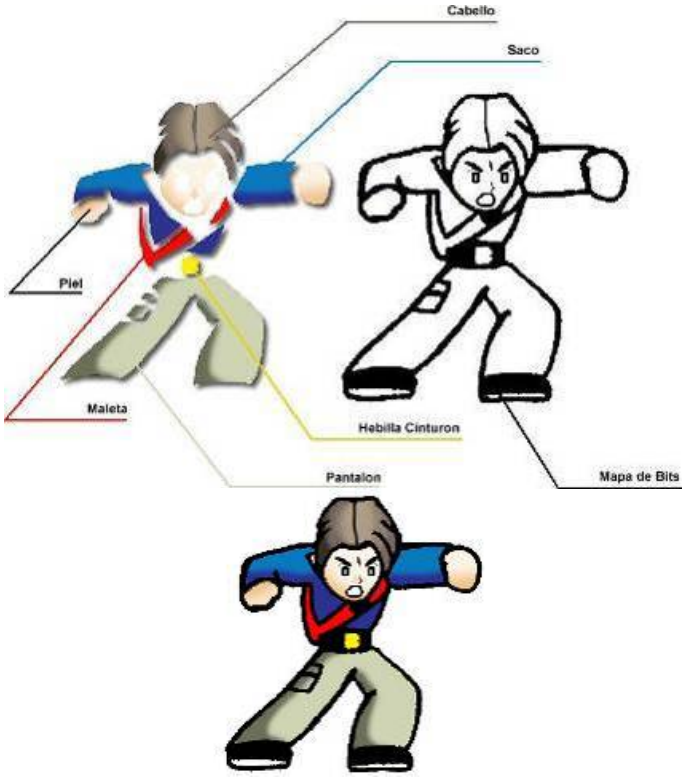
5.2. CAPAS

Las capas permiten trabajar con un elemento de una imagen sin modificar los otros. Se puede pensar en las capas como si fuesen acetatos organizados uno encima de otro.

La aplicación del color en las imágenes del juego se hizo creando una capa cada elemento de la imagen (Cabello, Pelo, Piel, Ropa, Calzado, etc.), lo que hace más fácil su aplicación.

La utilización de capas permite aplicar color a varios elementos pertenecientes a imágenes distintas.

Figura 7. Aplicación de Color por medio de Capas



Fuente: Los Autores

5.3. SOMBRAS

La implementación de las sombras se hace utilizando las capas que ya tienen el color base, para pintar sobre ellas el color respectivo de sombra.

Se utiliza el pincel con estilo aerógrafo, para obtener un mejor resultado de fusión entre la sombra y el color base debido a que este dispersa el color de forma uniforme.

5.4. ELABORACIÓN DE GRÁFICAS TABLERO DE HISTORIA

Los tableros de historia tienen por objeto dibujar un borrador de los elementos que conformaran las escenas.

La calidad de las graficas es irrelevante debido a que el tablero de historia es esencialmente una guía informativa para la planeación de los escenarios, se diseña en papel pergamino cada una de las escenas y sus respectivas capas que serán implementadas en el juego, incorporándole los elementos tales como árboles, casas, personajes, enemigos, etc. en los lugares específicos en que aparecerán en la escena.

Con esto se definen las acciones que debe ejecutar el personaje principal, lugares con eventos y sitios de colisión, ver Anexo F. tablero de historias.

6. MODELADO DEL SOFTWARE DE VIDEOJUEGO

6.1. DEFINICIÓN DE REQUERIMIENTOS

6.1.1. Requerimientos candidatos

- Desarrollo de un videojuego que involucre aspectos de la mitología colombiana
- Debe estar dirigido a una población adolescente, de trece años en adelante
- Debe contar con una interfaz gráfica amigable y configurable
- Debe contar con las siguientes opciones: (Configuración, juego, mitología)
- Debe ser posible guardar los avances realizados en el juego
- Debe ser un juego de roll, es decir el jugador toma el papel del personaje principal e interactúa con su entorno
- Debe contar con animaciones introductorias al inicio del juego y en la mayoría de las escenas.
- Debe existir una interfaz de ayuda interactiva
- Los diálogos deben estar en idioma español y ofrecer soporte futuro para otros idiomas
- La interfaz de juego de ser 2D
- Debe ser optimizado para correr bajo maquinas Pentium III 450Mhz, tarjeta de video de 32 bits y control genius G08
- La resolución del juego será de 640*480 y debe soportar modo de ventana y modo full screen
- Que sea innovador
- Que cuente con una interfaz de control manejable en cada una de sus modalidades
- Que sea acto para expertos e inexpertos
- Que posea una pantalla de entrenamiento
- Que permita introducir cheats
- Que posea diálogos entre los personajes y en la narración permita ejecutar <skip>
- Que exista una pagina Web con información relevante al juego
- Crear diferentes souveniers de juego: (protector de pantalla, papel tapiz, música)

6.1.2. Clasificación de requerimientos

6.1.2.1. Requerimientos funcionales

- Debe contar con las siguientes opciones: (Configuración, juego, mitología)
- Debe contar con una interfaz gráfica amigable y configurable
- Debe ser posible guardar los avances realizados en el juego
- Debe ser un juego de roll, es decir el jugador toma el papel del personaje principal e interactúa con su entorno
- Los diálogos deben estar en idioma español y ofrecer soporte futuro para otros idiomas
- La interfaz de juego de ser 2D
- Debe ser optimizado para correr bajo maquinas Pentium III 450Mhz, tarjeta de video de 32 bits y control genius G08
- La resolución del juego será de 640*480 y debe soportar modo de ventana y modo full screen
- Que cuente con una interfaz de control manejable en cada una de sus modalidades
- Que posea diálogos entre los personajes y en la narración permita ejecutar <skip>
- Desarrollo de un videojuego que involucre aspectos de la mitología colombiana

6.1.2.2. Requerimientos no funcionales

- Debe estar dirigido a una población adolescente, de trece años en adelante
- Debe contar con animaciones introductorias al inicio del juego y en la mayoría de las escenas.
- Debe existir una interfaz de ayuda interactiva
- Que sea innovador
- Que sea acto para expertos e inexpertos
- Que permita introducir cheats
- Que exista una pagina web con información relevante al juego
- Crear diferentes souveniers de juego: (protector de pantalla, papel tapiz, música)

6.1.2.3. Requerimientos no contemplados

- Que posea una pantalla de entrenamiento

6.2. DEFINICIÓN DE CASOS DE USO PARA REQUERIMIENTOS FUNCIONALES

- Debe contar con las siguientes opciones: (Configuración, juego, mitología)(Caso de uso: Configuración Aplicación)
- Debe contar con una interfaz gráfica amigable y configurable(Caso de uso: Iniciar Aplicación)
- Debe ser posible guardar los avances realizados en el juego(Caso de uso: Administrar partidas)
- Debe ser un juego de roll, es decir el jugador toma el papel del personaje principal e interactúa con su entorno
- Caso de uso: Administrador de escenarios, administrador de personajes, manejador de diálogos
- Los diálogos deben estar en idioma español y ofrecer soporte futuro para otros idiomas
- Caso de uso: manejador de diálogos
- La interfaz de juego de ser 2D (Caso de uso: Iniciar Aplicación)
- Debe ser optimizado para correr bajo maquinas Pentium III 450Mhz, tarjeta de video de 32 bits y control genius G08 (Caso de uso: Iniciar Aplicación, Jugar)
- La resolución del juego será de 640*480 y debe soportar modo de ventana y modo full screen (Caso de uso: Juego)
- Que cuente con una interfaz de control manejable en cada una de sus modalidades (Caso de uso: Iniciar Aplicación)
- Que posea diálogos entre los personajes y en la narración permita ejecutar <skip> (Caso de uso: Manejador de diálogos)
- Desarrollo de un videojuego que involucre aspectos de la mitología colombiana (Caso de uso: Diseño)

6.3. DEFINICIÓN DE CASOS DE USO PARA REQUERIMIENTOS NO FUNCIONALES

- Debe estar dirigido a una población adolescente, de trece años en adelante (Caso de uso: Diseño)
- Debe contar con animaciones introductorias al inicio del juego y en la mayoría de las escenas. (Caso de uso: Mostrar Display)
- Debe existir una interfaz de ayuda interactiva (Caso de uso: Ayuda)
- Que sea innovador (Caso de uso: Diseño)
- Que sea acto para expertos e inexpertos (Caso de uso: Diseño)
- Que permita introducir cheats. (Caso de uso: Juego)
- Que exista una página web con información relevante al juego (Caso de uso: -0-)
- Crear diferentes souveniers de juego: (protector de pantalla, papel tapiz, música) (Caso de uso: -0-)

6.4. CLASIFICACIÓN DE CASOS DE USO

Caso de Uso	Prioridad
• Iniciar aplicación	Primario
• Mostrar displays	Opcional
• Mostrar menú principal	Secundario
• Mostrar menú de configuración	Secundario
• Configuración sonido	Secundario
• Configuración video	Opcional
• Configuración idioma	Opcional
• Configuración entrada	Secundario
• Administrador de partidas	Primario
• Iniciar partida nueva	Primario
• Continuar partida	Primario
• Eliminar partida	Primario
• Selector de escenas	Primario
• Jugar	Primario
• Mostrar menú de ítems	Secundario
• Barra de estado	Primario
• Salir de escena	Secundario
• Administrador de Escenas	Primario
• Administrador de personajes	Primario
• Cargar ítems	Secundario
• Trucos	Opcional
• Manejador de Diálogos	Primario
• Ayuda	Opcional
• Pantalla de entrenamiento	Opcional
• Texto ayuda	Primario

6.5. DESCRIPCIÓN DE CASOS DE USO EN FORMATO ESPECIAL

6.5.1. Casos de uso primarios

6.5.1.1. Iniciar aplicación

Nombre:	Iniciar Aplicación
Actores:	Jugador(iniciador)
Propósito:	Cargar la aplicación

Descripción: Solicitar al sistema operativo los recursos que necesita la aplicación, es decir, crear una interfaz para utilizar los dispositivos de video, sonido y entrada. Carga la configuración de juego muestra el intro.
Precondiciones: Instalación exitosa
Curso de Eventos
Actor(Acción)

1. Hace doble clic en el icono, .exe.lnk

Sistema (Respuesta)

2. Carga en memoria la aplicación
3. Carga interfaz de dispositivos
4. Muestra Intro del juego

Poscondiciones:Elementos completos para iniciar el juego. Intro del juego
Cursos Alternativos
Precondiciones

Poscondiciones:Elementos completos para iniciar el juego. Intro del juego

Excepciones:No cumple requisitos del sistema

Dispositivos en conflicto

Intro no puede ser visualizado

6.5.1.2. Administrar partidas

Nombre:Administrar Partidas
Actores:Jugador (iniciador), filesystem
Propósito:Proporcionar la utilidad de administración de partidas nuevas y anteriores; debe permitir guardar, copiar, eliminar, renombrar, crear.
Descripción: El administrador de partidas almacena las características del nivel de avance en una partida, como los escenarios, personajes e ítems
Precondiciones:
Curso de Eventos:
Actor (Acción)

1) En el menú principal se elige la opción Jugar

Sistema (Respuesta)

2) muestra en pantalla el administrador de partidas con sus opciones.
Crear Nuevo
Cargar
Eliminar
Copiar
Renombrar

Actor (Acción)

3) Elige crear nuevo y proporciona los datos que se solicitan

Sistema (Respuesta)

4) Captura los datos de la partida
5) Añade al archivo de partidas una estructura de juego

6) Muestra el display de Inicio del juego
7) Muestra el Mapa de escenas

Poscondiciones:Se muestra el Mapa de escenas

Cursos Alternativos:

En paso 3
Precondiciones:La partida a cargar existe
Actor (Acción)
3) Elige cargar
Sistema (Respuesta)
4) Busca la partida solicitada en el archivo de partidas
5) lee y carga la estructura de juego para la partida
6) Muestra el Mapa de escenas
Poscondiciones: Se muestra el Mapa de escenas con el nivel de avance grabado

En paso 3
Precondiciones:La partida a eliminar existe
Actor (Acción)
3) Elije Eliminar
Sistema (Respuesta)
4) Busca la partida solicitada en el archivo de partidas
5) Borra la estructura de la partida del archivo de partidas
6) vuelve al administrador de partidas
Poscondiciones:Se muestra el Administrador de partidas sin la partida eliminada

En paso 3
Precondiciones:La partida a copiar existe
Actor (Acción)
3) Elije Copiar
Sistema (Respuesta)
4) Busca la partida solicitada en el archivo de partidas
5) Genera una copia de la estructura de juego
6) vuelve al administrador de partidas
Poscondiciones:Se muestra el Administrador de partidas con la partida copiada

En paso 3
Precondiciones: La partida a renombrar existe
Actor (Acción)
3) Elije Renombrar y proporciona los datos que se solicitan
Sistema (Respuesta)
4) Busca la partida solicitada en el archivo de partidas
5) cambia el nombre a la estructura de la partida
Poscondiciones: La partida tiene otro nombre
Excepciones: Errores de lectura y escritura de archivos

6.5.1.3. Jugar

Nombre:Jugar
Actores:Jugador (iniciador)
Propósito:Que el usuario pueda interactuar con el juego
Descripción: El jugador comienza a interactuar con el juego, ingresa los comandos para moverse dentro de la escena
Precondiciones:Partida cargada exitosamente
Curso de Eventos
Actor (Acción)
1) se elige una escena en el mapa
Sistema (Respuesta)
2) Cargar escena ref: Adm. Escenarios
3) Cargar personajes ref: Adm Personajes
Actor (Acción)
4) entrar comandos
Sistema (Respuesta)
5) Procesar entradas
6) Procesar salidas
Ref: Adm Escenarios
Ref: Adm Personajes
Actor (Acción)
7) Repetir 4 hasta evento de salida
Sistema (Respuesta)
8) volver a Mapa de escenas
Actor (Acción)
9) Selección de guardar partida
Sistema (Respuesta)
10) ref: Administrador de Partidas
11) Cargar mapa de Escenas
Actor (Acción)
12) Elige salir del mapa de escenas
Sistema (Respuesta)
13) Se muestra menú principal
Actor (Acción)
14) Elige salir de aplicación
Sistema (Respuesta)
15) Devuelve los recursos utilizados
Poscondiciones:Interacción con el jugador
Cursos Alternativos

En paso 9
Precondiciones:Escena anterior superada
Actor (Acción)
9) elige una nueva escena para jugar
Poscondiciones:Se continúa con el caso de uso
Excepciones:Errores de entrada y salida

6.5.1.4. Ayuda

Nombre:Ayuda
Actores:Jugador (iniciador)
Propósito:Presentar una al jugador la manera de jugar
Descripción: Se presenta una ayuda en la que se explica todos los aspectos de administración de la aplicación
Precondiciones
Curso de Eventos
Actor (Acción)
1) Elige Ayuda
Sistema (Respuesta)
2) Carga la interfaz de ayuda
Actor (Acción)
3) Se mueve a través de la ayuda
4) Se cierra la ayuda
Sistema (Respuesta)
5) Se muestra el menú principal
Poscondiciones:Menú principal
Cursos Alternativos
Precondiciones
Actor (Acción)
Sistema (Respuesta)

Poscondiciones

Excepciones

6.5.2. Casos de uso secundarios

6.5.2.1. Mostrar menú principal

Nombre:Mostrar menú principal
Actores:Jugador (iniciador), Motor
Propósito:Cargar el menú principal, permitiendo al usuario elegir las siguientes opciones, jugar, Configuración, ayuda y salir.
Descripción: Solicitar al Engine la carga y visualización del menú principal del juego y la selección de sus distintas opciones
Precondiciones: Juego iniciado
Curso de Eventos
Actor (Acción)
1) Iniciar la aplicación
Sistema (Respuesta)
2) Dar avance en el intro del juego
3) Mostrar en pantalla la interfaz del menú Principal.
4) Mostrar Opciones de Menú:

Jugar
Menú de Configuración
Menú de Entrada
Ayuda
Salir
Actor (Acción)
5) Dar Clic en una opción del menú
Poscondiciones:Elementos completos para iniciar y configurar el juego.
Cursos Alternativos
En Paso 5
Precondiciones:Interfaz de Menú Cargada con Éxito
Actor (Acción)
1) Elegir la opción de Jugar
Sistema (Respuesta)
2) Cargar Administrador de Partida
Poscondiciones:Elementos completos para iniciar una partida.
Cursos Alternativos

En Paso 5
Precondiciones:Interfaz de Menú Cargada con Éxito
Actor (Acción)
1) Elegir la opción de configuración
Sistema (Respuesta)
2) Cargar el Menú de Configuración
Poscondiciones:Elementos completos para configurar el juego.
Cursos Alternativos

En Paso 5
Precondiciones:Interfaz de Menú Cargada con Éxito
Actor (Acción)
1) Elegir la opción de Ayuda
Sistema (Respuesta)
2) Cargar la ayuda del juego
Poscondiciones: Elementos de ayuda para el manejo del juego
Cursos Alternativos

En Paso 5
Precondiciones:Interfaz de Menú Cargada con Éxito
Actor (Acción)
1) Elegir la opción de Salir
Sistema (Respuesta)
2) Mostrar ventana de advertencia preguntado que “Realmente deseas salir?”
2) Cerrar interfaz de dispositivos
3) Liberar la memoria al retirar la aplicación.
Poscondiciones:Entregar el control total de la máquina al Sistema Operativo

6.5.2.2. Menú de configuración.

Nombre:menú de Configuración
Actores:Jugador (iniciador)
PropósitoPresentar al jugador el menú de configuración del Descripción. Se presenta una ayuda en la que se explica todos los aspectos de administración de la aplicación
Precondiciones:menú principal
Curso de Eventos
Actor (Acción)
1) Elige Menú de Configuración
Sistema (Respuesta)
2) Carga la interfaz del menú de Configuración
3) Mostrar las opciones del menú de configuración:
Configuración de Sonido
Configuración de Video
Configuración de Idioma
Salir de la Configuración
Actor (Acción)
4) Elegir Opción
Sistema (Respuesta)
5) Cargar la opción seleccionada por el Jugador.
Poscondiciones:
Cursos Alternativos

En Paso 4
Precondiciones:menú Principal
Actor (Acción)
1) Seleccionar Opción Configuración sonido
Sistema (Respuesta)
2) Cargar la interfaz de configuración de sonido.
Actor (Acción)
3) Elegir el nivel de sonido deseado
Sistema (Respuesta)
4) Incrementar o disminuir el nivel del sonido dentro del juego.
Actor (Acción)
5) Salir de la configuración de sonido
Sistema (Respuesta)
6) Volver al menú de Configuración
Poscondiciones: Se aumenta o se disminuye los sonidos y música de fondo del juego

Cursos Alternativos:

En Paso 4
Precondiciones:menú Principal
Actor (Acción)
1) Seleccionar Opción Configuración de Video
Sistema (Respuesta)
2) Cargar la interfaz de configuración de video

Actor (Acción)

3) Elegir la resolución deseada.
Sistema (Respuesta)
4) Cambiar la resolución de pantalla por la seleccionada por el usuario
Actor (Acción)
5) Salir de la configuración de video.
Sistema (Respuesta)
6) Volver al menú de Configuración
Poscondiciones:Se modifica la resolución de video del juego a la seleccionada por el jugador

En Paso 4
Precondiciones:
Actor (Acción)
1) Seleccionar la Opción de menú de Entrada
Sistema (Respuesta)
2) Cargar la interfaz del menú de entrada
3) Mostrar las opciones de dispositivo de entrada:
Teclado
Joystick
Actor (Acción)
4) Elegir el dispositivo de entrada
Sistema (Respuesta)
5) Cargar la configuración del dispositivo.
Actor (Acción)
6) Salir de la configuración de Entrada
Sistema (Respuesta)
7) Volver al menú de Configuración
Poscondiciones:Se carga el menú de configuración del juego

En Paso 4
Precondiciones:
Actor (Acción)
1) Seleccionar Opción Configuración Idioma
Sistema (Respuesta)
2) Cargar la interfaz de configuración de Idioma
Actor (Acción)
3) Elegir el idioma deseado
Sistema (Respuesta)
4) Cambiar el idioma del juego por el seleccionado por el usuario
Actor (Acción)
5) Salir de la configuración de Idioma
Sistema (Respuesta)
6) Volver al menú de Configuración

Poscondiciones:Se carga el juego con el idioma seleccionado por el jugador

En Paso 4
Precondiciones:
Actor (Acción)
1) Seleccionar la Opción de salir de la Configuración
Sistema (Respuesta)
2) Volver al menú principal
Actor (Acción)
3) Elegir el idioma deseado
Sistema (Respuesta)
4) Cambiar el idioma del juego por el seleccionado por el usuario
Actor (Acción)
5) Salir de la configuración de Idioma
Sistema (Respuesta)
6) Volver al menú de Configuración
Poscondiciones:Se carga el menú de configuración

6.5.2.3. Cargar ítems

Nombre:Cargar Items
Actores:Jugador (iniciador)
Propósito:Cargar los ítems elegidos por el jugador
Descripción: Solicitar al sistema la carga del ítem seleccionado
Precondiciones:Carga exitosa del menú de ítems
Curso de Eventos
Actor (Acción)
1) Seleccionar el ítem deseado
Sistema (Respuesta)
2) Asociar el ítem deseado a la tecla o el botón de uso de ítems
Actor (Acción)
3) Salir del menú de Ítems
Sistema (Respuesta)
4) Restablecer la posición del personaje dentro de la escena
Poscondiciones:Cambio del ítem asociado a la tecla o botón de uso de ítems.

Cursos Alternativos

En Paso 4
Precondiciones
Actor (Acción)
1) Seleccionar el ítem de alimento o cura.
Sistema (Respuesta)
2) Incrementar el nivel de energía del personaje
Actor (Acción)
3) Salir del menú de ítems
Sistema (Respuesta)
4) Restablecer la posición del personaje dentro de la escena
Poscondiciones:Incremento del nivel de energía del personaje

Excepciones:No existen ítems cargados en el menú de ítems

El nivel de energía del personaje esta al máximo y no se permite incrementar más

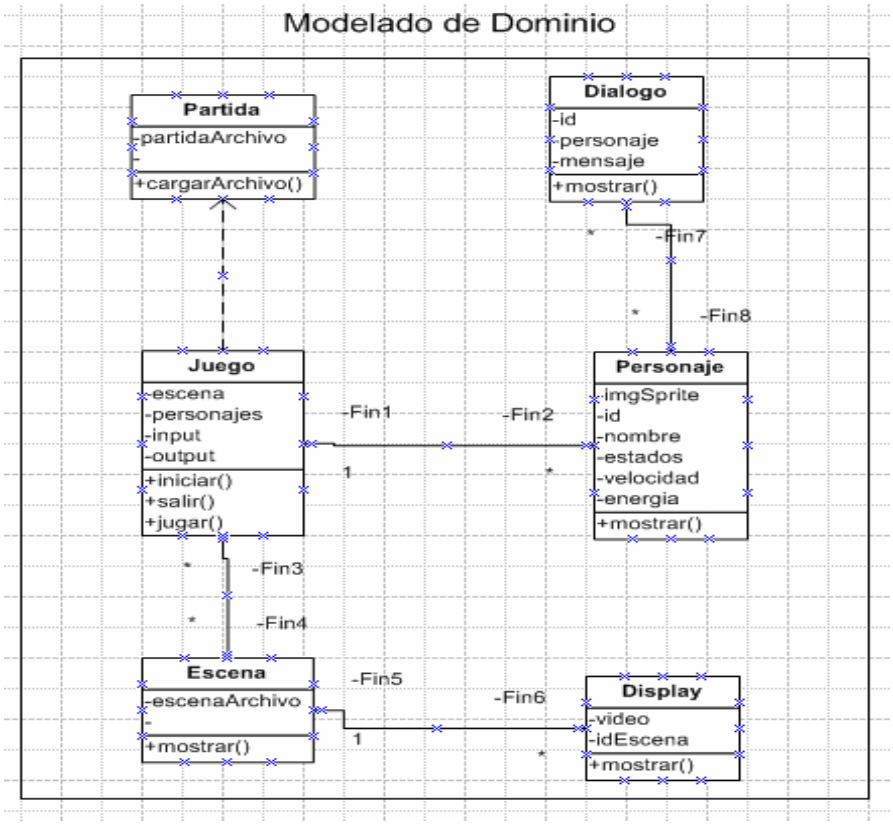
6.5.2.4. Mostrar menú de ítems

Nombre:Mostrar menú de ítems
Actores:Jugador (iniciador)
Propósito:Cargar el menú de ítems
Descripción: Solicitar al sistema la carga del menú de ítems
Precondiciones:Carga exitosa de una partida
Curso de Eventos
Actor (Acción)
1) Dentro de una escena oprimir la tecla enter en el teclado o el botón equivalente en el Joystick
Sistema (Respuesta)
2) Detener el avance del personaje dentro de la escena.
3) Carga interfaz del menú de ítems
4) Mostrar los distintos Ítems que se tienen en el momento
Actor (Acción)
7) Salir del menú de Ítems
Sistema (Respuesta)
8) Restablecer la posición del personaje dentro de la escena
Poscondiciones: Cambio del ítem asociado a la tecla o botón de uso de ítems.
Cursos Alternativos
Se deja el menú de ítems abierto hasta que el jugador selecciona un ítem o seleccione la opción de salir del menú
Excepciones:No existen ítems cargados en el menú de ítems

7. MODELADO UML DEL VIDEOJUEGO

7.1. MODELADO DE DOMINIO

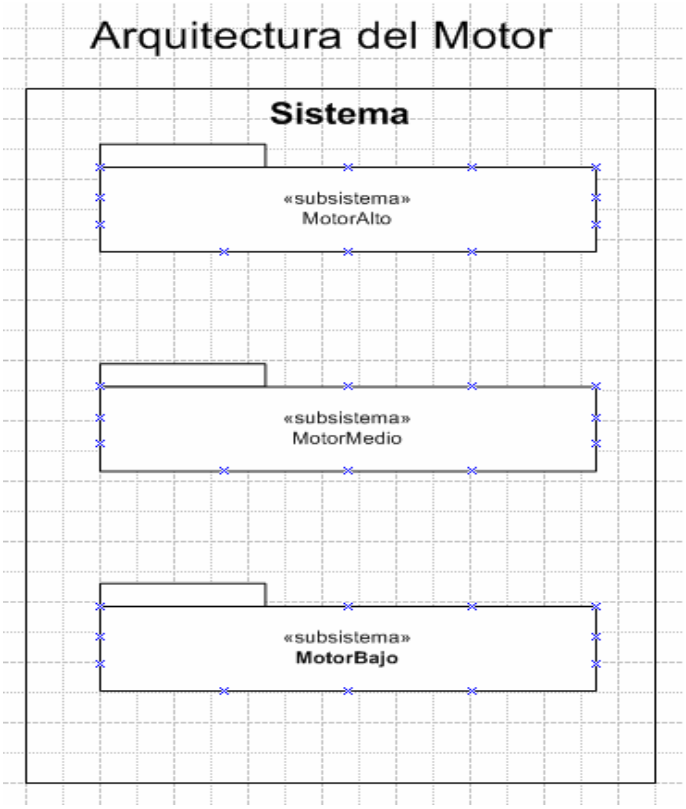
Figura 8. Modelado de Dominio



Fuente: Los Autores

7.2. ARQUITECTURA DEL MOTOR

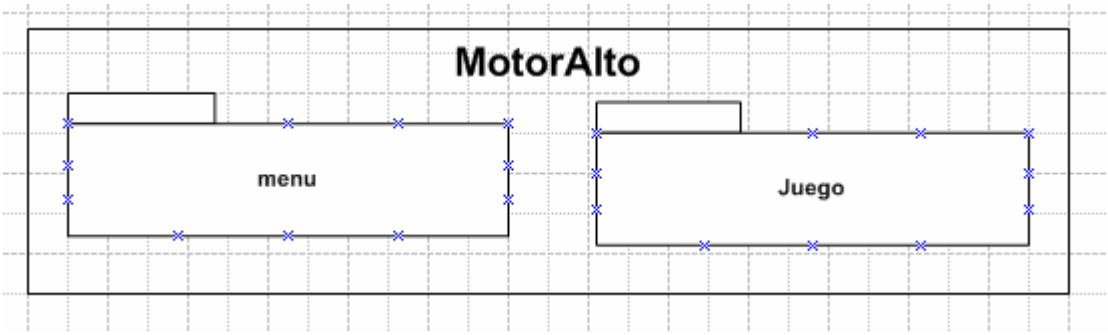
Figura 9. Arquitectura del motor



Fuente: Los Autores

7.3. MOTOR ALTO

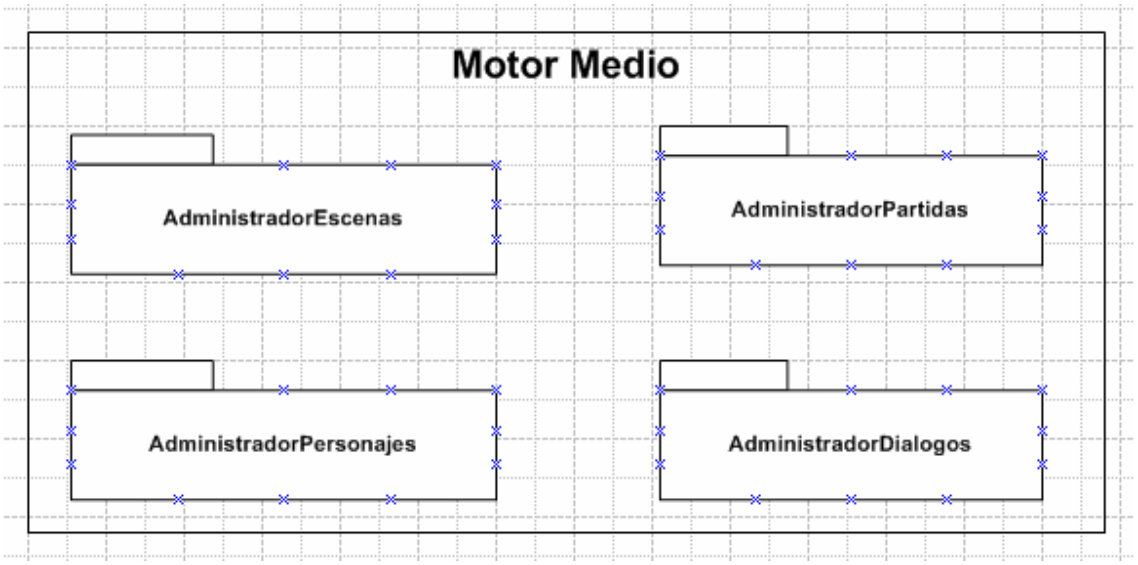
Figura 10. Motor alto



Fuente: Los Autores

7.4. MOTOR MEDIO

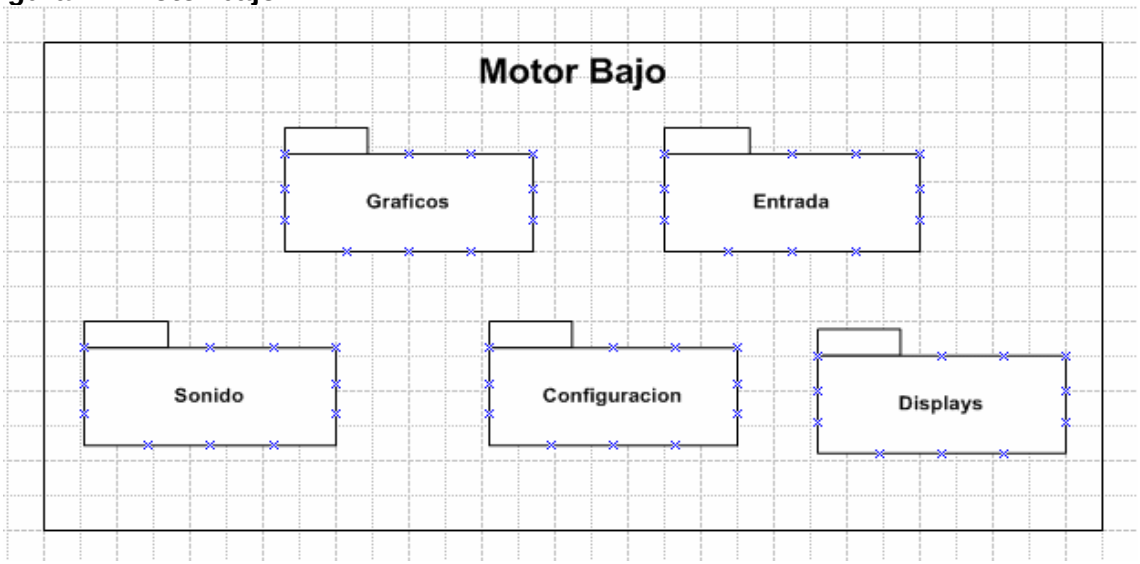
Figura 11. Motor medio



Fuente: Los Autores

7.5. MOTOR BAJO

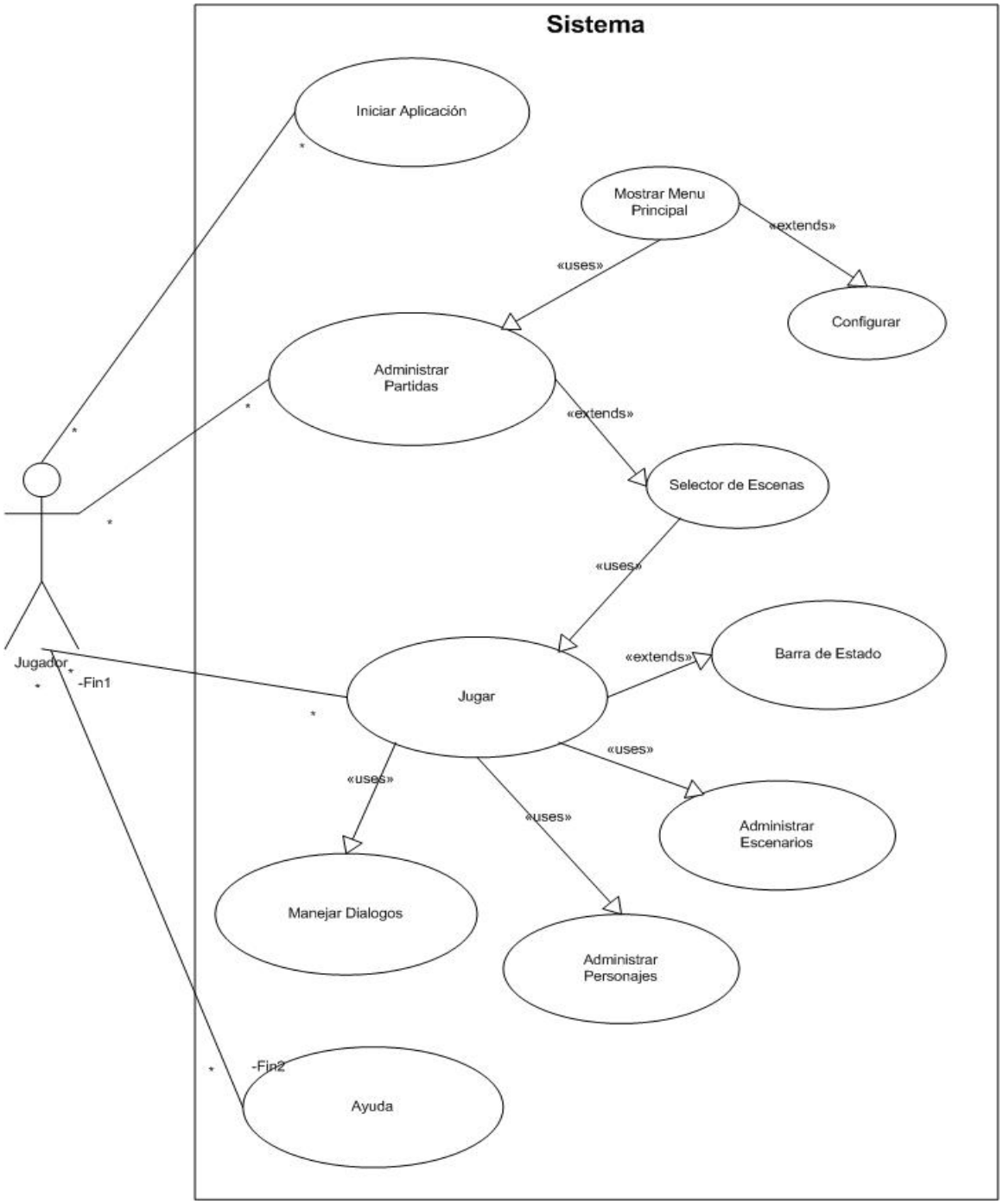
Figura 12. Motor bajo



Fuente: Los Autores

7.6. MODELO DE CASOS DE USO

Figura 13. Motor bajo



Fuente: Los Autores

7.7. PLAN DE ITERACIONES PARA EL SISTEMAS

7.7.1. Iteración 1 Integración de Estados en subsistemas.

- Modelo de estados entre subsistemas
- Modelo de flujos de mensajes entre subsistemas
- Modelado de Prototipo de mensajes
- Implementación de Prototipo de Estados
- Implementación de Prototipo de Estados con directx

7.7.2. Iteración 2 Cargue de Escenario

- Desarrollo de Caso de Uso Administrador de Escenarios
- Modelado de Prototipo de Cargue Mapa de Partidas
- Implementación de Prototipo de Cargue Mapa de Partidas
- Modelado de Prototipo de Cargue Escena
- Implementación de Prototipo de Cargue Escena

7.7.3. Iteración 3 Cargue de Personajes

- Desarrollo de Caso de Uso Administrador de Personajes
- Desarrollo de Caso de Uso Administrador de Diálogos
- Modelado de Prototipo de Cargue de Personajes a Escena
- Implementación de Prototipo de Cargue de Personajes a Escena
- Modelado de Prototipo de Cargue de Diálogos a Personajes
- Implementación de Prototipo de Cargue de Diálogos a Personajes

7.7.4. Iteración 4 Integración Escena Completa

- Desarrollo de Caso de Uso Jugar
- Integración de Casos de Uso Administrador de Escenarios, Administrador de Personajes, Administrador de Diálogos
- Implementación de Prototipo Escena Completa

7.7.5. Iteración 5 Cargue de Partida

- Desarrollo de Caso de Uso Administrador de Partidas
- Modelado de Prototipo de Cargue de Juego
- Implementación de Prototipo de Cargue de Juego

7.7.6. Iteración 6 Depuración de subsistemas

- Refinamiento de Integración entre subsistemas

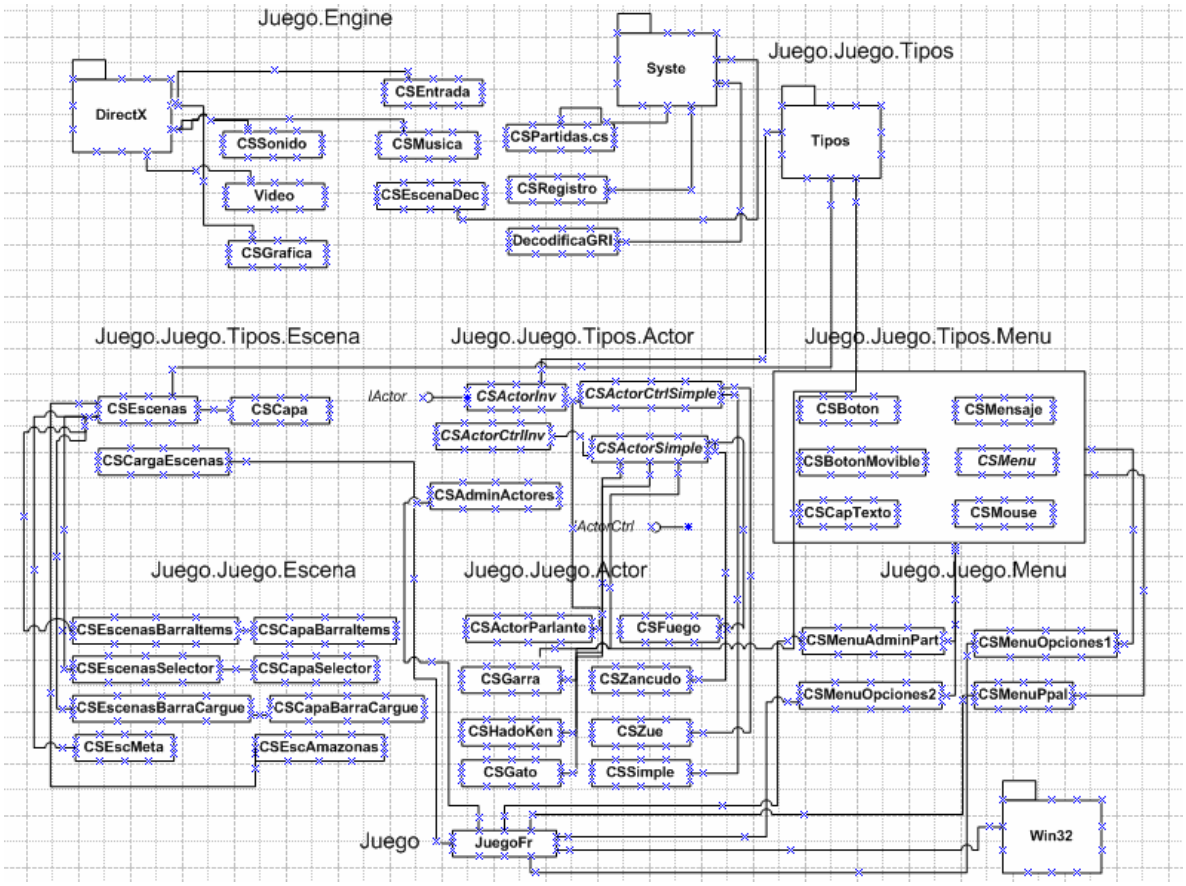
7.7.7. Iteración 7 Casos de Uso Secundarios

- Desarrollo de Caso de Usos Mostrar menú principal, Mostrar menú de configuración, Mostrar menú de ítems, Cargar ítems
- Implementación de Prototipo de Casos de Uso Secundarios

7.7.8. Iteración 8 Refinamiento, Optimización y Pruebas

7.8. DIAGRAMA DE CLASES

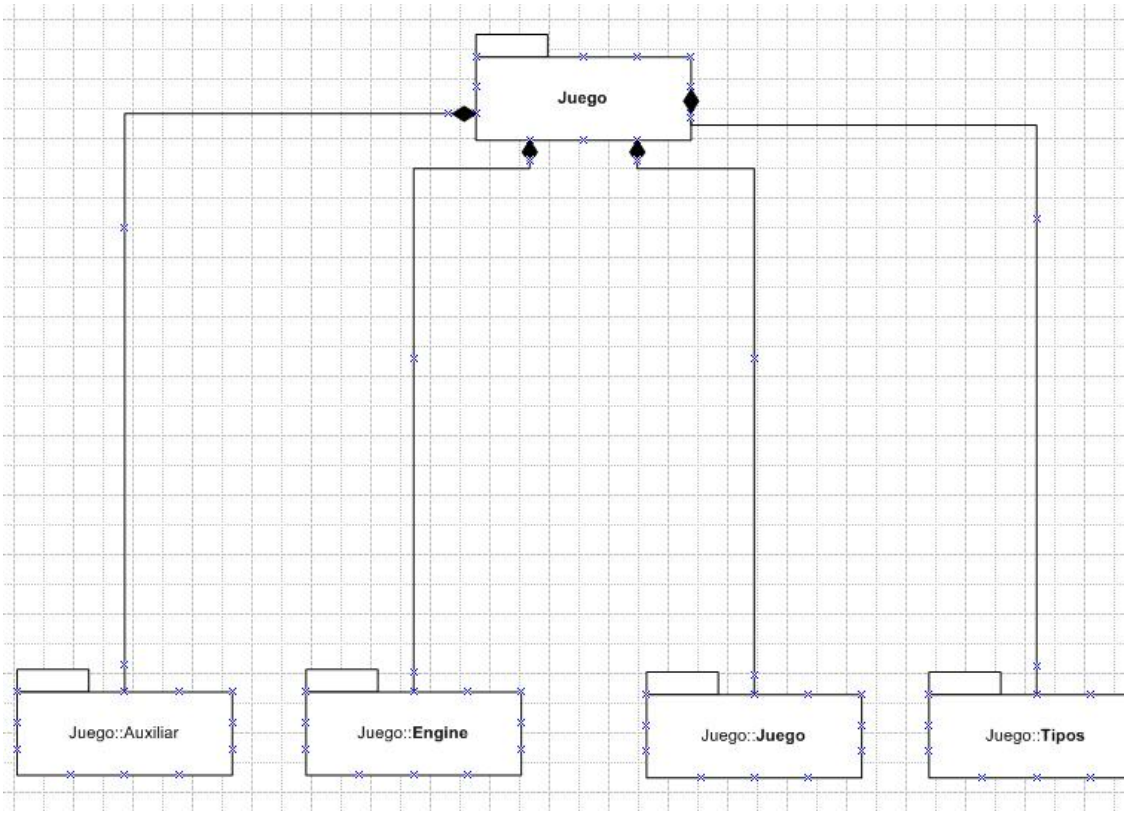
Figura 14. Diagrama de clases



Fuente: Los Autores

7.9. DIAGRAMA DE PAQUETES

Figura 15. Diagrama de paquetes

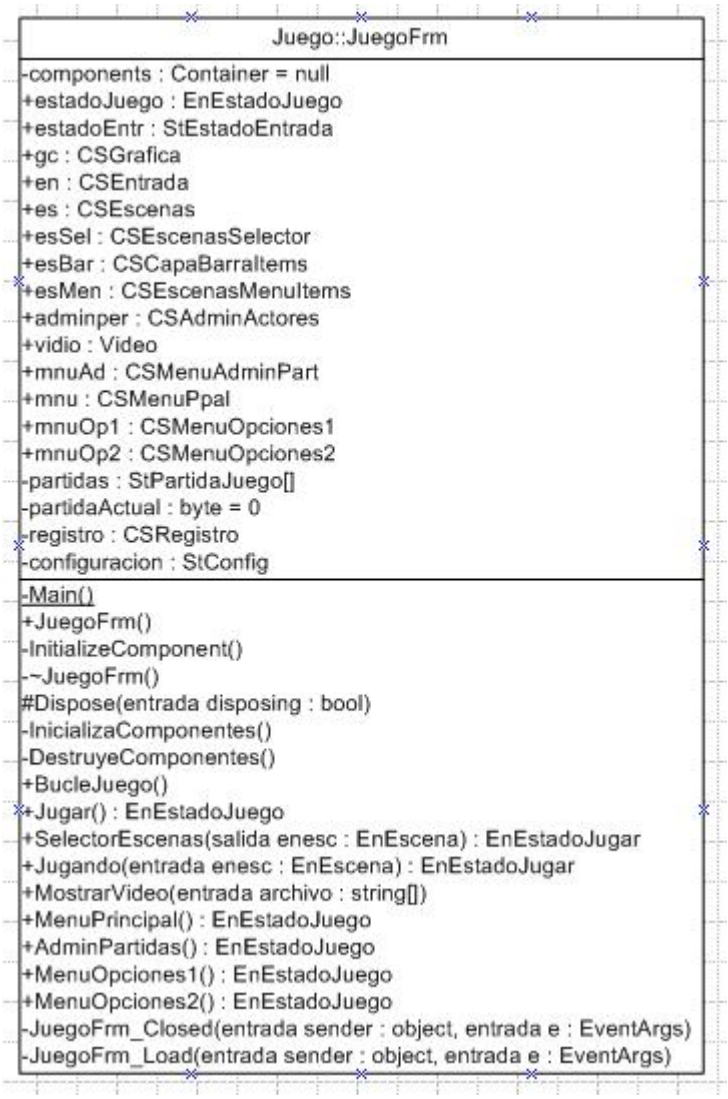


Fuente: Los Autores

7.10. CLASES

7.10.1. Juego.

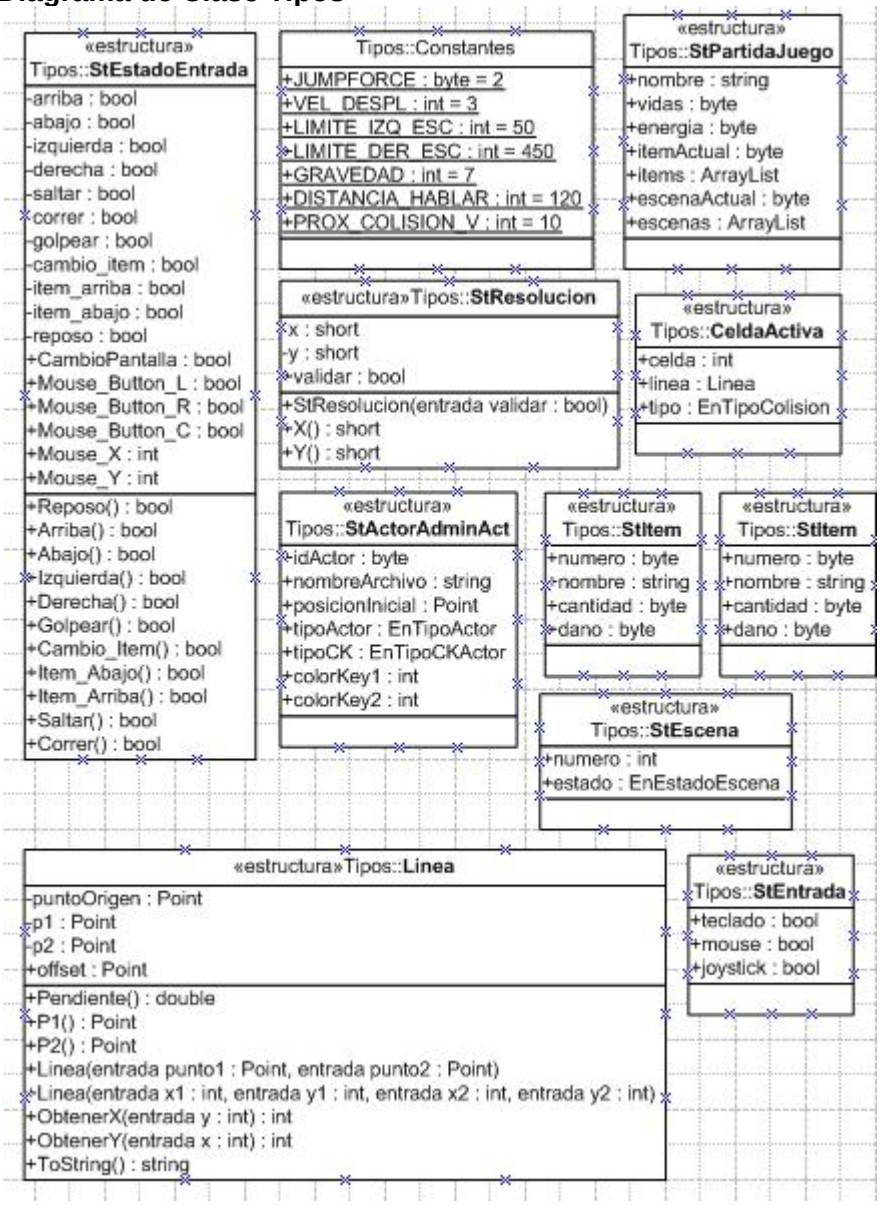
Figura 16. Diagrama de Clase JuegoFrm



Fuente: Los Autores

7.10.2. Juego.Tipos.

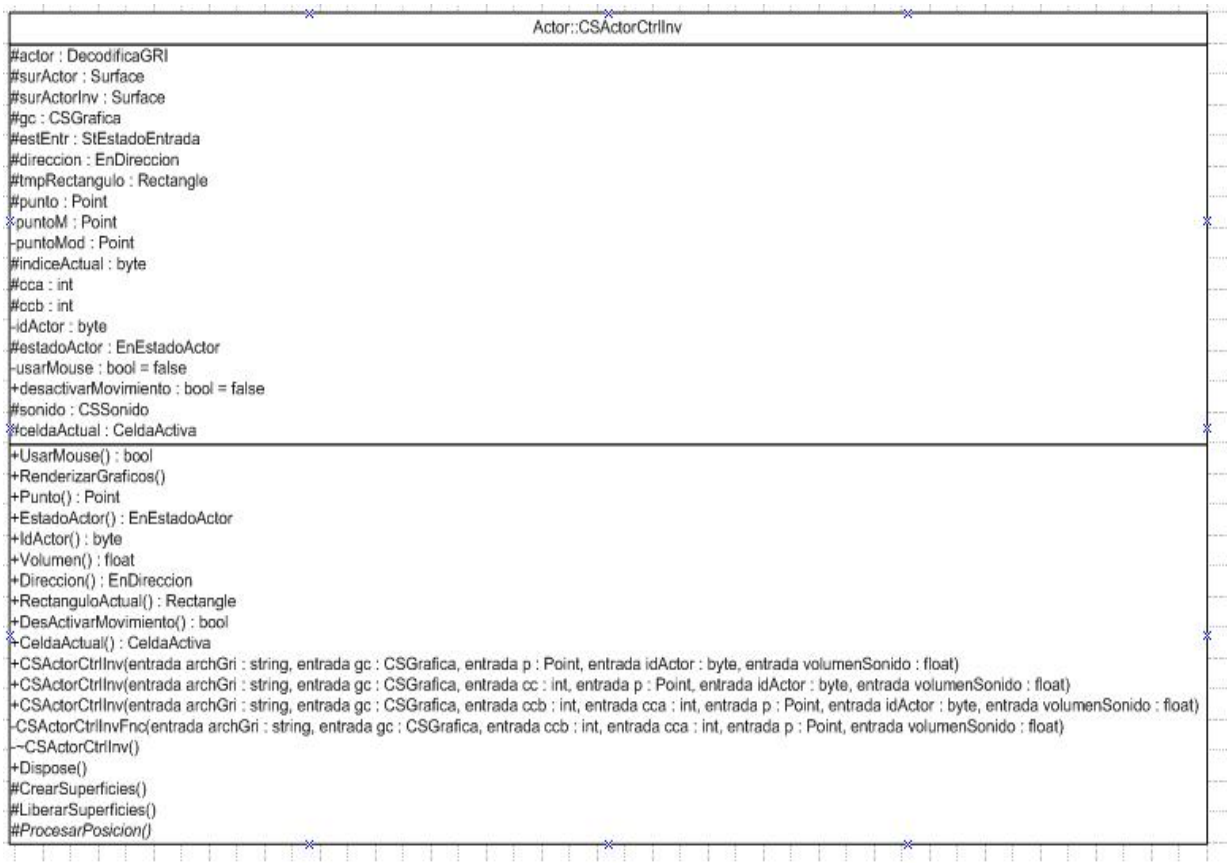
Figura 17. Diagrama de Clase Tipos



Fuente: Los Autores

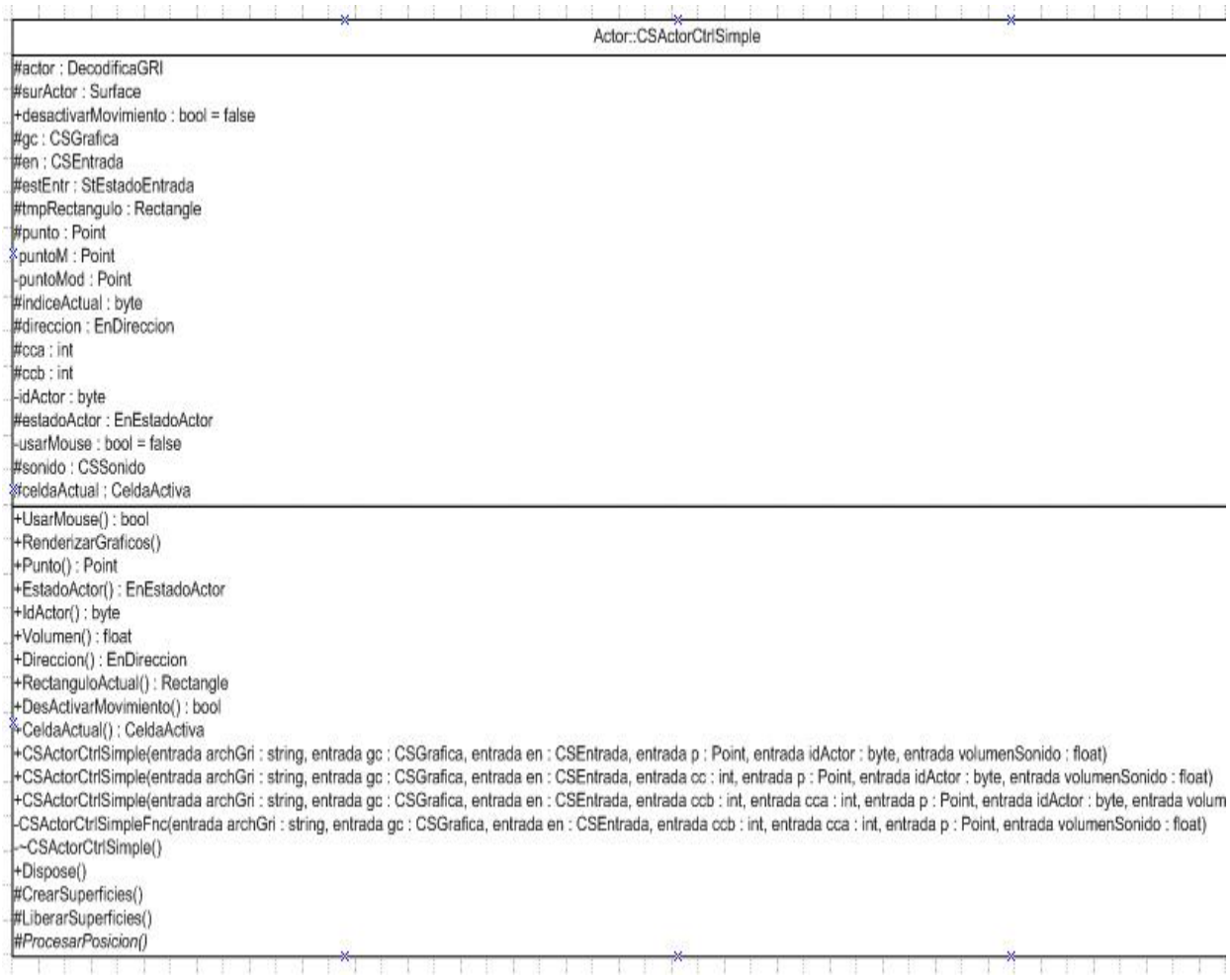
7.10.3. Juego.Tipos.Actor.

Figura 18. Diagrama de Clase CSActorCtrlInv



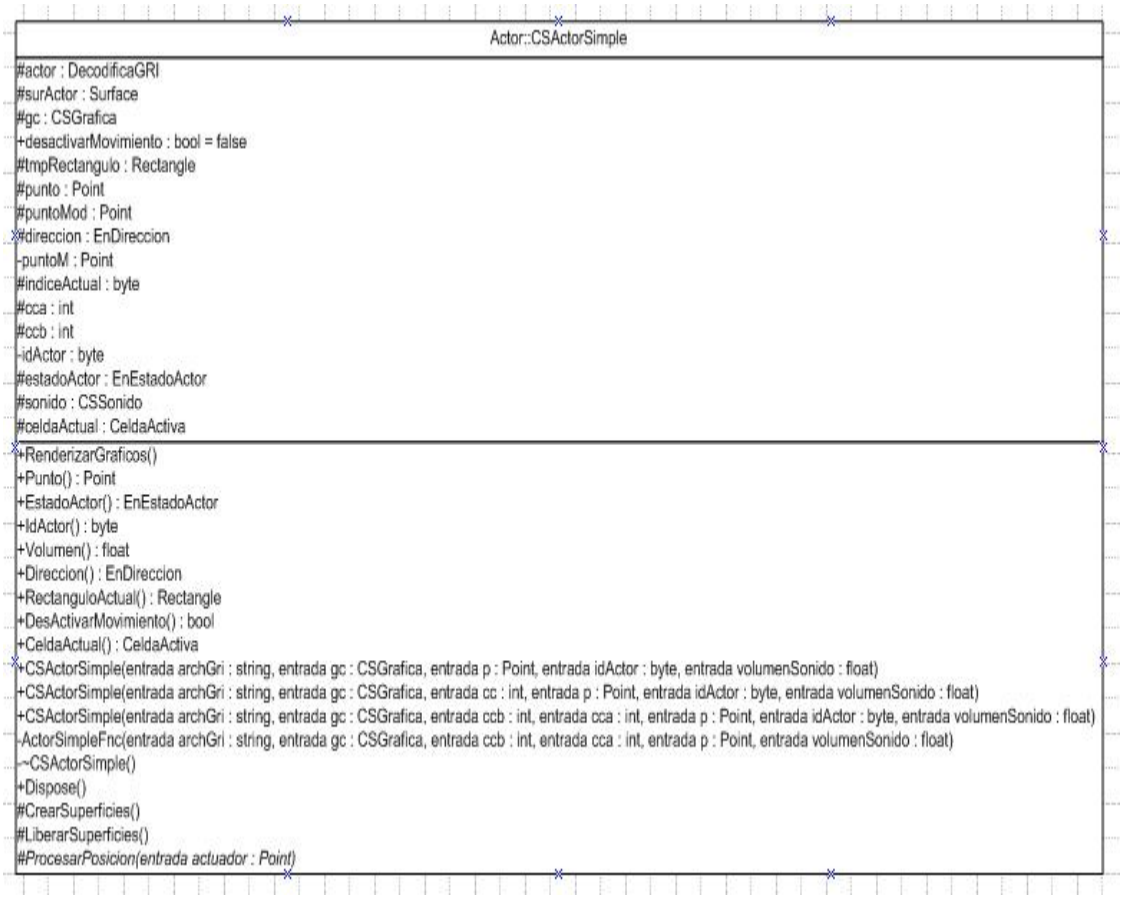
Fuente: Los Autores

Figura 19. Diagrama de Clase CSActorCtrlSimple



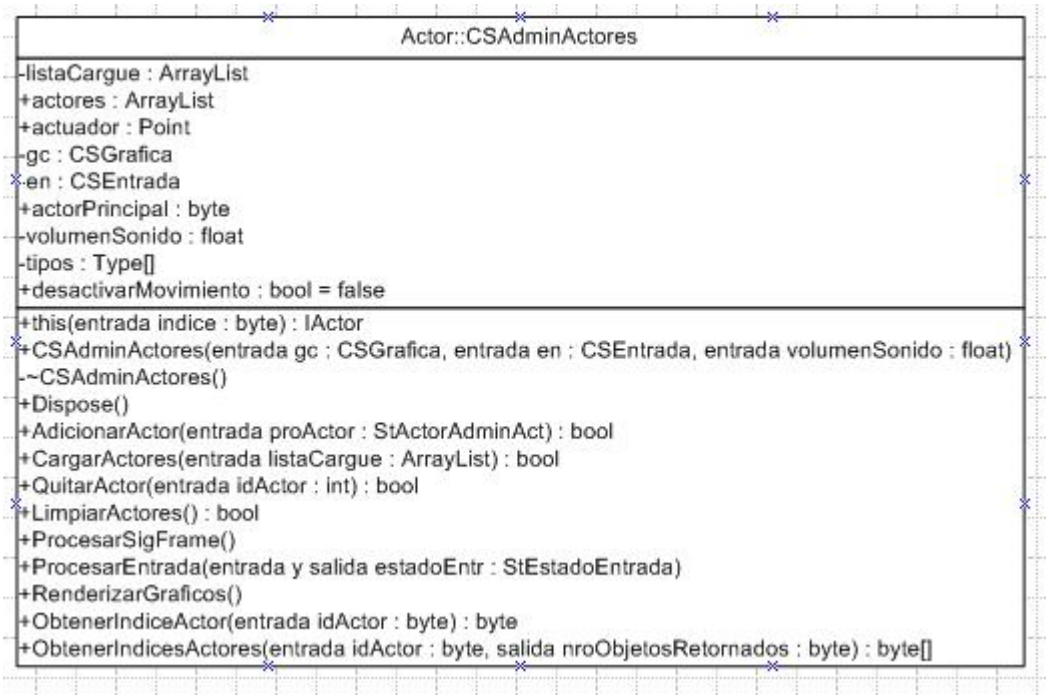
Fuente: Los Autores

Figura 20. Diagrama de Clase CSActorSimple



Fuente: Los Autores

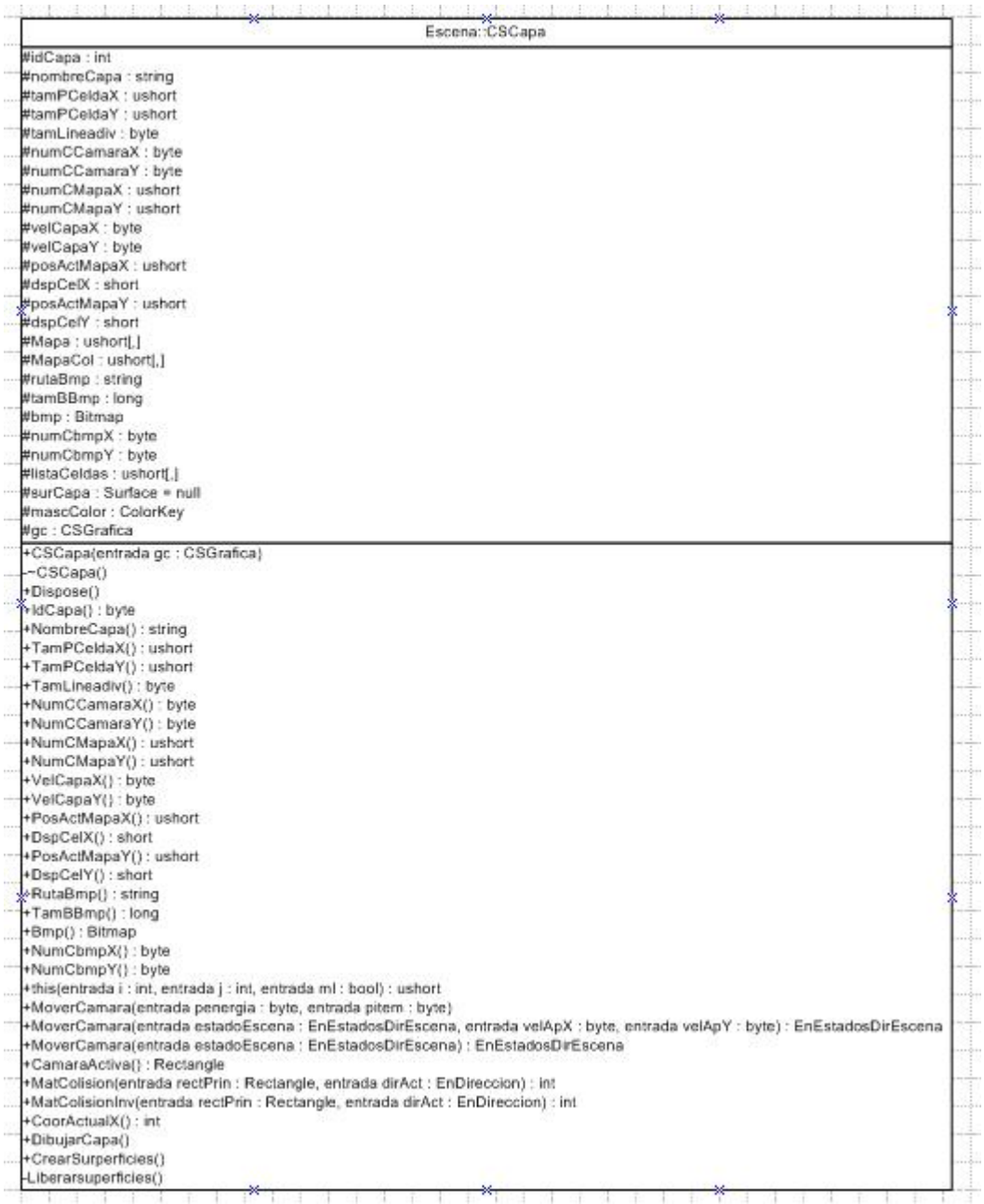
Figura 21. Diagrama de Clase CSAdminActores



Fuente: Los Autores

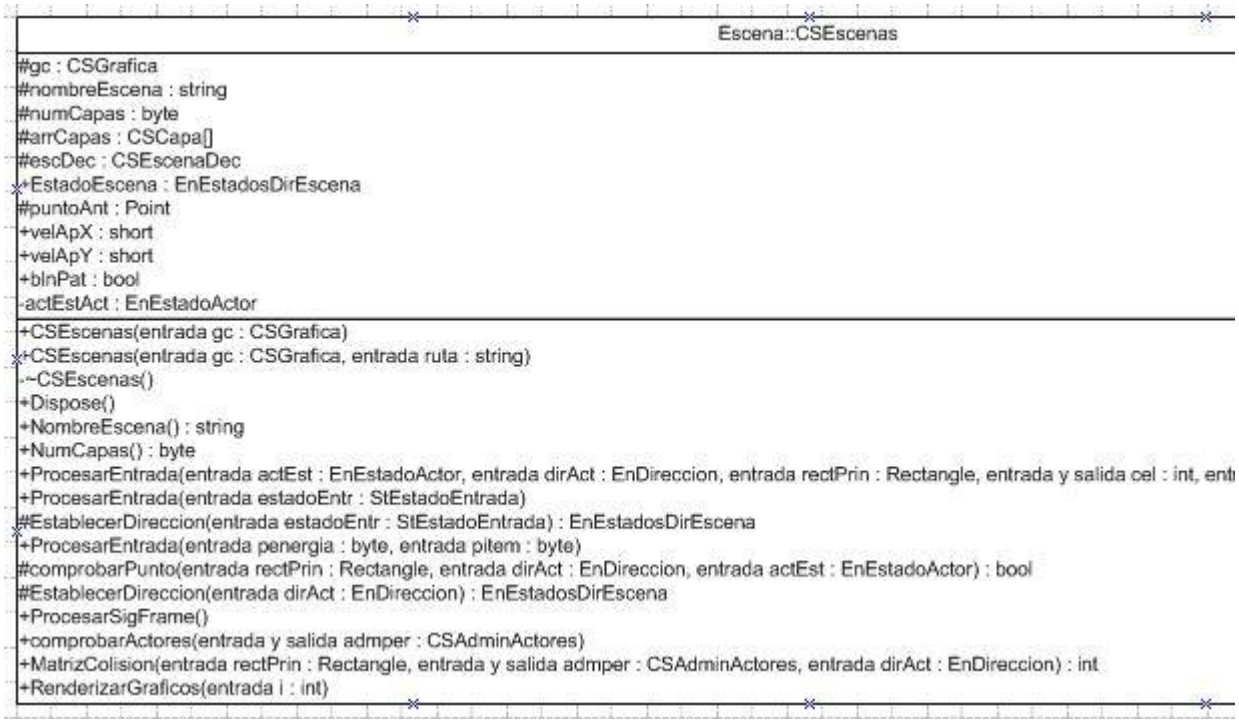
7.10.4. Juego.Tipos.Escena.

Figura 22. Diagrama de Clase CSCapa



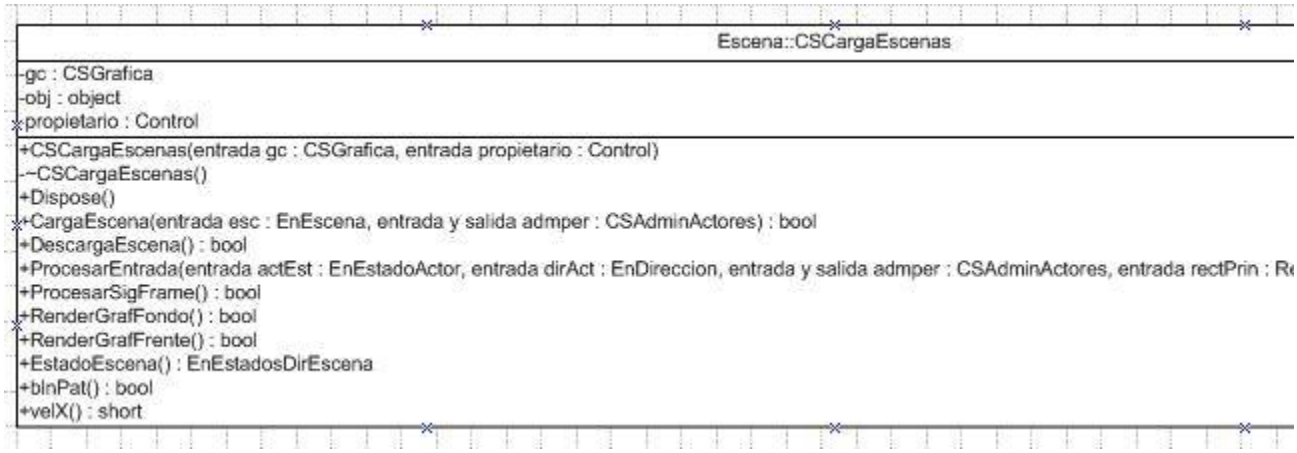
Fuente: Los Autores

Figura 23. Diagrama de Clase CSEscena



Fuente: Los Autores

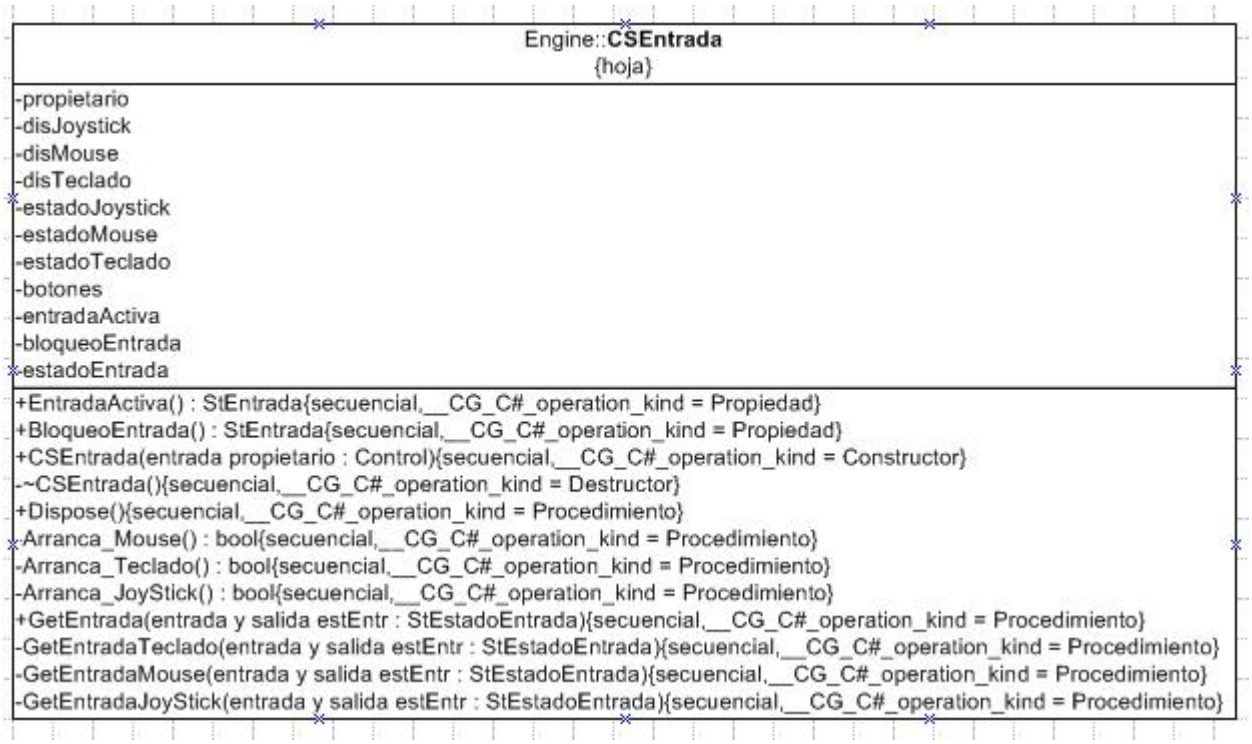
Figura 24. Diagrama de Clase CSCargaEscenas



Fuente: Los Autores

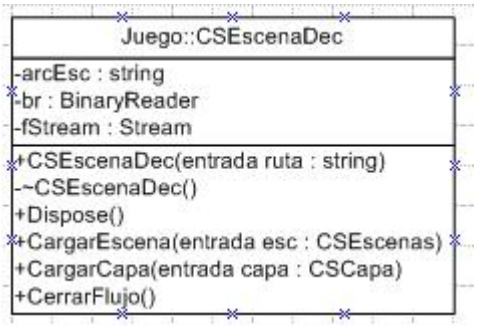
7.10.5. Juego.Engine.

Figura 25. Diagrama de Clase CSentrada



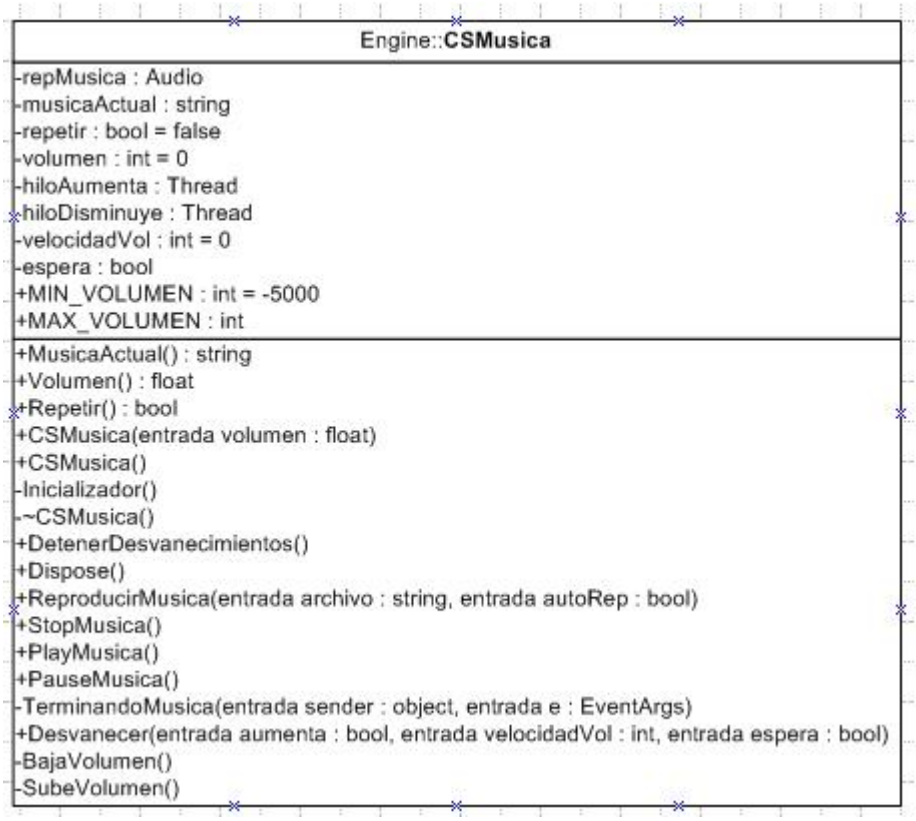
Fuente: Los Autores

Figura 26. Diagrama de Clase CSEscenaDec



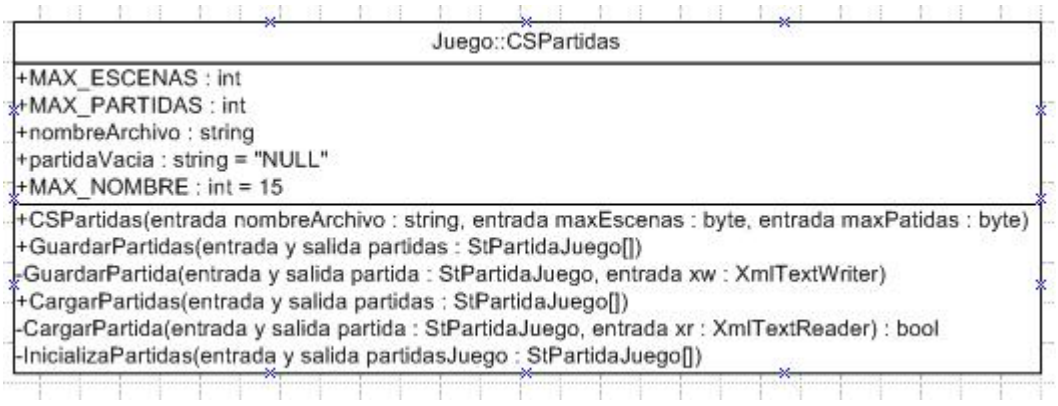
Fuente: Los Autores

Figura 27. Diagrama de Clase CSMusica



Fuente: Los Autores

Figura 28. Diagrama de Clase CSPartidas

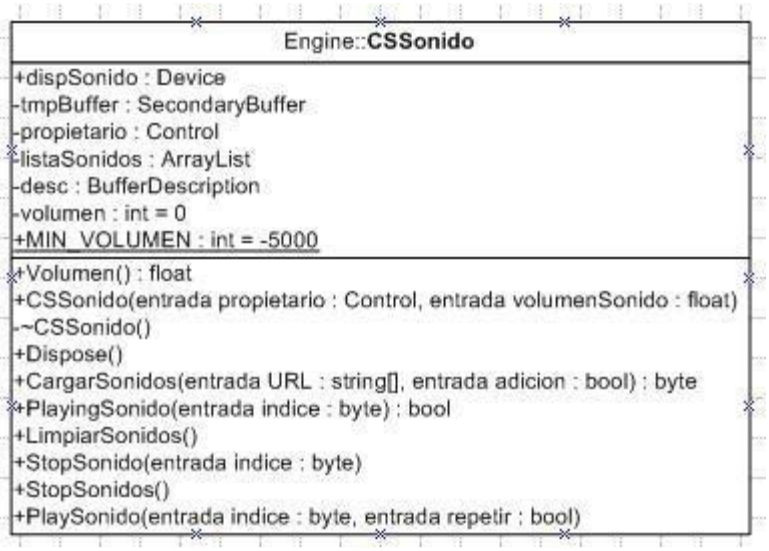


Fuente: Los Autores

Figura 29. Diagrama de Clase CSRegistro

Fuente: Los Autores

Figura 30. Diagrama de Clase CSSonido

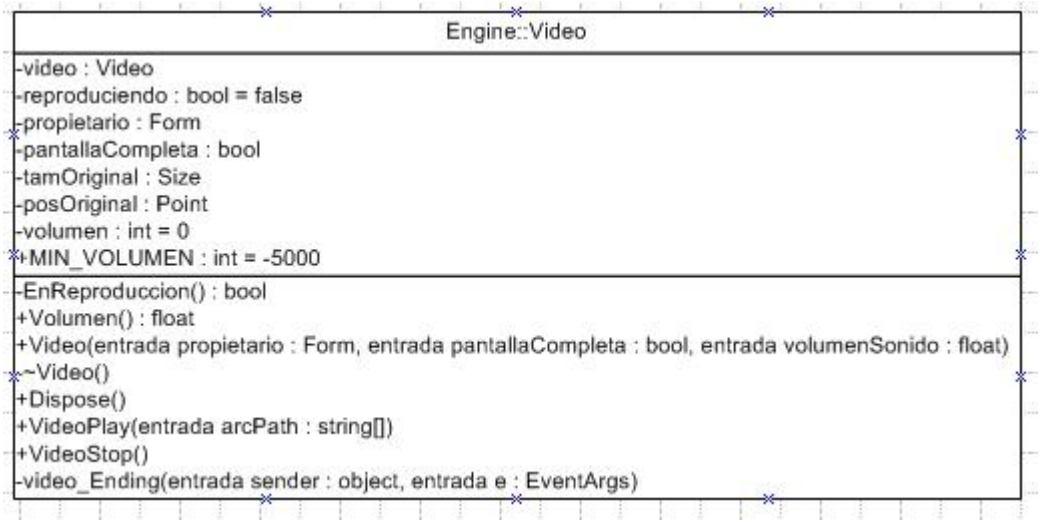


Fuente: Los Autores

Figura 31. Diagrama de Clase decodificaGRI

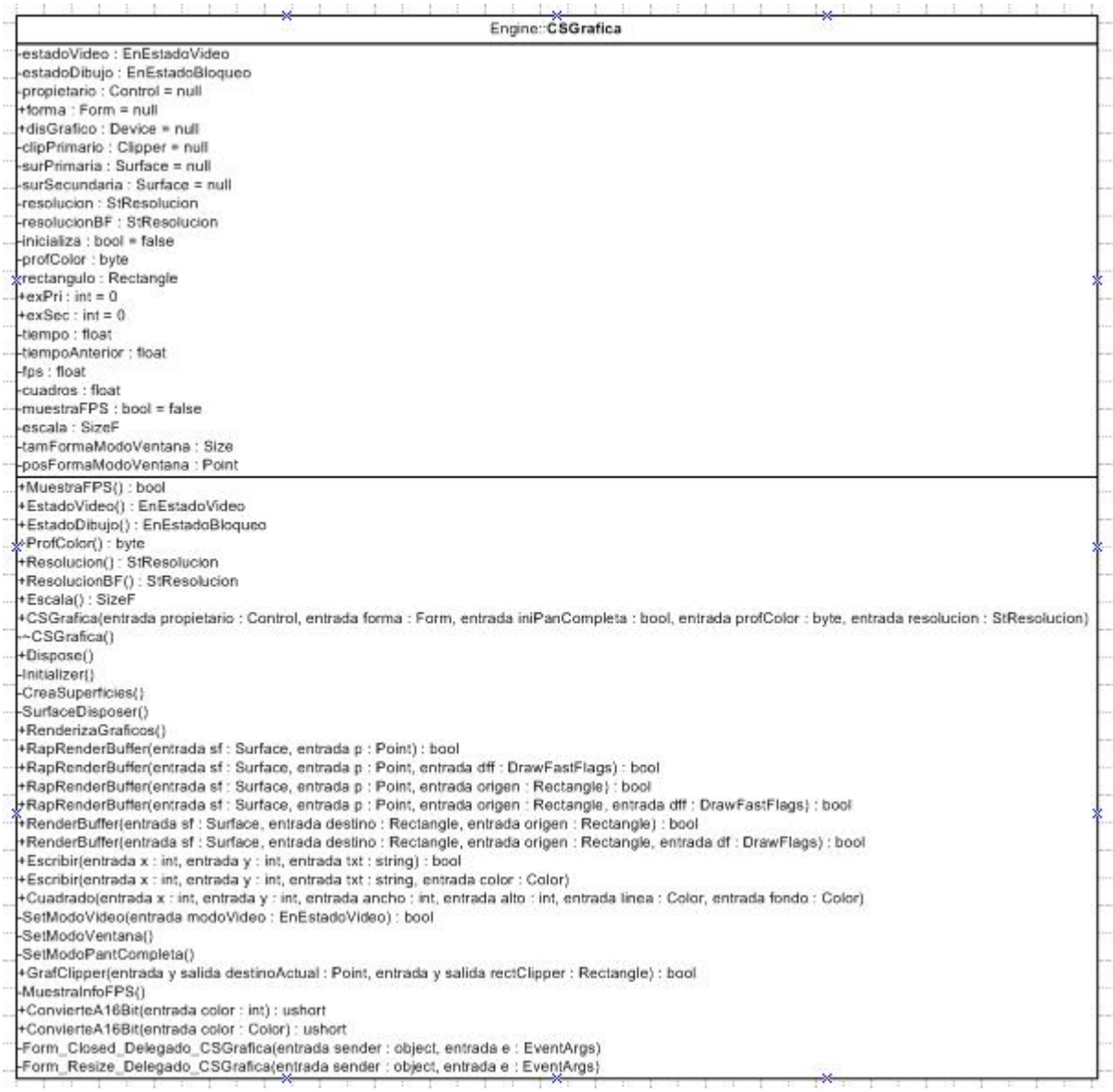
Fuente: Los Autores

Figura 32. Diagrama de Clase video



Fuente: Los Autores

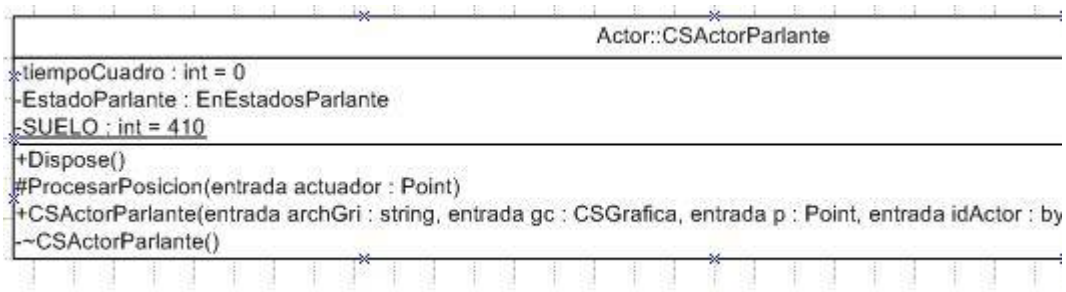
Figura 33. Diagrama de Clase CSGrafica



Fuente: Los Autores

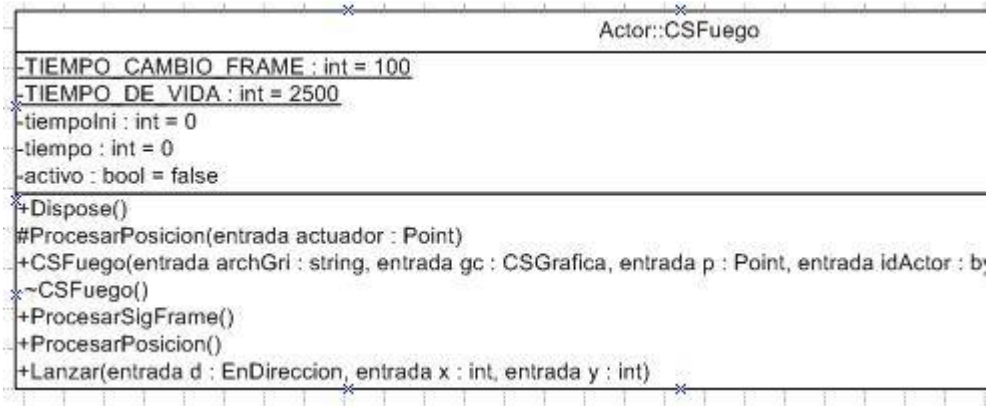
7.10.6. Juego.Juego.Actor.

Figura 34. Diagrama de Clase CSActorParlante



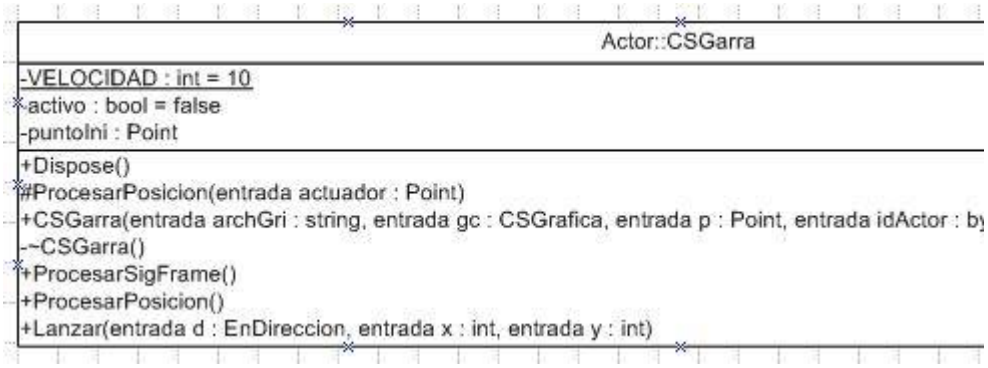
Fuente: Los Autores

Figura 35. Diagrama de Clase CSFuego



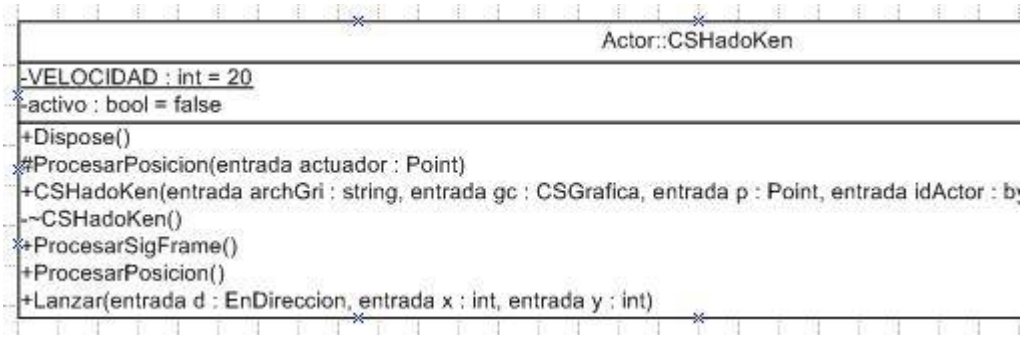
Fuente: Los Autores

Figura 36. Diagrama de Clase CSGarra



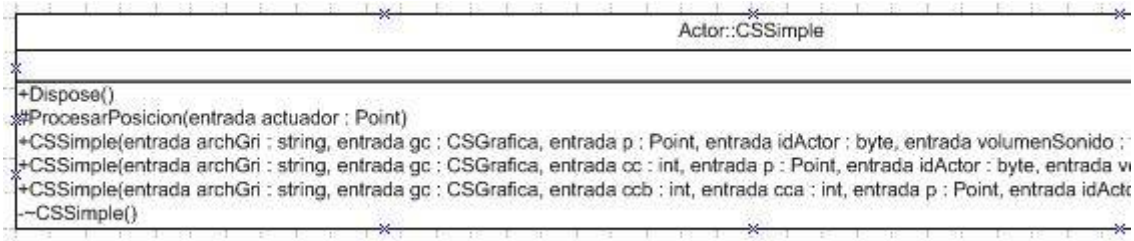
Fuente: Los Autores

Figura 37. Diagrama de Clase CSHadoKen



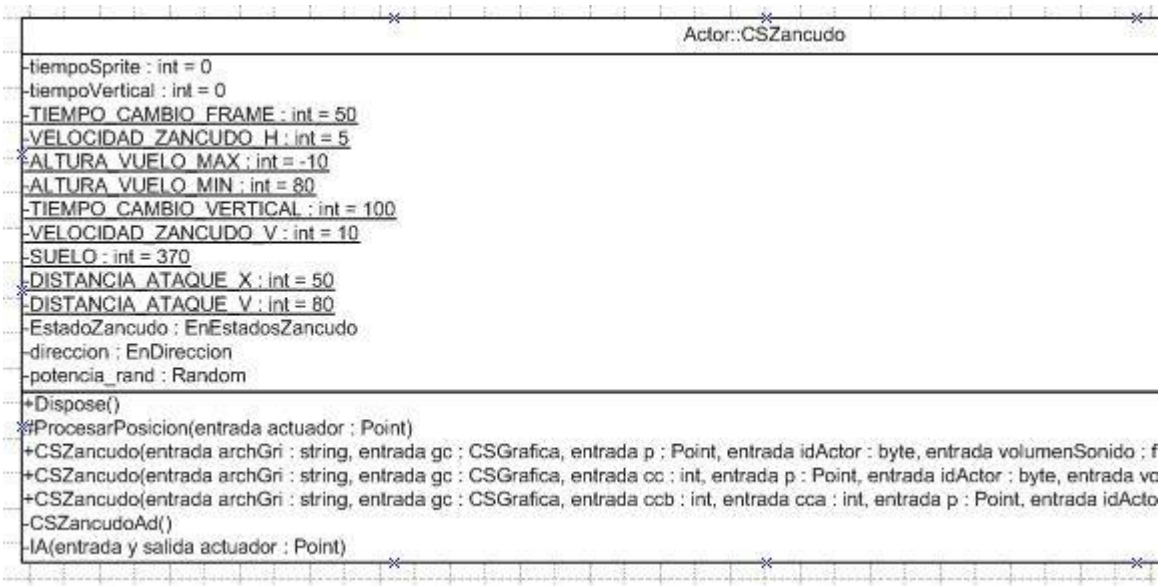
Fuente: Los Autores

Figura 38. Diagrama de Clase CSSimple



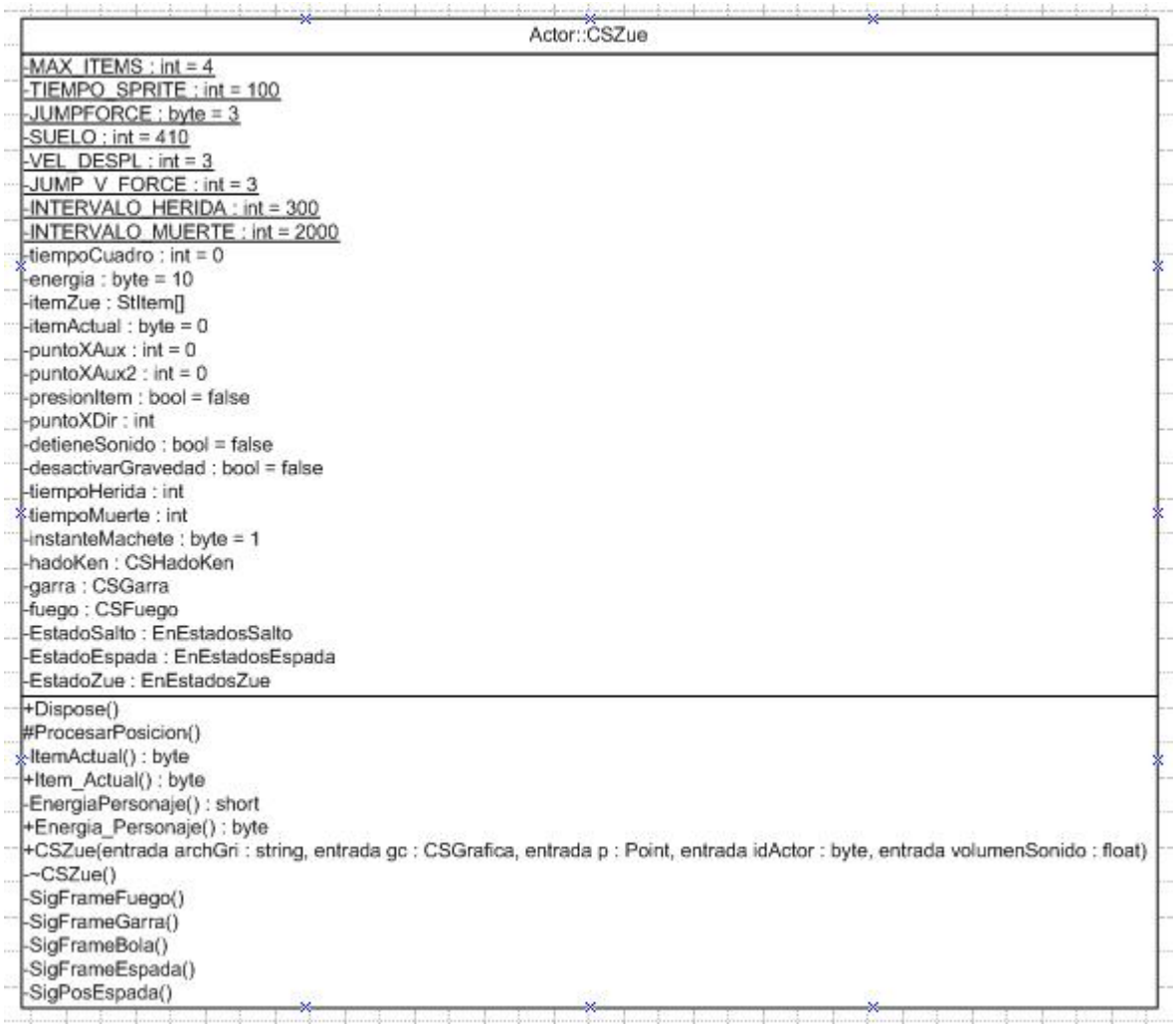
Fuente: Los Autores

Figura 39. Diagrama de Clase CSZancudo



Fuente: Los Autores

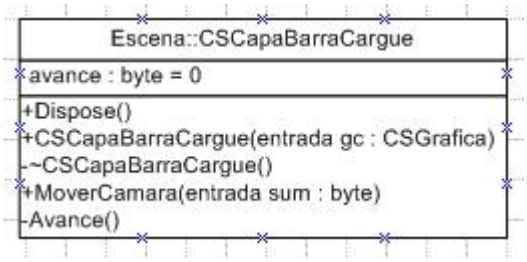
Figura 40. Diagrama de Clase CSZue



Fuente: Los Autores

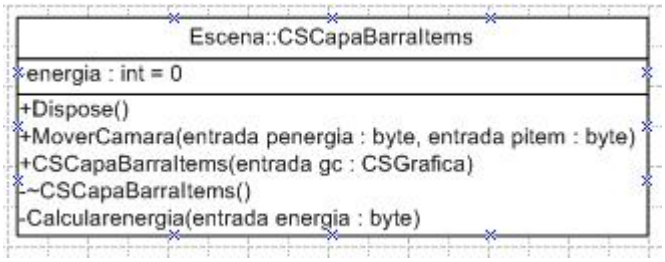
7.10.7. Juego.Juego.Escena.

Figura 41. Diagrama de Clase CSCapaBarraCargue



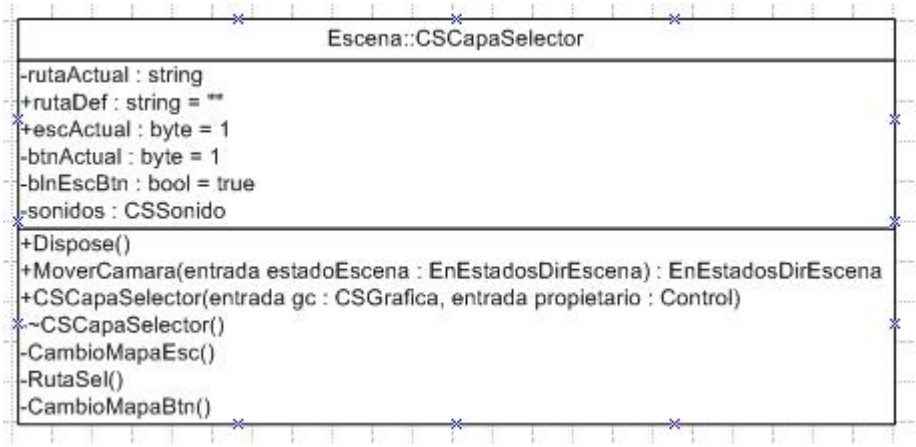
Fuente: Los Autores

Figura 42. Diagrama de Clase CSCapaBarraItems



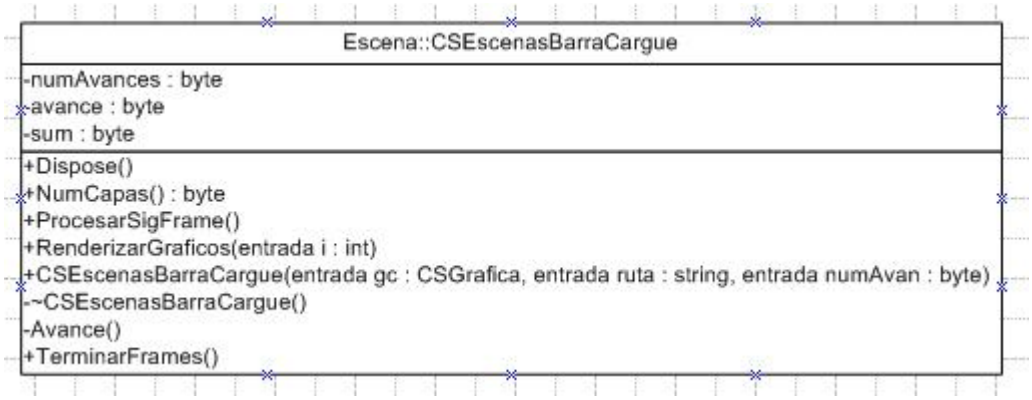
Fuente: Los Autores

Figura 43. Diagrama de Clase CSCapaSelector



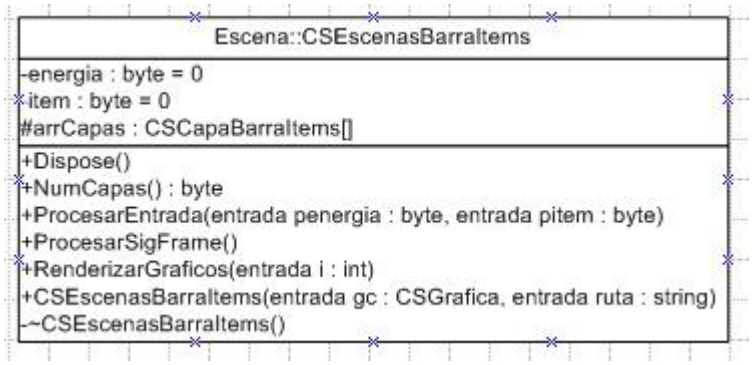
Fuente: Los Autores

Figura 44. Diagrama de Clase CSEscenasBarraCargue



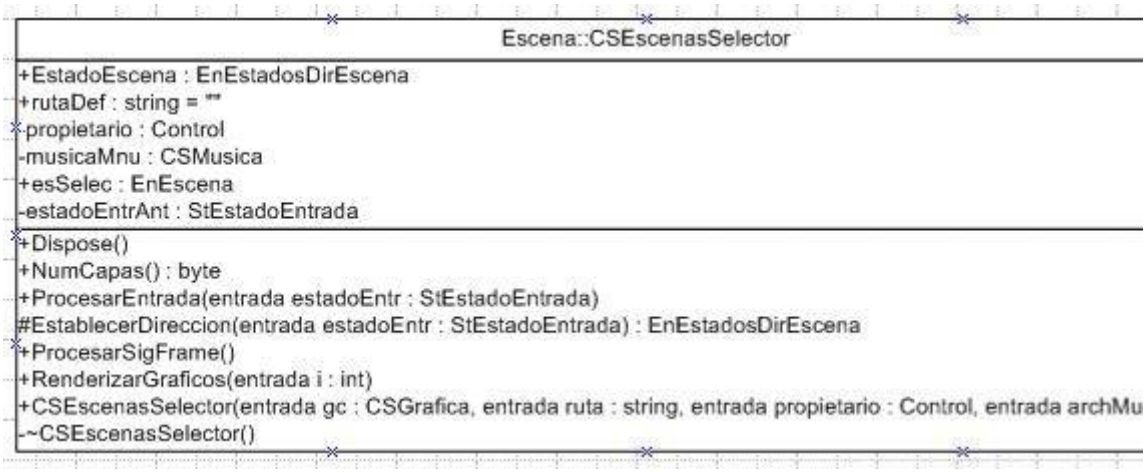
Fuente: Los Autores

Figura 45. Diagrama de Clase CSEscenasBarraItems



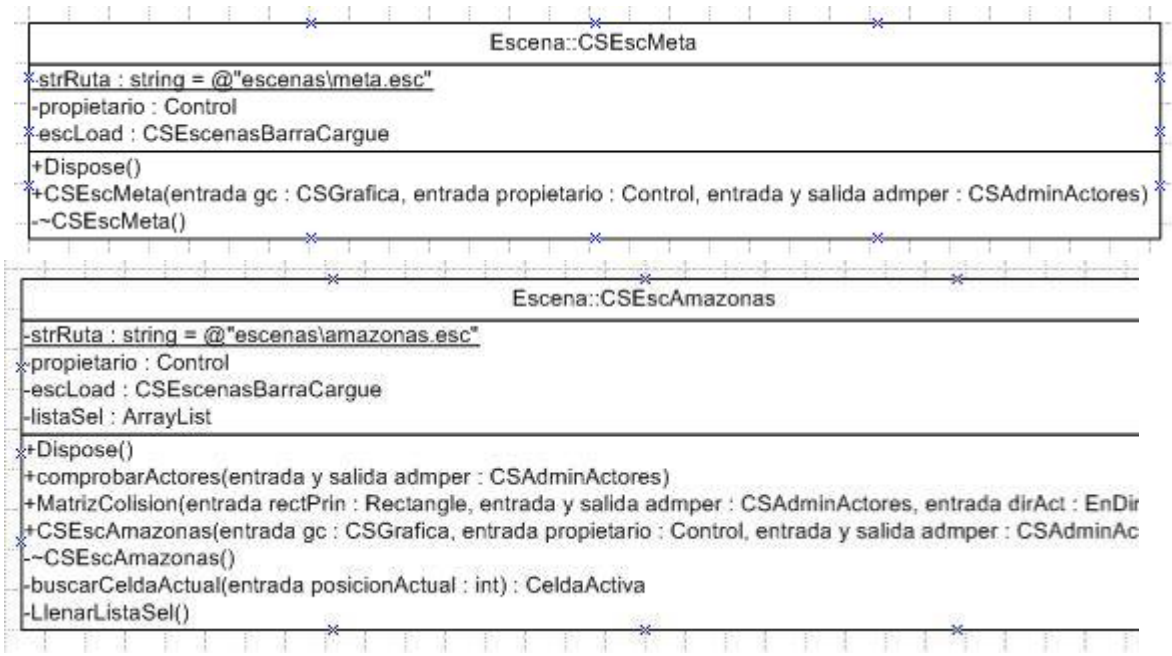
Fuente: Los Autores

Figura 46. Diagrama de Clase CSEscenasSelector



Fuente: Los Autores

Figura 47. Diagrama de Clase CSEscMeta y CSEscAmazonas



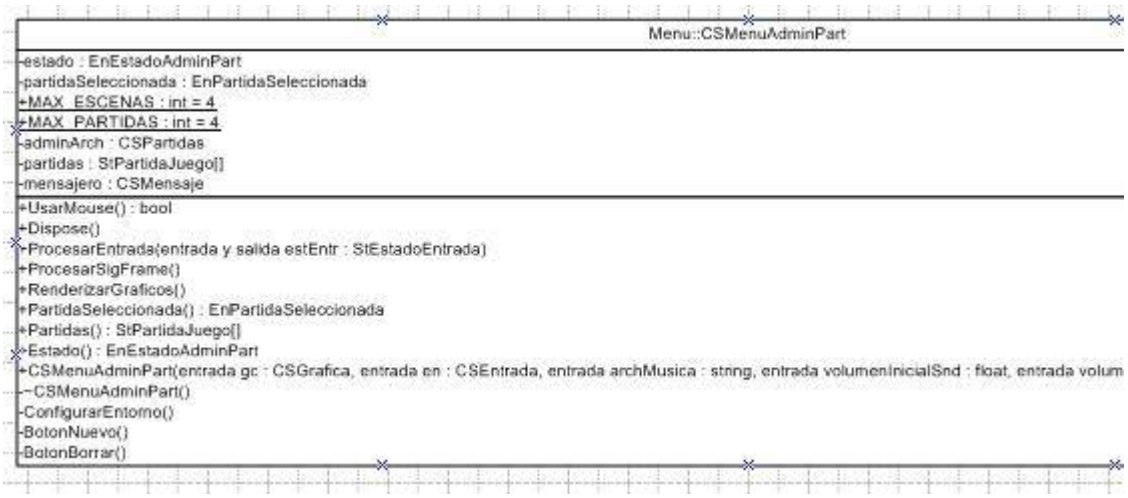
Fuente: Los Autores

7.10.8. Juego.Juego.Menu.

Figura 48. Diagrama de Clase CSMenuPpal

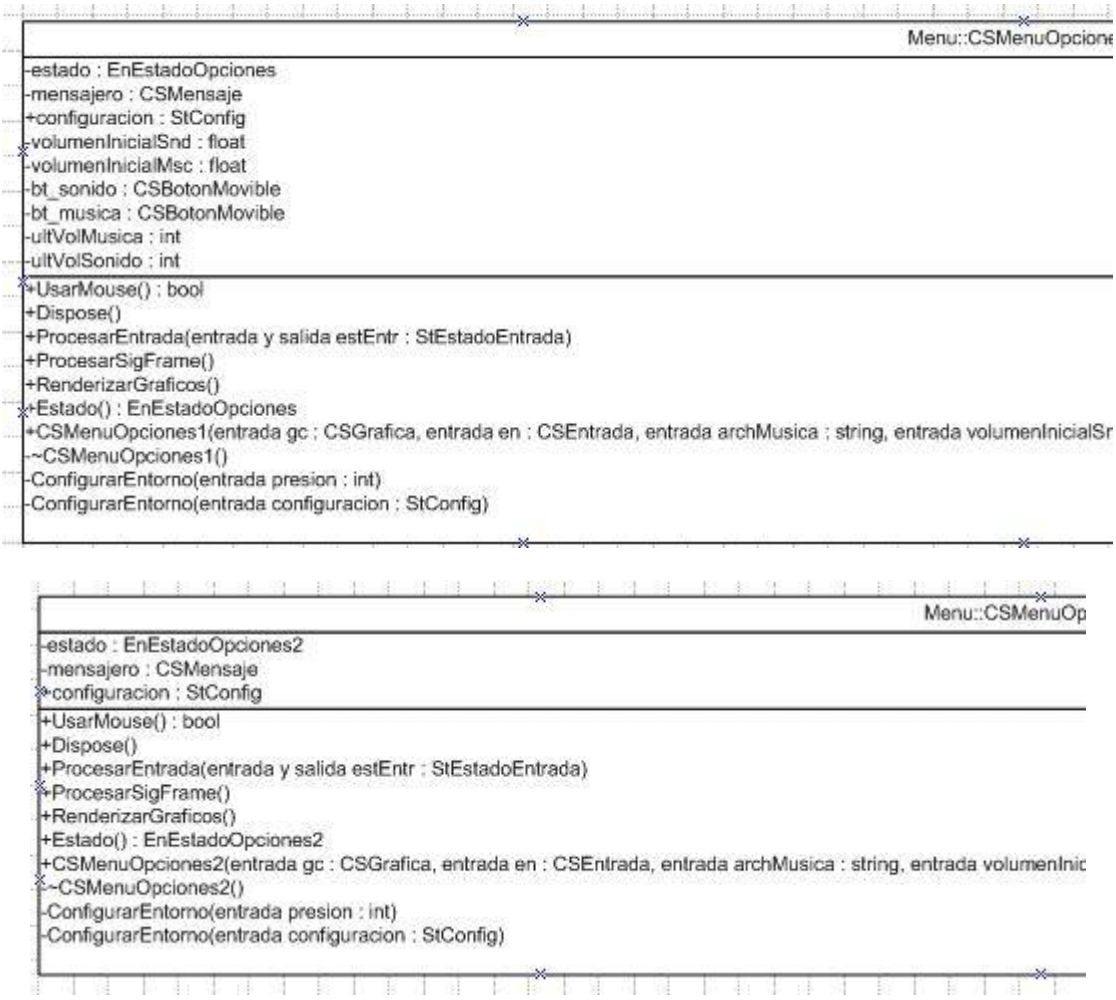
Fuente: Los Autores

Figura 49. Diagrama de Clase CSMenuAdminPart



Fuente: Los Autores

Figura 50. Diagrama de Clase CSMenuOpciones1 y CSMenuOpciones2



Fuente: Los Autores

8. EL MOTOR [ENGINE]

El motor de un videojuego es el núcleo de software que maneja todos los componentes necesarios para la ejecución del juego en un ambiente de hardware específico. En otras palabras es la parte de un videojuego encargada de abstraer la complejidad del hardware y permitir al desarrollador preocuparse solo por los detalles que hacen del videojuego único y retador.

El motor de un videojuego es el corazón del mismo, muchas veces el motor esta tan relacionado con el videojuego que se consideran como uno solo, un buen motor de videojuego es modular, reutilizable, y lo suficientemente flexible para ser usado por múltiples videojuegos similares.

Los requerimientos para un motor de videojuegos dependen en gran parte de qué tipo de videojuegos soportará; por ejemplo las diferencias a bajo nivel pueden ser pocas entre dos motores tan distintos como uno para carreras de autos y otro de juegos de aventuras pero en un nivel mas alto los cálculos necesarios para describir los mundos pueden hacer de estos dos motores algo totalmente diferente.

Los requerimientos básicos para cualquier motor no deberían cambiar. Para todos los videojuegos es necesario los siguientes:

- Manejador Gráfico
- Manejador de Entrada y Salida
- Manejador de Sonido
- Manejador de Música
- Manejador de Video

Cada uno de estos debe permitir al desarrollador de videojuegos utilizar toda la potencia de la computadora o hardware especializado [consolas]. Para lograr un juego que agrade al jugador.

En un nivel más alto un motor debe soportar los requerimientos específicos del tipo de juegos a soportar. Por ejemplo en un motor para videojuegos de aventuras debe existir en el motor la capacidad de manejar los personajes, las escenas y las colisiones entre ellos; además de ambientes como menús o pantalla de opciones.

9. MANEJADOR GRÁFICO

Este módulo es el encargado de realizar toda la labor gráfica involucrada dentro del desarrollo del videojuego y es solo a través de el que cualquier otro módulo puede interactuar con las interfaz de video de la maquina. Dentro de sus requerimientos funcionales se diferencian los siguientes grupos:

- Controlar todos los modos y variantes de video soportados por el motor
- Proveer diversidad de métodos para Dibujar en la pantalla
- Proveer métodos para el trabajo con texto y diálogos (cuadrados)
- Ocultar al usuario la complejidad del hardware
- Brindar una interfaz capaz de soportar las necesidades de los demás módulos principales del motor y de las capas superiores del mismo.

La operación del módulo es de manera asíncrona ya que solo requiere procesamiento una vez por ciclo de juego.

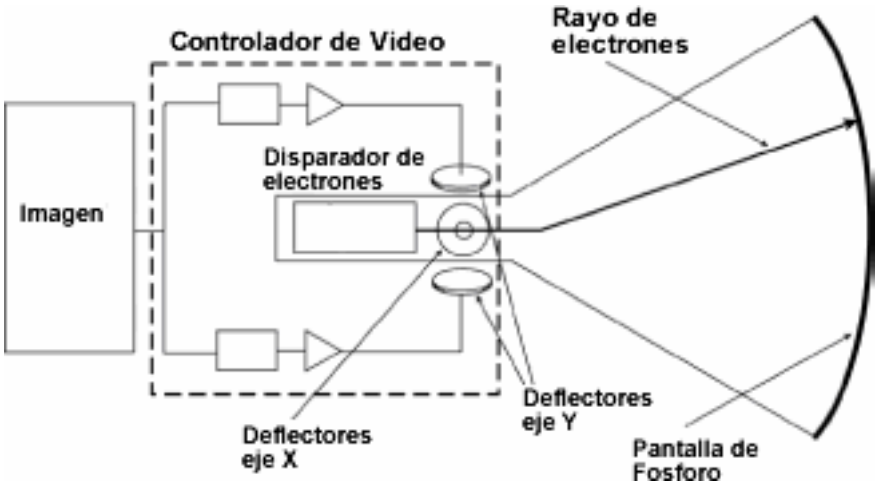
9.1. REQUERIMIENTOS RELATIVOS AL CONTROL DE LOS MODOS, VARIANTES Y PROBLEMAS DE VIDEO SOPORTADOS POR EL MOTOR.

Para lograr esto se tienen en cuenta diferentes aspectos como los son:

- El barrido vertical de la pantalla
- El modo de pantalla completa y el modo de ventana
- El volcado de información a la memoria de video
- Los modos de resolución
- Los modos de color
- Recuperación ante los cambios de contexto (Cambios de modo)

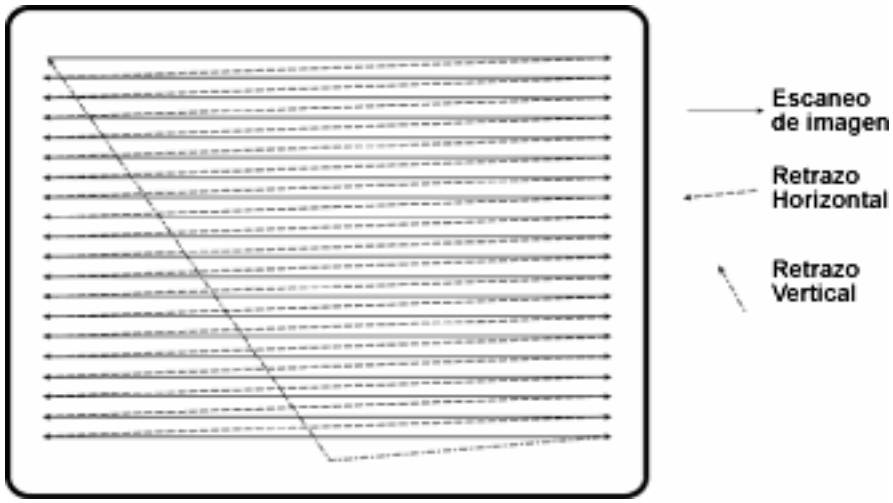
9.1.1. El barrido Vertical de la pantalla (Vertical Retrace). Al momento de dibujar en el dispositivo de video hay que tener en cuenta el funcionamiento del mismo (Figura 51), dado que la forma en que se proyecta una imagen en pantalla influye bastante al momento de mostrar una imagen y una animación de calidad. Los monitores generalmente poseen un ciclo de refresco de la información que se muestra en ellos, dicho refresco ocurre generalmente cada 60 veces por segundo es decir cada segundo la imagen del monitor es dibujada 60 veces por el dispositivo, este detalle técnico tiene gran incidencia al dibujar, ya que si se dibuja en el momento que esta sucediendo el barrido sobre la pantalla ocurre un fenómeno llamado en ingles flicker lo cual es un salto o parpadeo dentro del dibujo o la animación en pantalla .(figuras 52 y 53). El motor debe proveer los medios para evitar las diferentes clases de parpadeo de imagen.

Figura 51. Funcionamiento de un monitor CRT



Fuente: Los Autores

Figura 52. El barrido de pantalla



Fuente: Los Autores

Figura 53. El efecto de flicker*

Fuente: Los Autores

Para controlar esta situación se hace necesario implementar o utilizar rutinas capaces de diferenciar en que momentos esta ocurriendo el barrido vertical de la pantalla y en qué momentos este ha finalizado o comenzado a efectuarse, de tal modo que se evita el efecto de flicker.

9.1.2. El volcado de la información a la memoria de video. Para dibujar en la memoria de video no solo se debe tener en cuenta el retrazado vertical de la pantalla, ya que los tiempos involucrados en realizar un dibujo de una u otra manera, tienen un impacto considerable en el rendimiento del dispositivo y esto desencadena nuevamente efectos desagradables como el del flicker. Puede suceder que al dibujar una imagen píxel por píxel esto implique que por cada píxel que se dibuje (dibujar implica directamente escribir en la memoria de video) se haga la espera del retazado vertical para evitar el parpadeo, una imagen esta constituida por miles o millones de píxeles así que dibujar una imagen muy pequeña de 50 x 50 píxeles implicaría esperar 250 barridos verticales de la pantalla para dibujarla es decir alrededor de tres segundos, igualmente sucede en el caso de que se requiera dibujar varias imágenes a la vez ya que implica esperar varios retrasos verticales antes de dibujar cada una y cada una por si misma produciría muchos efectos de parpadeo al dibujarse.

Para dar solución a este inconveniente existen muchas técnicas bien conocidas, entre las cuales se destaca la técnica de ‘dirty rectangles’ y la técnica de ‘doble buffer’, cada cual tiene sus ventajas y desventajas según el caso en que se utilice cada una pero para efectos de este proyecto solo se ha usado la técnica de doble buffer ya que esta se acomoda al gran volumen de datos volcados a la vez a la memoria de video. Esta técnica simplemente consiste en dibujar en una porción diferente de la memoria de video visible,

* Se ha exagerado el efecto para hacerlo notar en la imagen

es decir en la misma memoria de video o bien en la memoria principal de sistema, si se hace de esta manera no es importante verificar el retrazo vertical de pantalla por cada píxel que se dibuje, debido a que lo que se dibuja no es visible en pantalla; una vez se ha dibujado todo en esta porción de memoria es tiempo para verificar el retrazado vertical, y una vez este halla sucedido se copia en una sola pasada toda la información a la memoria de video visible (Figura 54).

Dicho de otra manera tenemos los siguientes elementos involucrados:

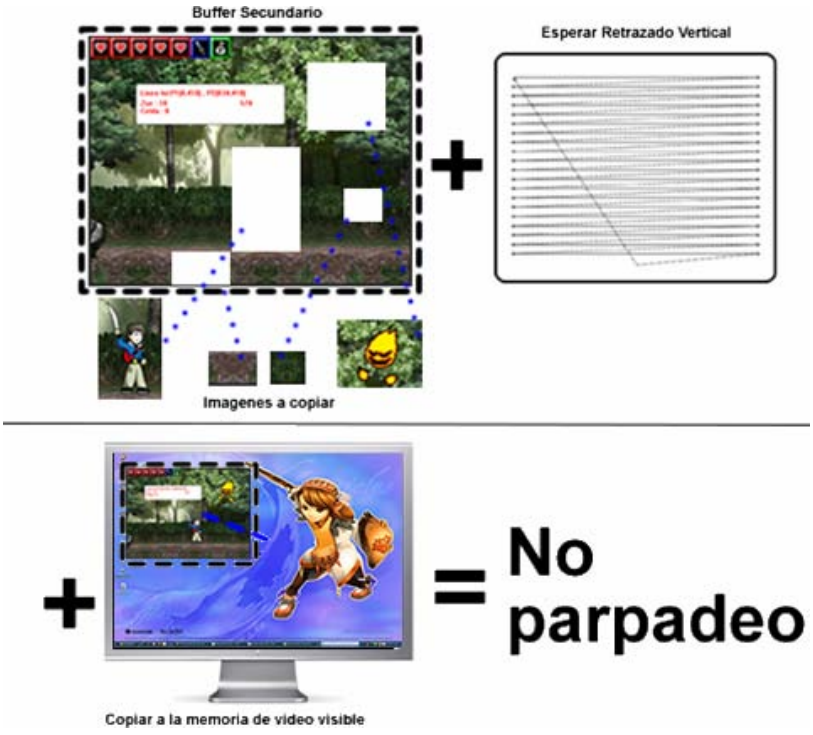
- Buffer principal (memoria de video visible)
- Buffer secundario (porción de memoria diferente de la memoria de video visible)
- Retrazado vertical

Para crear cualquier dibujo se siguen los siguientes pasos:

- Copiar o dibujar todo en el buffer secundario
- Esperar el barrido de pantalla
- Copiar el buffer secundario al buffer principal.

El motor debe soportar operaciones de copiado entre secciones de memoria, de tal manera que se soporte la técnica de doble buffer.

Figura 54. Técnica de doble buffer



Fuente: Los Autores

9.1.3. Modo pantalla completa y modo ventana. El videojuego generalmente tiene la posibilidad de jugarse en dos modos diferentes, uno de ellos es el modo ventana donde el videojuego se ejecuta en una ventana tradicional del sistema operativo, lo cual le permite al usuario alternar fácilmente con otras aplicaciones mientras esta jugando (Figura 55), el otro modo es el modo de pantalla completa (Figura 56) el cual utiliza todo el dispositivo de video para su ejecución y no permite al usuario alternar fácilmente con el uso de otras aplicaciones; Cada uno de los modos posee ventajas y desventajas, sin embargo para los jugadores es mucho mas habitual y cómodo el modo de pantalla completa ya que este permite un desempeño completo del software en la máquina y adicionalmente se alcanza un mayor nivel de detalle en las gráficas del juego. El motor debe tener soporte para alternar entre modo ventana y modo pantalla completa, y como es habitual en los juegos actuales, debe soportar el cambio dinámico entre estos modos en tiempo de juego.

Figura 55. Modo ventana

Fuente: Los Autores

Figura 56. Modo pantalla completa

Fuente: Los Autores

9.1.4. Modos de resolución de video. Los dispositivos de video soportan múltiples resoluciones que varían de acuerdo a las características del monitor y de la tarjeta de video, dentro de las resoluciones más utilizadas se encuentran:

- 640*480
- 800*600
- 1024*768

Siendo la resolución mas común entre los usuarios finales la de 800*600 px., sin embargo muchos de los profesionales de la computación prefieren resoluciones mayores para facilitar su trabajo generalmente resoluciones superiores a 1024*768 px.

Los videojuegos trabajan generalmente con resoluciones que van desde 320*240 px. en adelante, pero debido a los cambios en las tecnologías actuales, cada vez es menos común ver videojuegos que se ejecuten a menos de 640*480 px. debido a la consecuente disminución de los detalles en la imagen, de hecho los videojuegos más modernos incluyen soporte para resoluciones altas de hasta 1280*1024 px. o más, pero son pocos las personas que disponen de recursos suficientes para adquirir tarjetas de video de gama alta que pueden soportar estas resoluciones. El motor debe incorporar soporte para resoluciones:

- 640*480
- 800*600
- 1024*768

9.1.4.1. Escalamiento de superficies. Dentro del videojuego sin embargo no solo es necesario conocer la resolución de pantalla[*], la complejidad de los volcados de memoria y de las operaciones de superposición entre mapas de bits implican tener presentes varias resoluciones diferentes para poder trabajar de manera mas precisa y versátil.

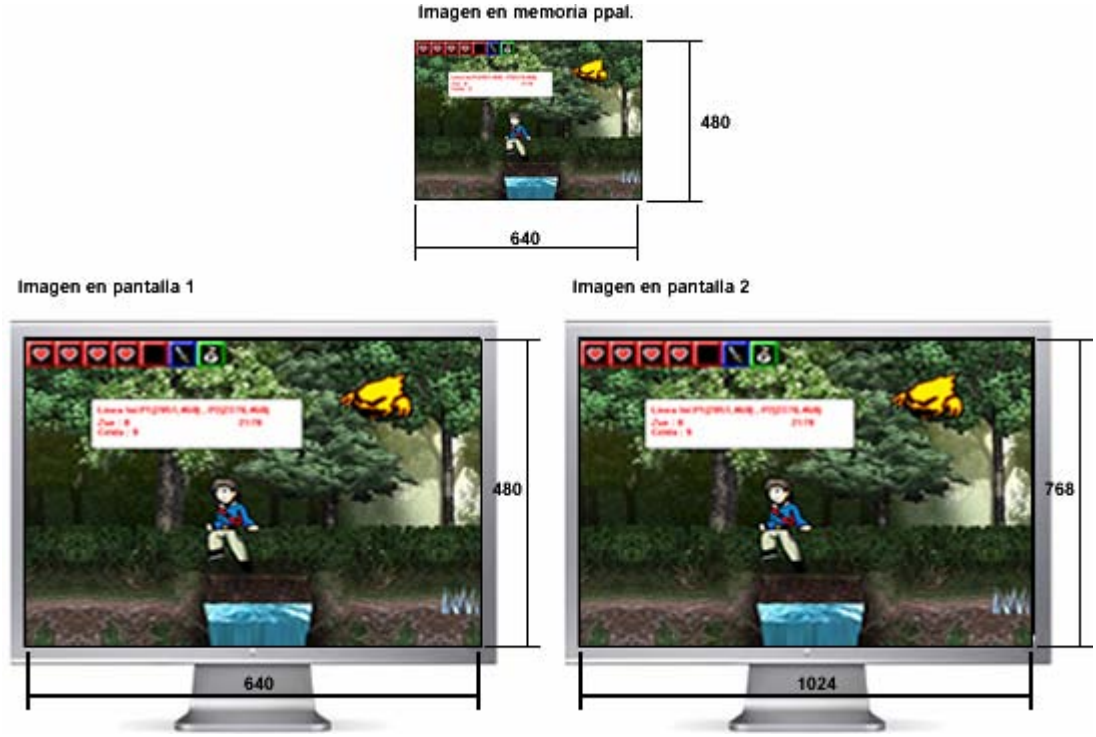
Cada imagen en memoria posee sus propias dimensiones, al mostrar esta imagen en pantalla hay que tener en cuenta la resolución de la pantalla para poder mostrar la imagen pues aunque la imagen tenga una resolución de 1024*768 px. si la pantalla no tiene esa resolución sino una mayor o una inferior entonces será necesario redimensionar la imagen al momento de copiarla a la memoria de video, este redimensionamiento implica pérdida de la información de la imagen así que se hace necesario crear una copia redimensionada de la imagen y esta copia será la que se pegará en la memoria de video.

Para realizar este proceso dentro de un videojuego es necesario tener en cuenta otros aspectos, ya que estas operaciones se harían de manera drásticamente diferente dependiendo si el videojuego se esta ejecutando en modo ventana o en modo de pantalla completa, en modo pantalla completa la imagen en memoria tiene sus propias dimensiones, la pantalla posee también su resolución propia, así que aunque la imagen original sea de 640*480 en memoria principal, al pegarla en la memoria de video esta

[*] En adelante también se hará referencia a la porción de memoria de video visible como pantalla, para facilitar la comprensión del lector.

debe ser del tamaño de la memoria de video visible, ya que independientemente de la resolución seleccionada por el jugador el juego debe ocupar la totalidad e la pantalla en todo momento (Figura 57).

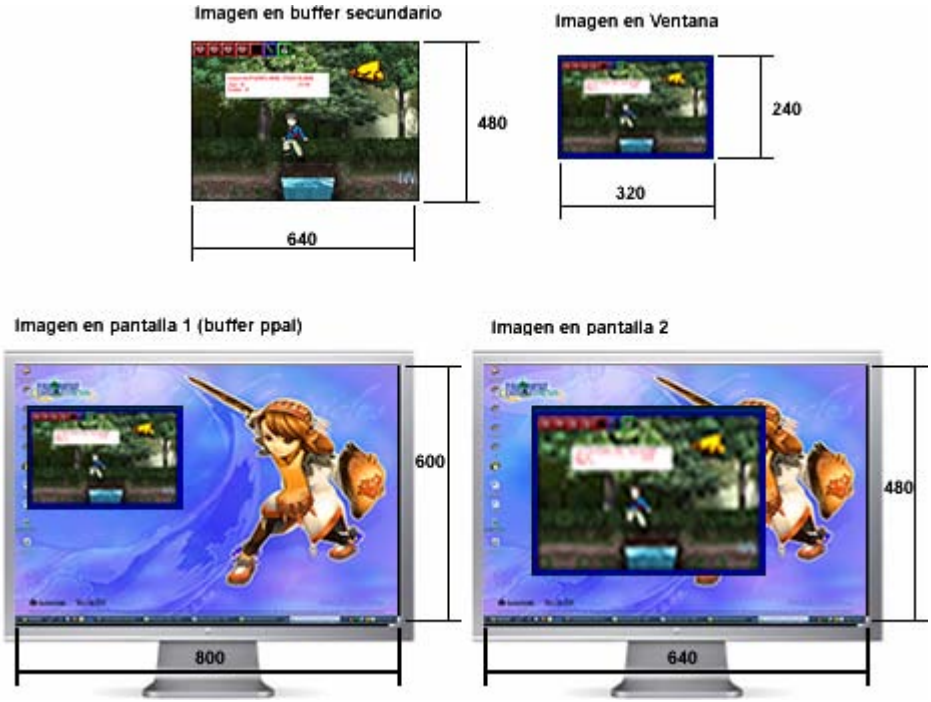
Figura 57. Copiado en modo de pantalla completa



Fuente: Los Autores

En el caso del copiado en modo ventana esta labor se hace un poco mas dispendiosa, debido a que adicionalmente a los aspectos tratados en modo pantalla completa se deben tener en cuenta otro tipo de variables, en cuanto a la resolución se refiere se debe tener en cuenta el tamaño de la ventana donde se dibujará la imagen ya que aunque la resolución de la pantalla sea una, la ventana generalmente tiene dimensiones inferiores a esta, es decir una pantalla que se encuentre a 1024*768 px. puede tener la ventana de juego a tan solo 320*240 px. de hecho esta misma ventana podría estar en una pantalla configurada a 800*600 px. y su tamaño sería diferente (Figura 58), por estos motivos en modo ventana se debe realizar escalamientos en tres niveles diferentes de conversión, a diferencia del modo ventana completa en el cual solo se realiza un escalamiento. El motor debe incorporar interfaces y funcionalidades que permitan realizar escalamiento de imágenes o bien realizarlas de manera automática.

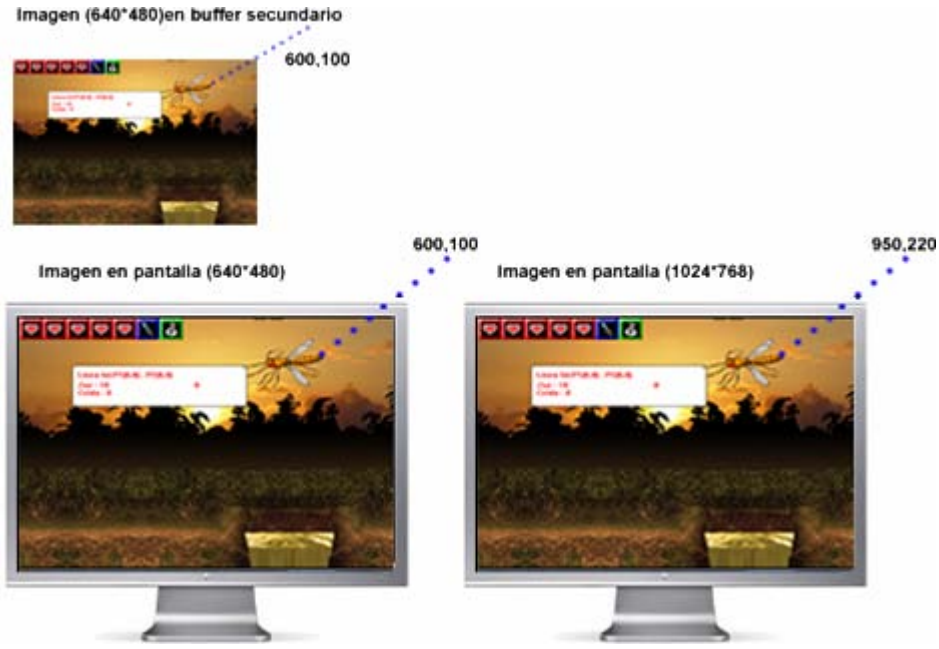
Figura 58. Copiado en modo ventana



Fuente: Los Autores

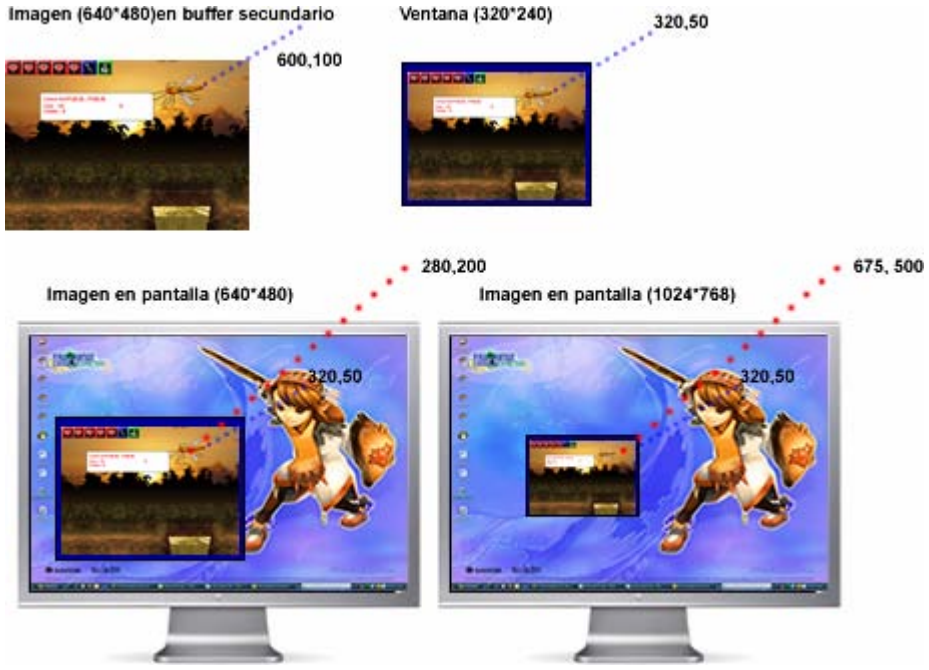
9.1.4.2. Sistema de coordenadas. Al igual que en la escala de la imagen, también se deben realizar cálculos y conversiones para el sistema de coordenadas dependiendo del modo de resolución en el que se ejecute el juego. En modos de pantalla completa se debe realizar la equivalencia de manera directa es decir se calcula a que coordenada corresponde un punto de la imagen al convertirlo en coordenadas de memoria de video visible (Figura 59), mientras para modo ventana el cálculo nuevamente depende de más variables, la coordenada real (imagen), la coordenada en resolución de pantalla (conversión 1), tamaño de la ventana (conversión 2) y la ubicación de la ventana en la pantalla (desplazamiento, offset) (Figura 60). El motor debe incorporar interfaces y funcionalidades que permitan realizar conversiones entre coordenadas o bien realizarlas de manera automática.

Figura 59. Conversión de coordenadas en modo pantalla completa



Fuente: Los Autores

Figura 60. Conversión de coordenadas en modo ventana



Fuente: Los Autores

9.1.5. Modos de color. Para la máquina el color puede representarse de diferentes maneras, para representar un color se requiere dividir este en cada uno de sus componentes, los cuales pueden variar de acuerdo al color que se use.

- Escala de grises con 8 o 16 bits
- 256 Colores 1 byte para formar el color, 3 bits para rojo, 3 bits para verde, 2 bits para azul
- Color de 16 bits (2 bytes – color de alta densidad), 5 bits para rojo, 6 bits para verde, 5 bits para azul, con lo cual se pueden obtener 65.536 combinaciones de color diferentes.
- Color de 24 bits (3 bytes – color verdadero), 8 bits para cada color, con lo cual se pueden obtener 16’777.216 combinaciones diferentes.
- Color de 32 bits (4 bytes – color real), 8 bits para cada color y 8 bits para la transparencia, con lo cual se pueden obtener 16’777.216 combinaciones diferentes, mas 255 niveles de transparencia adicionales por cada color, lo cual da un total de 4.294’967.296 combinaciones diferentes.

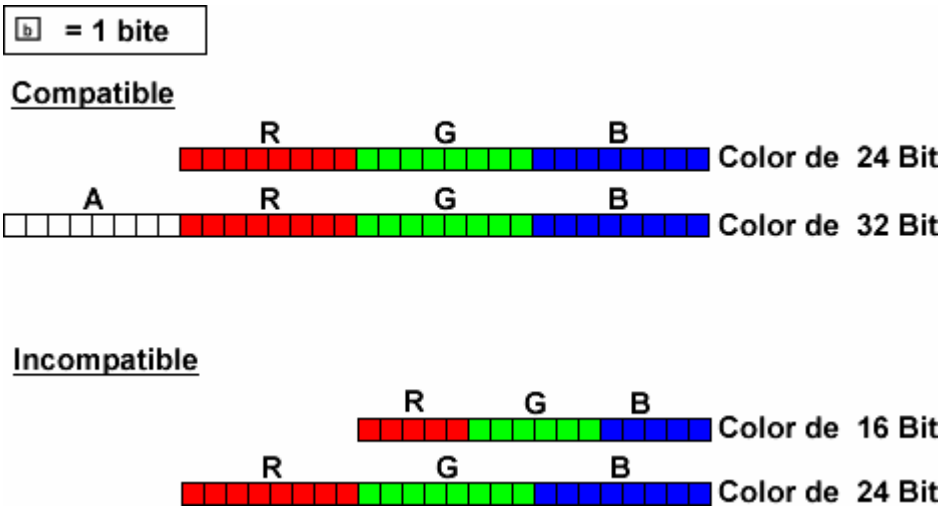
El modo de 32 bits produce casi idénticos colores que el modo de 24 bits, su implementación nació debido a que los PC actuales los cuales mueven sus datos en paquetes de 32 bits, por ello resulta más rápido mover un color de 32 bits en un solo ciclo de reloj que mover uno de 24 bits en 2 o 3 ciclos de reloj (16,4 o 8, 8,8).

Los modos de color inferiores al color de 16 bits, no poseen la cantidad de colores suficiente para representar una imagen con calidad superior, razón por la cual se hace necesario trabajar diferentes tipos de paleta de color para cada dibujo, esto demanda mas trabajo no solo al momento de realizar la programación sino principalmente requiere gran esfuerzo por parte del artista o dibujante ya que para obtener buenos resultados con tan pocos colores disponibles, se requiere de gran cantidad de tiempo lo cual con las tecnologías actuales podría representar un costo innecesario para muchas empresas.

El motor debe soportar modos de color de 16,24 y 32 bits.

9.1.5.1. Compatibilidad entre modos de color. Los modos de color verdadero y color real son ahora los más utilizados, aun en los PC más viejos el modo de color de alta densidad es ampliamente usado. Debido a esta situación el videojuego debe poseer soporte para reproducir y manipular imágenes en modos de color de 16,24 y 32 bits, los modos de 24 y de 32 bits son automáticamente compatibles entre si dada su estructura, pero el modo de 16 bits por la misma razón no es compatible con los modos de 24 y 32 bits (Figura 61), por este motivo es necesario que el motor se encargue o provea los mecanismos de conversión entre diferentes modos de color, puntualmente conversión de modo de 16 a 24 o 32 bits o viceversa (Figura 61), esto resulta una labor crítica al realizar dibujos con color de enmascaramiento. El motor debe brindar los métodos y medios necesarios para realizar conversiones de color entre estos modos.

Figura 61. Compatibilidad entre modos de color



Fuente: Los Autores

9.1.5.2. Modos de color según el modo de resolución. Cuando el dispositivo de video se encuentra funcionando en modo pantalla completa la transformación y/o aplicación del color funciona de manera directa, es decir se hace simplemente volcando el flujo de bits al dispositivo ya que la superficie principal ocupa toda la memoria de video visible, sin embargo cuando el dispositivo se encuentra en modo ventana ocurre un paso adicional, es decir todo el trabajo previo al dibujo directo en pantalla (pe. el copiado a la superficie secundaria) ocurre de la misma manera pero al ser la superficie principal un segmento de la pantalla, cada uno de los colores de esta superficie es transformado al modo de color que se encuentra en el modo de ventana.

Los siguientes ejemplos lo explican de una manera más clara 2 de las 6 situaciones que se pueden presentar:

- Modo pantalla completa esta configurado a 24 bits de color, la superficies principal y secundaria han sido inicializadas a 24 bits de color así que la copia en la superficie principal es directa y se visualiza de inmediato al ser parte de la memoria de video visible.
- Modo ventana esta configurado a 16 bits de color, la superficies principal y secundaria han sido inicializadas a 24 bits de color así que la copia en la superficie principal es directa pero no se visualiza de inmediato, es necesario copiar nuevamente en la sección de la memoria de video visible, en la ventana correspondiente, así que adicional a los cambios de escala y a la conversión de coordenadas[*] se requiere convertir el color de 24 bits a 16 bits.

[*] Ver Cáp. 9.1.4.1. Escalamiento de superficies y 9.1.4.2. Sistema de coordenadas del presente documento

El motor debe tener la capacidad de convertir el color de manera automática al alternar entre modos de resolución, preservando siempre la integridad del color y los valores de enmascaramiento.

9.1.6. Recuperación ante los cambios de contexto (Cambios de modo). Los cambios de modo implican siempre un cambio de contexto para el dispositivo de video, este cambio de contexto trae como consecuencia la corrupción de la información de algunos miembros de los objetos dependientes del dispositivo, es decir el cambio de contexto hace que se dañe la información de las superficies, debido a este inconveniente se hace necesario que el motor implemente rutinas capaces de detectar los daños en las diferentes superficies y reconstruirlas antes de continuar con su uso.

9.1.7. Escalamiento y desplazamiento automático de la imagen el modo ventana. Al cambiar el tamaño de la venta, el motor debe reaccionar automáticamente cambiando las proporciones de copiado del buffer secundario y del buffer principal con el fin de ajustar la memoria de video visible a la posición de la ventana en pantalla y de esta forma se de el efecto de que la imagen cambia su posición y su tamaño de acuerdo al movimiento de la pantalla, de no realizarse estos ajustes se perdería el renderizado de la superficie ya que seria ocultado por las demás ventanas abiertas (Figura 62, 63).

Figura 62. Desplazamiento automático de superficies en modo ventana

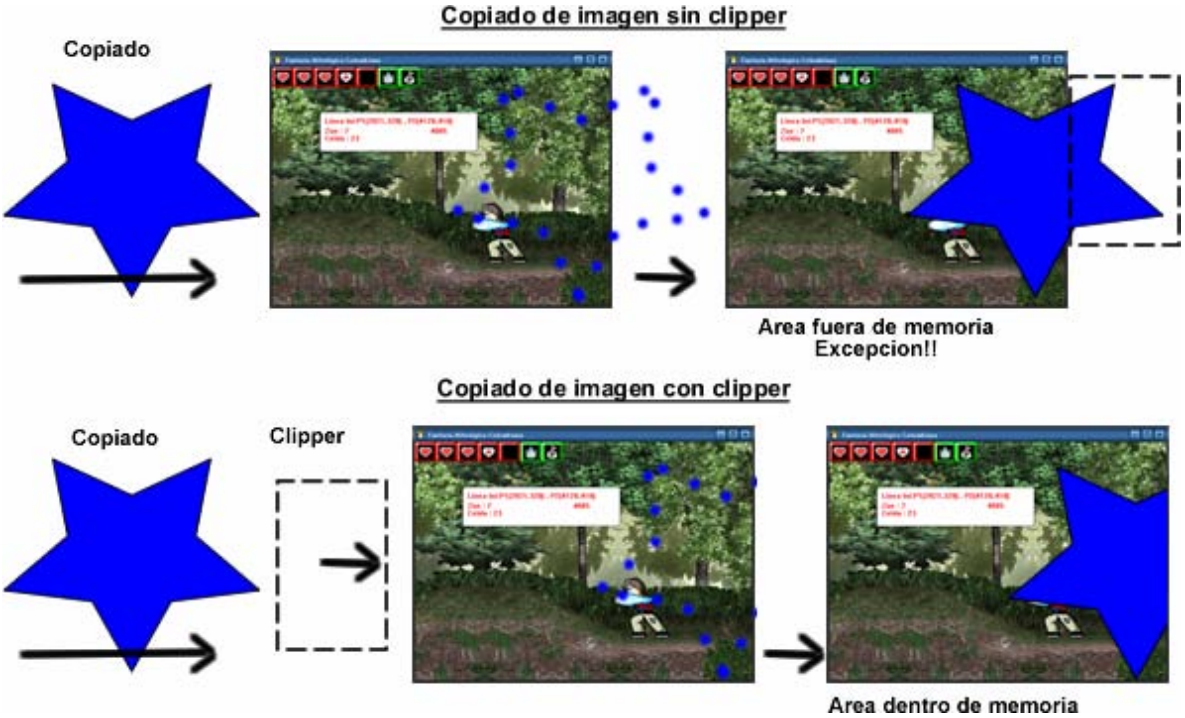
Fuente: Los Autores

Figura 63. Escalamiento automático de superficies en modo ventana

Fuente: Los Autores

9.1.8. Cortador de imágenes (clipper) de la superficie principal. Cuando se dibuja en la superficie principal (memoria de video visible) hay que ser cuidadosos de no dibujar en áreas que se encuentren por fuera del segmento de memoria asignado, ya que si se llega a dibujar por fuera de este segmento se produce un error de excepción. Para evitar ello se hace necesario implementar una rutina capaz de detectar cuando una imagen o una porción de la imagen serán dibujadas en un área fuera de memoria y recortarla hasta obtener una imagen o un segmento de imagen del tamaño justo para no sobrepasar los límites (Figura 64).

Figura 64. Funcionamiento del recortador (clipper)



Fuente: Los Autores

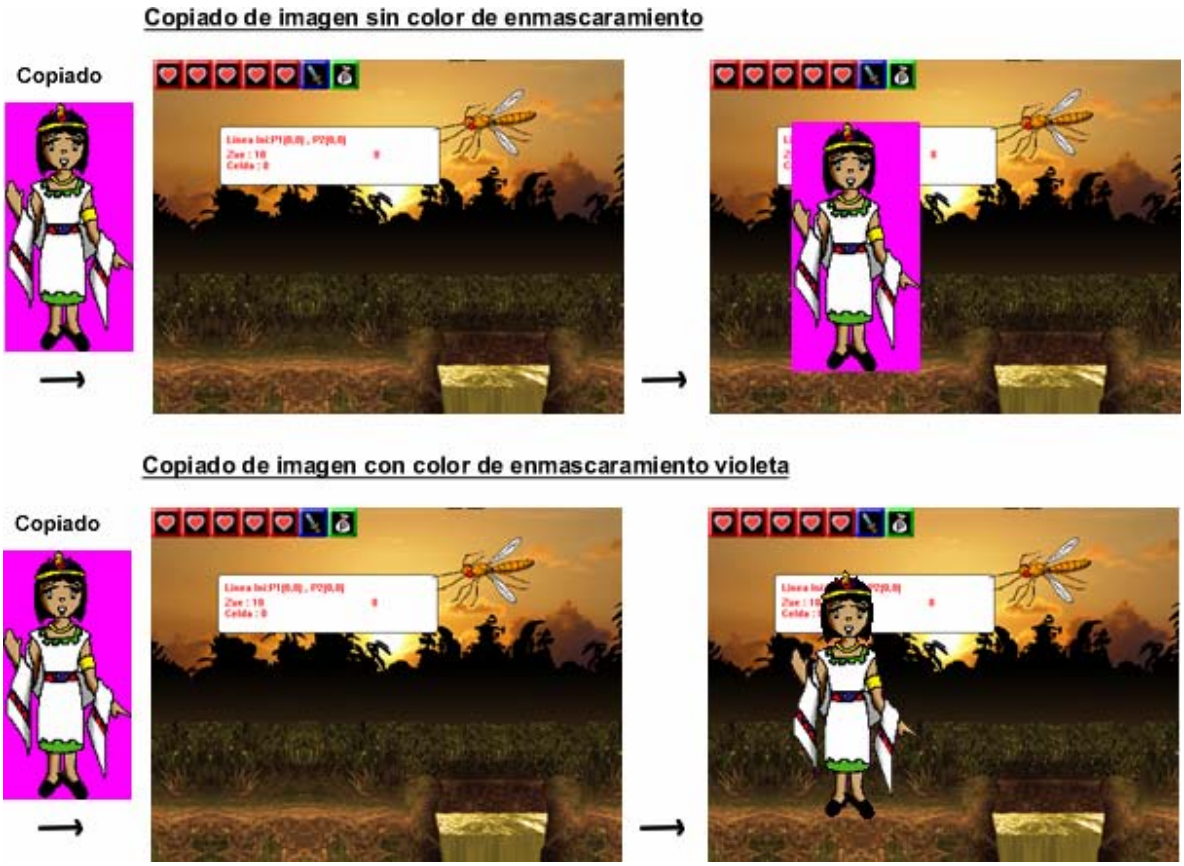
9.2. REQUERIMIENTOS RELACIONADOS CON LAS CARACTERÍSTICAS DE DIBUJO.

Otra parte importante y necesaria para crear un videojuego es la posibilidad de hacer uso de algunas opciones gráficas dentro del motor, estas opciones incluyen efectos variados sobre las imágenes como enmascaramiento y escalamiento los cuales son incluídos en el motor, otros efectos podrían ser incorporados como por ejemplo rotación, semitransparencias, etc. pero no son parte de una construcción de motor básica.

Habitualmente en los videojuegos se utiliza mucho el dibujo de figuras sencillas para mostrar diálogos entre los personajes o para mostrar determinados tipos de información en pantalla, así que el motor también incorpora estas funcionalidades en su nivel básico, de forma habitual también se incorporan funcionalidades para medir el rendimiento del motor, esta implementación muestra los cuadros que se dibujan por segundo (FPS, frames per second) en un instante determinado de animación.

9.2.1. Enmascaramiento de imágenes. El enmascaramiento consiste en indicar al motor que un área determinada de la imagen no sea dibujada en pantalla, normalmente este enmascaramiento se hace haciendo uso de un color de enmascaramiento, es decir que solo se dibujarán las áreas de la imagen que no posean dicho color, Esta característica es la que permite dibujar formas irregulares, no cuadradas, en una superficie de dibujo (Figura 65).

Figura 65. Enmascaramiento, formas irregulares

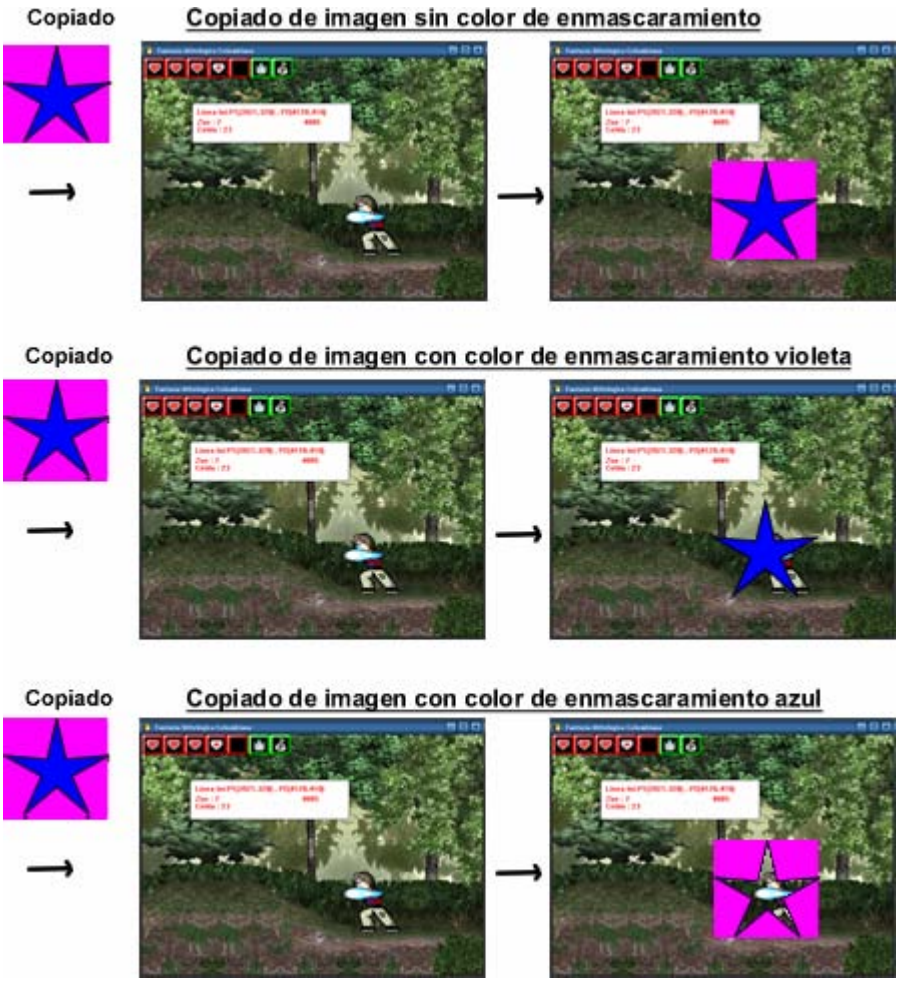


Fuente: Los Autores

El color de enmascaramiento puede ser cualquier color a gusto del desarrollador o del diseñador gráfico, razón por la cual el motor no solo implementa el color de enmascaramiento estándar (RGB = 255, 0, 255 = violeta) sino que soporta cualquier color o gama de color* (Figura 66).

* La gama va definida por un número entero es decir que los colores no están ordenados según un círculo cromático sino según su equivalente al convertirlos a su representación entera de 24 o 32 bits.

Figura 66. Color de enmascaramiento



Fuente: Los Autores

9.2.2. Escalamiento de imágenes. El escalamiento de imágenes permite decidir que tamaño tendrá una imagen al ser dibujada, el cual no necesariamente es el mismo tamaño de la imagen original, esta característica es útil para reutilizar mapas de bits y ahorrar espacio en la memoria principal (Figura 67), el escalamiento no necesariamente debe ser conservando las proporciones de las imágenes, con lo cual se pueden lograr algunos efectos interesantes como aplastamientos, estiramientos, personajes engordados, o un mismo personaje en varios estados diferentes.

Figura 67. Escalamiento de imágenes

Fuente: Los Autores

9.2.3. Reflexión de imágenes. La reflexión es una técnica de dibujo que permite ahorrar memoria principal, consiste en tener una serie de dibujos apuntando a una dirección y luego dibujarla apuntando a la dirección inversa sin tener que poseer una versión invertida de las imágenes sino simplemente realizando el volcado de la matriz de bytes por software o por hardware. En las primeras versiones del motor se incluyo esta funcionalidad para tiempo de ejecución, sin embargo para la ejecución por hardware se hacia necesaria una tarjeta aceleradora de video mas potente que la que tiene el común de las personas y en la gran mayoría de los casos el motor tenia que realizar esta tarea por software para suplir las carencias a nivel de hardware de las tarjetas de video, lo cual hacía a esta característica demasiado lenta para poder ser usada realmente por lo cual fue eliminada pera esta versión.

Las implementaciones de reflexión se delegaron para las capas posteriores del motor en las cuales se usa en tiempo de ejecución a manera de precargue antes de la utilización de las nuevas superficies generadas.

9.2.4. Formas geométricas. El dibujo de formas geométricas es una de las utilidades mas básicas del motor, para esta versión del motor únicamente se ha implementado el dibujo de cuadrados con bordes redondeados ya que es el único que se utilizará para la creación del juego de Fantasía mitológica colombiana, y por cuestiones de tiempo.

La implementación debe soportar el tamaño, el color de fondo, el color de línea y la posición de dibujo.

Dibujo de texto simplificado. Dibujar cualquier texto es fundamental para poder realizar operaciones de depuración y para muchas otras necesidades del juego, razón por la cual el motor posee una sencilla funcionalidad de escritura, la cual soporta posición del texto, color de letras, color de fondo de letra, y cadena de caracteres a dibujar.

Cuadros por segundo (FPS). Esta característica permite habilitar de manera automática la impresión en pantalla de la cantidad de cuadros por segundo, lo cual es un dato importante y muy útil al momento de realizar mediciones de rendimiento.

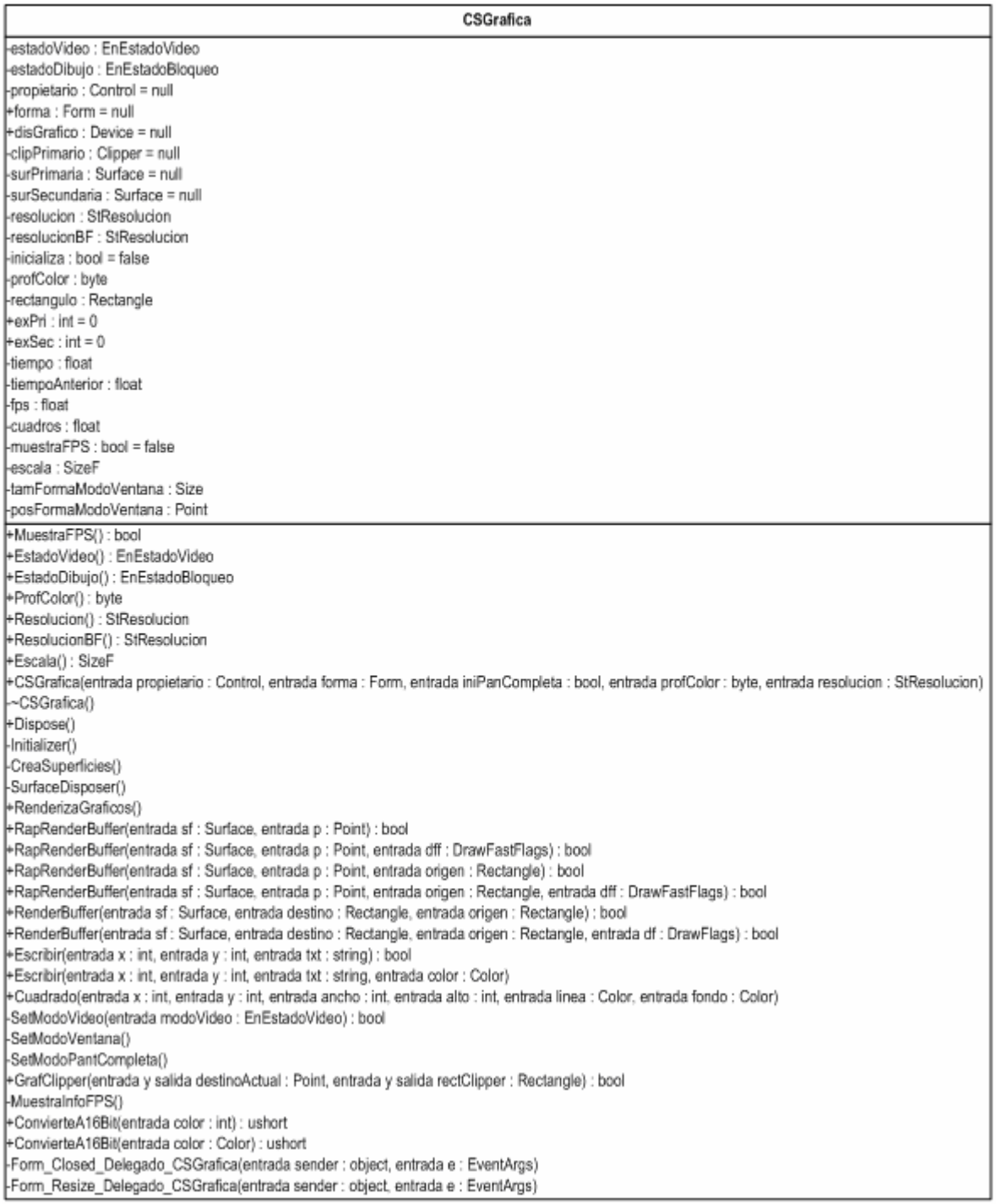
9.3. ABSTRACCIÓN PARA EL USUARIO FINAL

El usuario final debe poder acceder a todos y cada uno de los modos del motor, siendo prioridad del motor reconocer automáticamente si el usuario ha solicitado un parámetro de configuración no soportado por el dispositivo de hardware y subsecuentemente iniciar la recuperación automática a un modo si admitido, todo esto sin que el usuario pierda el control del proceso en ningún momento.

9.4. DIAGRAMA DE CLASES

Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 68. Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A. (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 68. Diagrama de clases del módulo gráfico



Fuente: Los Autores

10. MANEJADOR DE ENTRADA Y SALIDA

Este módulo es el encargado de realizar toda la gestión de los periféricos de entrada y salida soportados por el motor, los periféricos soportados son:

- Mouse
- Teclado
- Joystick

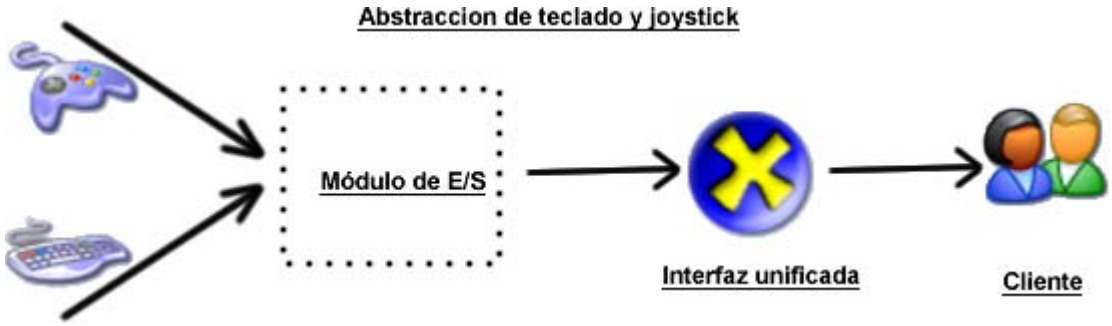
El manejador oculta la complejidad de los dispositivos y los convierte en una interfaz mucho más amigable y fácil de comprender, tiene la función principal de interconectar al juego con el usuario final, es decir es el responsable de realizar la interactividad.

Si bien el módulo se encarga de operar automáticamente toda su funcionalidad, este también provee al cliente la posibilidad de parametrizar cada funcionalidad a través de un conjunto determinado de métodos, sin que el módulo deje de estar al tanto de los posibles errores que esto pudiese generar. La operación del módulo es de manera asíncrona ya que solo requiere procesamiento una vez por ciclo de juego.

10.1. INTERFAZ UNIFICADA PARA TECLADO Y JOYSTICK

10.1.1. Desde el punto de vista del desarrollador. El módulo de E/S brinda una interfaz única para teclado y joystick, es decir el desarrollador cliente no debe preocuparse por saber si esta manejando un dispositivo u otro en un momento determinado, sino que simplemente hace uso de las características que brinda el módulo las cuales son un conjunto de funcionalidades que internamente hacen una abstracción del dispositivo para que este sea visto de una única manera, un ejemplo claro de esta situación se ve representada en la Figura 69.

Figura 69. Abstracción de teclado y joystick



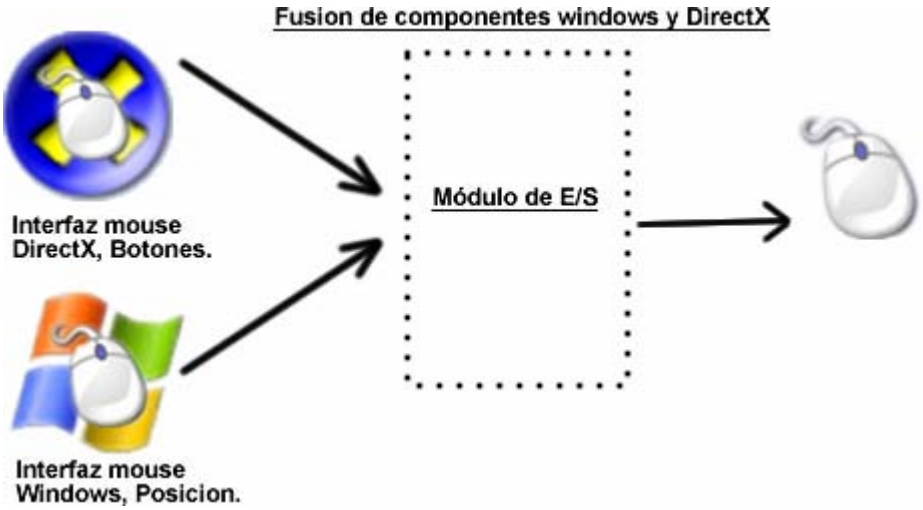
Fuente: Los Autores

10.1.2. Desde el punto de vista del usuario final. Para el usuario final el uso de un dispositivo u otro es muy relevante, sin embargo y pese a ello lo único que le debe preocupar al usuario al momento de su uso es habilitar o deshabilitar una funcionalidad del juego. La intención de brindar un nivel de abstracción al usuario es para que el mismo se despreocupe de los pormenores de la configuración, y sea el propio módulo quien se encargue de resolverla, de este modo si el usuario selecciona que desea usar joystick, pero este no esta conectado o no esta funcionando correctamente, el motor se encarga de pasar automáticamente al modo de uso por teclado.

10.2. INTERFAZ DE TRABAJO PARA EL MOUSE

El módulo ofrece una interfaz simplificada para el manejo del Mouse, en esta versión se puede tener acceso a los botones que se encuentran presionados en un momento determinado, sin embargo para obtener la posición del mouse se debe recurrir a la API del sistema operativo de manera independiente al módulo (Figura 70).

Figura 70. Interfaz de trabajo del mouse

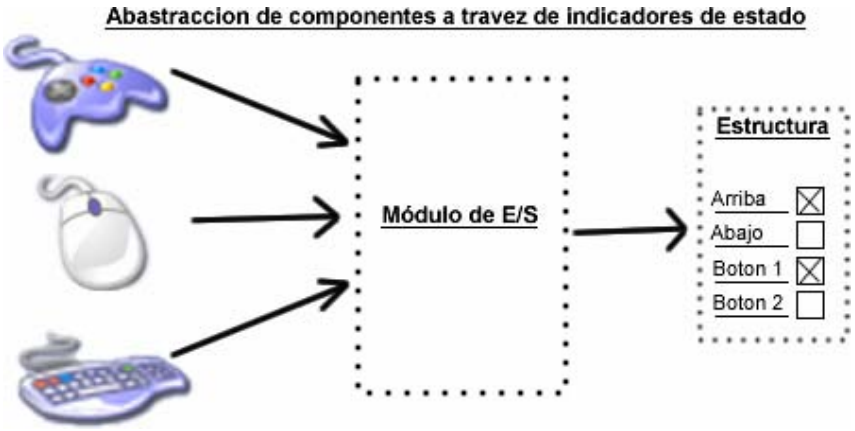


Fuente: Los Autores

10.3. ABSTRACCIÓN DE DISPOSITIVOS

La abstracción de dispositivos se logra utilizando una estructura única de indicadores de estado de dispositivo, de tal manera que al cambiar el estado de uno de los dispositivos el manejador activa algunas de las banderas de la estructura, el desarrollador cliente no necesita acceder nunca directamente al dispositivo sino que accede a los indicadores de la estructura (Figura 71).

Figura 71. Abstracción de componentes



Fuente: Los Autores

De esta forma ya no es necesario saber si esta presionada la flecha derecha del teclado o del Joystick, sino simplemente acceder ala funcionalidad derecha de la interfaz unificada, esto libera al desarrollador de la preocupación de saber si el usuario final utilizará uno u otro dispositivo ya que esta variante la resolverá el módulo de manera automática y la interfaz creada es independiente del dispositivo.

Esta funcionalidad se realiza a través de una simplificación de dispositivos ofrecida por DirectX, dicha simplificación permite intentar obtener el estado de los componentes de cada uno de los dispositivos en un momento determinado, cuando se habla de componentes se hace referencia a las partes funcionales del hardware, en el caso del mouse son cada uno de los botones, la rueda (si la posee), etc., para el joystick es cada uno de los botones, la cruz de direcciones y los demás aditamentos que pose, mientras que para el teclado es cada una de las tecla que se encuentran presionadas en un momento determinado.

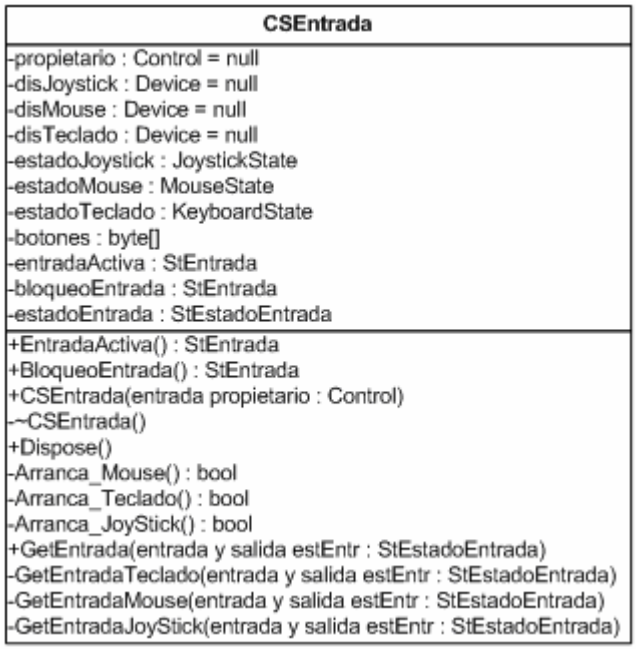
10.4. SOPORTE MULTIJUGADOR

El módulo es una implementación básica de un manejador de E/S razón por la cual tiene soporte para un conjunto de dispositivos a la vez, es decir en esta versión el módulo no posee soporte para un jugador.

10.5. DIAGRAMA DE CLASES

Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 72. Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de bits desarrollo) y en los capítulos 5 y 6.

Figura 72. Diagrama de clases del módulo de entrada y salida



Fuente: Los Autores

11. MANEJADOR DE SONIDO

El módulo manejador de sonido es el encargado de realizar la reproducción de sonidos dentro del videojuego, es importante diferenciar los sonidos de la música puesto que los sonidos son generalmente fragmentos cortos que se encuentran en archivos no comprimidos, ideales para una reproducción rápida, mientras que la música son archivos grandes y codificados que toman mayor tiempo para su cargue en memoria y tienen un uso mas alto de procesamiento para las operaciones de decodificación, por tanto el tamaño del módulo y de cada uno de los componentes del módulo reproductor de sonido deben ser lo mas pequeños posibles ya que dependiendo la complejidad de la escena suelen existir múltiples instancias de la clase cargadas a la vez.

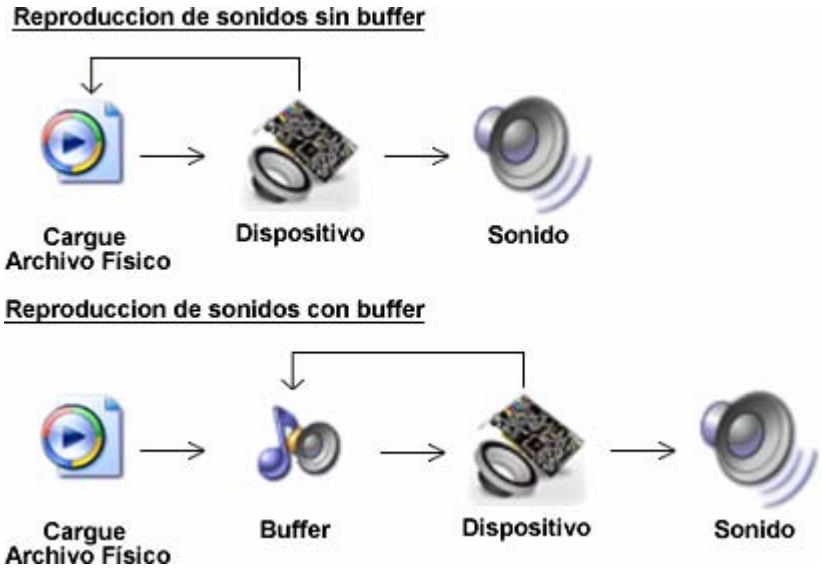
La reproducción de sonidos es síncrona dentro de su contexto pero es asíncrona internamente:

- Síncrona: Internamente no se puede iniciar la reproducción de un sonido sin que de alguna manera haya terminado la reproducción de un sonido previo o el dispositivo este disponible (salvo en múltiples instancias).
- Asíncrona: Al interactuar con los demás componentes del motor se debe iniciar la reproducción del sonido y aunque esta no haya finalizado, dar pie al procesamiento de las demás actividades del motor sin detener el sonido que se reproduce.

11.1. FUNCIONALIDADES

11.1.1. Funcionalidades clásicas de reproducción. El módulo funciona a manera de reproductor multimedia, es decir posee funcionalidades de inicio, pausa, parada, repetición, cargue y descargue, las funcionalidades de cargue y descargue se adecuan a las necesidades inherentes a un juego. Estas funcionalidades se realizan sobre una simplificación del dispositivo de sonido que es provista por Directx, esta simplificación es capaz de reproducir un sonido no codificado (formatos crudos como el wav, midi, etc.), sin embargo cada vez que se realiza la reproducción de sonido el archivo de audio es nuevamente cargado en el dispositivo lo cual es particularmente ineficiente a la hora de jugar pues los tiempos de carga oscilan entre 0.2 y 1 segundos, esto implicaría que si un personaje comienza a hablar sus labios se moverían un segundo antes de que el sonido comenzara, otro ejemplo seria que un objeto fuera golpeado y el sonido producto del choque ocurriera un segundo después de haber sucedido. Por estas razones se hace necesario tener los sonidos previamente cargados en memoria para que una vez se necesite reproducirlos simplemente sea necesario pasarlos al dispositivo a reproducción, de esta manera se ahorran los tiempos de cargue ya que solo se harían una vez (Figura 73).

Figura 73. Reproducción de sonidos



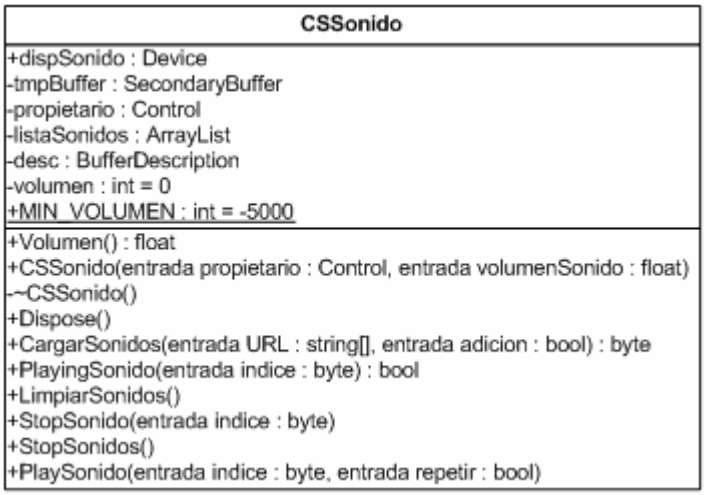
Fuente: Los Autores

11.1.2. Lista de reproducción. Debido a que un personaje o un objeto capaz de generar sonidos generalmente no requiere de un solo sonido sino que requiere de dos o más, se hace necesario generar una lista de reproducción para el dispositivo, para lo cual se genera con el uso de dos arreglos, uno de lista donde están los nombres de cada uno de los sonidos a reproducir y un segundo arreglo es una colección indexada de espacios de memoria que sirven de buffer para el precargue de sonidos. Cada uno de los miembros de la colección de búferes posee su propio atributo de volumen, lo cual permite realizar una nivelación de sonidos previa a su reproducción.

11.2. DIAGRAMA DE CLASES

Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 74. Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 74. Diagrama de clases del módulo de Sonido



Fuente: Los Autores

12. MANEJADOR DE MÚSICA

El módulo manejador de música es el encargado de realizar la reproducción de música dentro del videojuego, contraria a la reproducción de sonidos la música son archivos grandes y codificados que toman mayor tiempo para su cargue en memoria y tienen un uso mas alto de procesamiento para las operaciones de decodificación, por tanto sus componentes son objetos más pesados para un control avanzado de reproducción. El módulo generalmente es cargado solo una instancia por vez por lo cual se minimiza el impacto a nivel de rendimiento.

La reproducción de música es síncrona dentro de su contexto pero es asíncrona internamente:

- Síncrona: Internamente no se puede iniciar la reproducción de una pista sin que de alguna manera haya terminado la reproducción de un sonido previo o el dispositivo este disponible (salvo en múltiples instancias).
- Asíncrona: Al interactuar con los demás componentes del motor se debe iniciar la reproducción del música y aunque esta no haya finalizado, dar cabida al procesamiento de las demás actividades del motor sin detener el sonido de la pista que se reproduce.

12.1. FUNCIONALIDADES

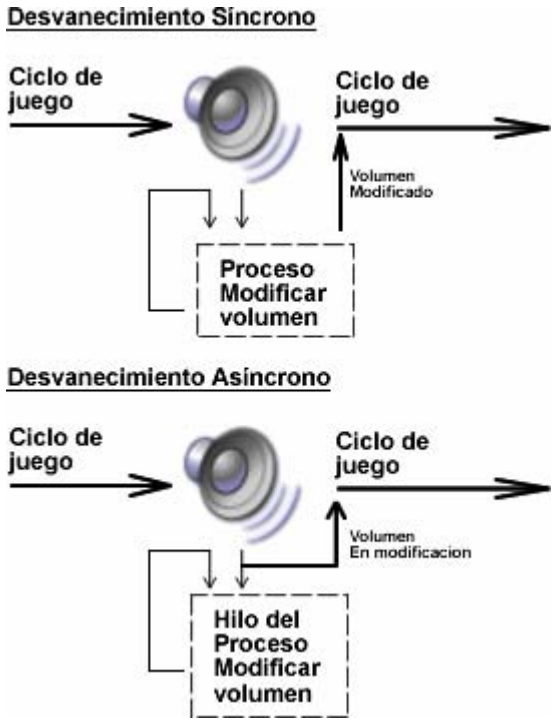
12.1.1. Funcionalidades clásicas de reproducción. El módulo al igual que el reproductor de sonido funciona a manera de reproductor multimedia, es decir posee funcionalidades de inicio, pausa, parada, repetición, cargue y descargue, las funcionalidades de cargue y descargue se adecuan a las necesidades inherentes a un juego. Estas funcionalidades se realizan sobre una simplificación del dispositivo de sonido que es provista por Directx (DirectX.AudioVideoPlayback), esta simplificación es capaz de reproducir música codificada (formatos complejos como el wma, mp3, asf etc.), cada vez que se realiza la reproducción de música el archivo de audio es nuevamente cargado en el dispositivo y este internamente segmenta la información en fracciones de 5 o 10 segundos dependiendo la complejidad de la onda, de esta manera el reproductor automáticamente genera sus propios búferes y no requiere intervención por parte del desarrollador.

Desvanecimientos (cross fading). Los efectos de desvanecimiento son complejos de realizar de manera asíncrona, sin embargo esa funcionalidad que se requiere para un juego puesto que algunas veces debe efectuarse el desvanecimiento mientras la acción continua dentro del juego, y otras veces se debe realizar hasta finalizar para luego continuar con la acción del juego.

Existen desvanecimientos positivos y negativos, en el primero el volumen de la música inicia desde un nivel de sonido casi nulo hasta su nivel de sonido máximo, y en el segundo el sonido inicia desde el nivel máximo y disminuye hasta casi nulo.

En el desvanecimiento síncrono basta con disminuir o aumentar el nivel de sonido en intervalos regulares de tiempo hasta obtener el resultado final, mientras que en el asíncrono se requiere disparar el mismo proceso que para el síncrono pero de manera paralela, es decir se requiere disparar un hilo de ejecución que se encargue de operar sobre los niveles de volumen mientras que los demás componentes del motor continúan su ejecución (Figura 75).

Figura 75. Funcionamiento de los desvanecimientos

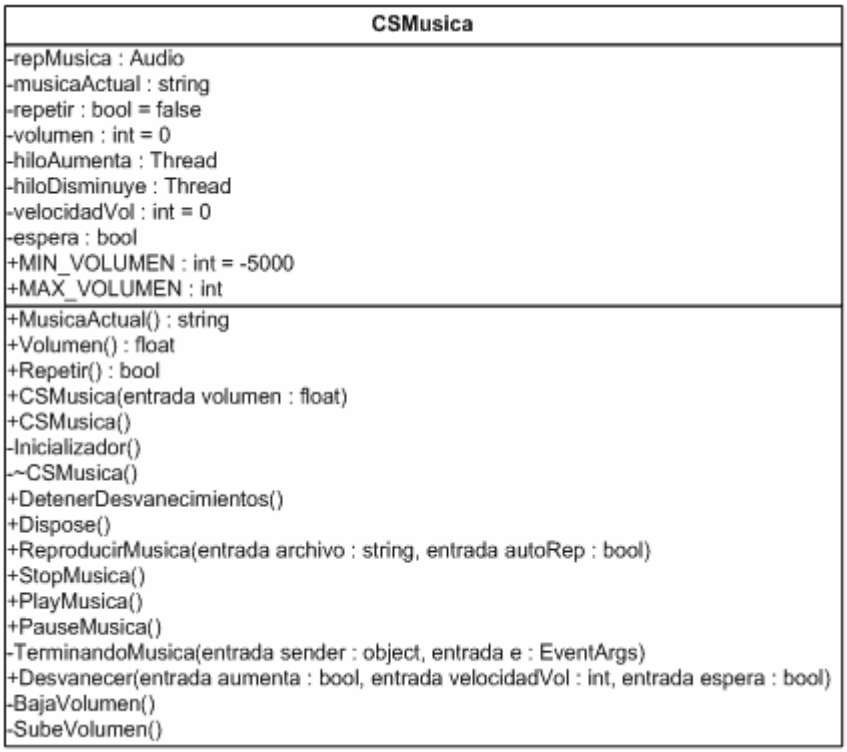


Fuente: Los Autores

12.2. DIAGRAMA DE CLASES

Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 76. Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 76. Diagrama de clases del módulo de Música



Fuente: Los Autores

13. MANEJADOR DE VIDEO

El módulo manejador de video es el encargado de realizar la reproducción de video clips dentro del videojuego, es uno de los componentes más pesados ya que aunque su implementación es pequeña hace uso del componente DirectX.AudioVideoPlayBack.Video para funcionar, esta clase es la implementación de video básica que posee la librería DirectX.

13.1. FUNCIONALIDADES

13.2.

13.2.1. Funcionalidades clásicas de reproducción. El módulo al igual que el reproductor de video funciona a manera de reproductor multimedia, es decir posee funcionalidades de inicio, pausa, parada y repetición pero no posee funcionalidades de cargue y descargue por cuanto su ejecución es asíncrona con los demás componentes del juego y consigo mismo es síncrona, es decir no se reproduce otro video mientras ya exista uno en ejecución (salvo en múltiples instancias), pero si se ejecutan otros componente del videojuego mientras se este reproduciendo el video. Es conveniente tener en claro que internamente el componente de video de DirectX también hace uso de DirectX.AudioVideoPlayBack.Audio.

Estas funcionalidades se realizan sobre una simplificación del dispositivo de video que es provista por Directx (DirectX.AudioVideoPlayback), esta simplificación es capaz de reproducir video codificado (formatos complejos como el wmv, mpg, avi etc.).

Aunque el módulo no posee una lista de reproducción persistente, si es posible indicarle al momento de iniciar la reproducción que ejecute un archivo tras otro.

13.2.2. Reproducción en modo pantalla completa. En este modo el módulo de video oculta la interfaz tradicional del juego y la suspende mientras que inicia la reproducción en primer plano en una superficie independiente, una vez terminada la reproducción el módulo se descarga y devuelve el control a la interfaz tradicional del juego.

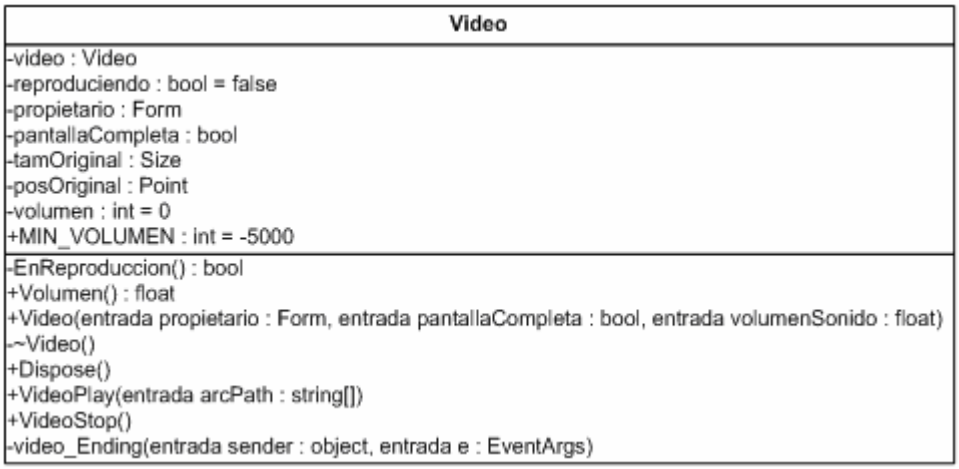
13.2.3. Reproducción en modo ventana. En este modo el módulo de video se integra en la interfaz tradicional del videojuego asumiendo las propiedades propias de la ventana en ejecución, para evitar errores en la interfaz de dibujo de la ventana y en la interacción de esta con los demás componentes del sistema operativo se realizan llamados al proceso del bucle de mensajes de ventana recurrentes en tanto el video este en ejecución.

13.2.4. Ínter bloqueo de procesos externos. El módulo no puede hacerse responsable de bloquear los demás procesos que se encuentren en el bucle de ejecución del videojuego ya que desconoce cuáles son y no tiene medios para conocerlos, por esta razón es responsabilidad del desarrollador deshabilitar los componentes que resulte conveniente, en general se hace necesario deshabilitar las clases de dibujo que afecten al interfaz clásica de juego ya que se podría entrar en conflictos de acceso a la superficie de dibujo.

13.3. DIAGRAMA DE CLASES

Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 77.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 77. Diagrama de clases del módulo de Video

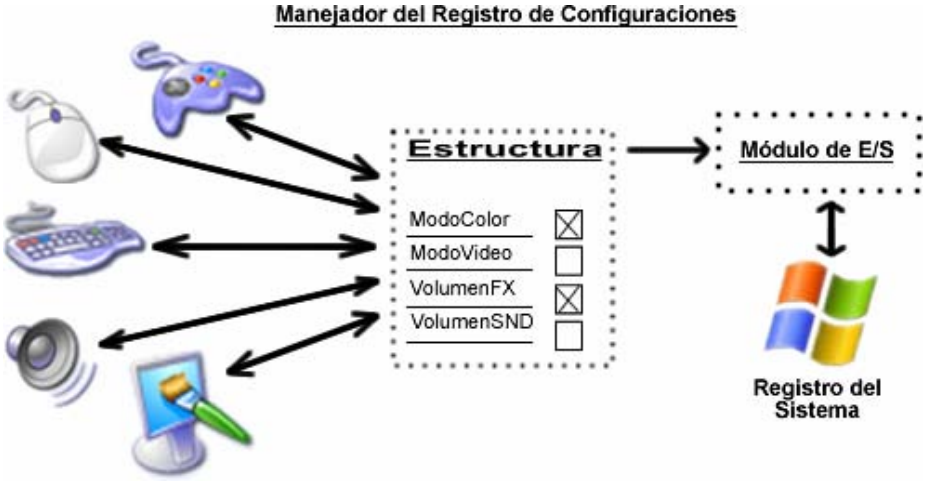


Fuente: Los Autores

14. MANEJADOR DE REGISTRO

El módulo manejador de registro es el encargado del cargue y descargue de configuraciones del juego al registro del sistema operativo (Windows). Existe una estructura que almacena cada una de las propiedades soportadas y las incorpora al registro del sistema, o bien puede ser leída por los diferentes componentes del juego para la configuración automática en el proceso de cargue (Figura 78).

Figura 78. Manejador del registro de configuraciones



Fuente: Los Autores

14.1. CONFIGURACIONES ALMACENADAS

Las configuraciones que es capaz de almacenar el módulo en l registro se enlistan a continuación:

- Resolución horizontal de pantalla
- Resolución vertical de pantalla
- Profundidad del color
- Volumen de sonido
- Volumen de música
- Iniciar en pantalla completa
- Usar mouse en los menús
- Usar teclado
- Posición x de ventana
- Posición y de ventana

- Tamaño x de ventana
- Tamaño y de ventana

14.2. DIAGRAMA DE CLASES

Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 79.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 79. Diagrama de clases del módulo de registro de configuraciones

Fuente: Los Autores

15. MANEJADOR DE PARTIDAS

El módulo manejador de partidas crea una Interfaz de archivos XML que permite almacenar y recuperar el estado de cada una de las partidas soportadas por el juego.

15.1. ESTRUCTURA DEL ARCHIVO XML

El archivo XML tiene una estructura capaz de soportar un número indeterminado de partidas y uno indeterminado de escenas, estos parámetros se configuran desde el constructor de la clase.

La estructura soporta el manejo de ítems y su cantidad, el estado de cada escena, y el estado del personaje lo cual conforma la estructura que se ve a continuación en el siguiente fragmento XML.

Figura 80. Fragmento XML del módulo manejador de partidas

```
<Partidas Juego="FMCZ">
 <Partida Nombre="claudio">
  <Zue Vidas="0" Energia="0">
   <Items ItemActual="0" Cantidad="0" />
  </Zue>
  <Escenas EscenaActual="1">
   <Escena Numero="1" Estado="NoIniciada" />
   <Escena Numero="2" Estado="NoIniciada" />
   <Escena Numero="3" Estado="NoIniciada" />
   <Escena Numero="4" Estado="NoIniciada" />
  </Escenas>
 </Partida>
 <Partida Nombre="Logan">
  <Zue Vidas="0" Energia="0">
   <Items ItemActual="0" Cantidad="0" />
  </Zue>
  <Escenas EscenaActual="1">
   <Escena Numero="1" Estado="NoIniciada" />
   <Escena Numero="2" Estado="NoIniciada" />
   <Escena Numero="3" Estado="NoIniciada" />
   <Escena Numero="4" Estado="NoIniciada" />
  </Escenas>
 </Partida>
</Partida>
```

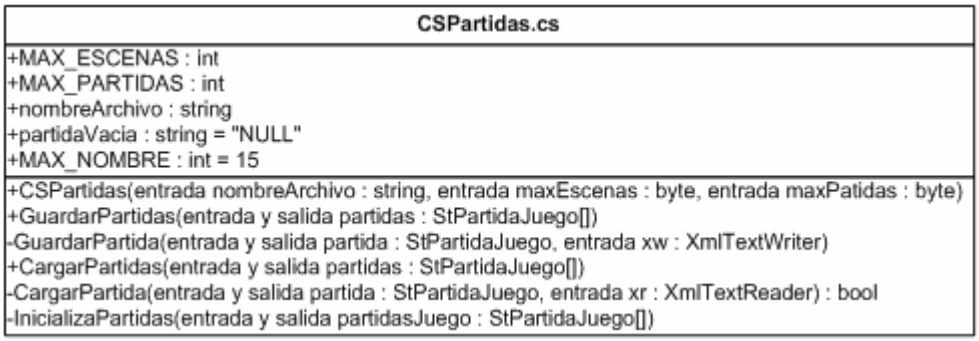
Fuente: Los Autores

El descriptor general de la estructura XML se encuentra en los Anexos B y C (descriptor XSD para la estructura XML del manejador de partidas)

15.2. DIAGRAMA DE CLASES

Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 81.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 81. Diagrama de clases del módulo manejador de partidas



Fuente: Los Autores

16. MANEJADOR DE ACTORES

El módulo manejador de actores tiene el objetivo, como su nombre lo indica, de centralizar la administración de los personajes involucrados en un momento determinado del juego, esto simplifica los llamados a los diferentes métodos de los actores ya que todos poseen interfaces comunes.

El módulo posee dos funciones principales:

- Simplificar el desarrollo de módulos donde se involucren uno o mas actores
- Enviar mensajes y requerimientos a cada uno de los actores, independientemente de la clase que implemente cada uno.

Básicamente el manejador administra un arreglo que contiene en cada uno de sus nodos objetos que aunque sean de clases diferentes que poseen una interfaz común.

16.1. ACTORES

Un actor es cualquier elemento que intervenga dentro del juego de manera interactiva, puede ser desde una piedra hasta el personaje principal, existen elementos que no se consideran como actores debido a su complejidad para el dibujo y a su gran tamaño, estos elementos se han trabajado como escenas lo cual se explica en el capitulo 18.

Un actor esta definido por dos componentes fundamentales:

- Un dibujo que lo representara en pantalla
- Un conjunto de funcionalidades definidas por un programador para interactuar con el juego

Los actores alternan entre diferentes estados para establecer sus diferentes comportamientos, de este modo un actor en estado ‘caminando’ alterna entre diferentes cuadros de animación para que se vea la imagen caminando, el mismo actor en estado ‘volando’ alterna sobre un conjunto de dibujos diferentes para que se vea su imagen volando.

16.1.1. Dibujo que representa al actor. Puede ser cualquier clase de dibujo, pero se debe tener en consideración el tamaño de este ya que un tamaño exagerado ocuparía demasiados recursos del sistema como memoria y tiempo de procesamiento para dibujarlo (Figura 82). El actor puede hacer uso de todas las facultades de dibujo brindadas por el motor, como el uso de color de enmascaramiento, escalabilidad, posicionamiento etc.

Figura 82. Dibujo Actor

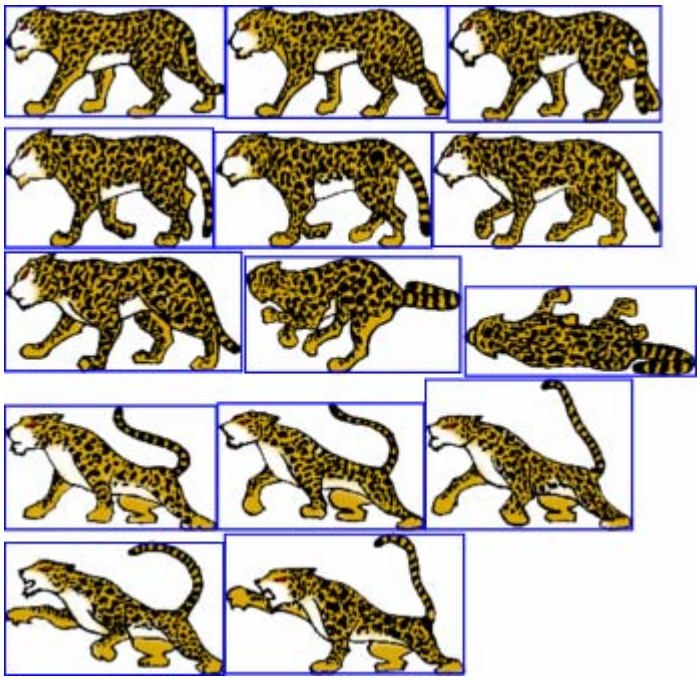
Fuente: Los Autores

Sin embargo un dibujo sencillo no es suficiente si el actor incorpora movimientos dentro de su comportamiento, se requiere un conjunto de imágenes que describan una secuencia animada, en el ejemplo de la Figura 83 se pueden apreciar las imágenes que componen la secuencia de movimiento del personaje, en la Figura 84 se puede apreciar un conjunto de imágenes más elaborado que describe un juego de varios movimientos diferentes.

Figura 83. Dibujo Actor con movimientos (hablando)

Fuente: Los Autores

Figura 84. Dibujo Actor con movimientos elaborados (tigre y sus diferentes movimientos)



Fuente: Los Autores

16.1.2. Funcionalidades. Las funcionalidades son creadas por programación, se pueden distinguir dos grupos de funcionalidades:

- Internas
- Externas

16.1.2.1. Funcionalidades Internas. Son las funcionalidades que interactúan de forma independiente con el ciclo habitual del juego, se ejecutan desde un llamado interno del actor. Entre estas funcionalidades tenemos:

- Reproducción de sonidos de actor
- Cambio de estado el actor
- Inteligencia Artificial

16.1.2.2. Funcionalidades Externas. Son las funcionalidades que interactúan de forma conjunta con el ciclo habitual del juego, generalmente desde el bucle principal de juego o desde uno de sus componentes. Entre estas funcionalidades tenemos:

- Procesamiento del siguiente cuadro
- Dibujo del cuadro actual en pantalla
- Procesamiento de dispositivos de entrada.

16.1.3. Tipos de Actor. Los actores están diferenciados en dos tipos de grupo

- Por su modo de interacción con el usuario
- Controlables
- No controlables
- Por la forma en que usan los recursos
- De dibujo normal
- De dibujo invertido

16.1.3.1. Actor controlable. Son los actores que poseen una interfaz con los dispositivos de entrada y salida para de esta manera interactuar con el usuario, es decir cambia de estado únicamente ante la acción del usuario final en uno de los dispositivos de entrada y salida manejados por el motor.

16.1.3.2. Actor no controlable. Son los actores que no poseen una interfaz con los dispositivos de entrada y salida, sus estados varían únicamente producto de la inteligencia artificial implementada internamente por el actor, o bien no posee sino un único estado.

16.1.3.3. Actor de dibujo normal. Son los actores que siempre se dibujan en una sola dirección, es decir sus dibujos siempre se dibujan tal cual como vienen en el archivo gráfico que compone al actor, generalmente este tipo de actor se usa en elementos estáticos de la pantalla (Figura 85).

16.1.3.4. Actor de dibujo invertido. Son los actores que no siempre se dibujan en una sola dirección, es decir sus dibujos pueden dibujarse tal cual como vienen en el archivo gráfico que compone al actor, o bien dibujar su reflejo, esto se usa para economizar espacio de memoria en la carga la imagen, ya que se carga solamente un juego de imágenes apuntando a una sola dirección y las demás se obtienen por reflexión de esta imagen (Figura 85). Inicialmente la reflexión iba a ser una característica soportada por el motor para ser realizada por hardware, pero debido a que la gran mayoría de personas en nuestro entorno no posee tarjetas aceleradoras de gráficos de mayores capacidades, se tomo la decisión de eliminar esta característica del motor y de implementarla por software en las capas superiores como lo es cada uno de los actores, de este modo al inicio de su cargue el actor carga el conjunto de imágenes que lo representa e inmediatamente después crea una copia reflejada de esta imagen en memoria. Se utiliza esta reflexión en la etapa de cargue puesto que la reelección por software es lenta para hacerla al momento directo del dibujo, así que se crea una copia a manera de caché de memoria para poder utilizarla luego rápidamente.

Figura 85. Tipos de actor por la forma en que usan los recursos

Fuente: Los Autores

16.1.4. Estados de Actor. Los actores comparten una seria de estados que indican al administrador de actores, al manejador de escenas o al desarrollador que están realizando una u otra actividad en un momento determinado, estos estados son:

- Muerto
- Quieto
- Movimiento
- Suspendido
- Salir Escena

16.2. FORMATO GRÁFICO GRI*.

Como se puede apreciar no basta con una imagen sencilla para brindar la información relevante a un personaje, cada personaje es diferente del otro en cuanto al numero de cuadros de animación que lo componen; Debido a la cantidad de trabajo que tiene elaborar un personaje no solo a nivel artístico sino también a nivel de programación, se requiere realizar una implementación de personajes bastante genérica con el fin de reducir los tiempos de desarrollo de manera considerable (aunque de igual forma larga).

Para dar solución a esta necesidad se ha diseñado un formato gráfico propio que brinda al desarrollador información complementaria al dibujo que compone el actor, para esta versión se ha incluido como información complementaria los siguientes aspectos:

- Número de cuadros existentes en la imagen
- Posición de cada cuadro dentro de la imagen
- Cantidad de archivos de sonido asociados al actor
- Nombre de los archivo de sonido asociados al actor

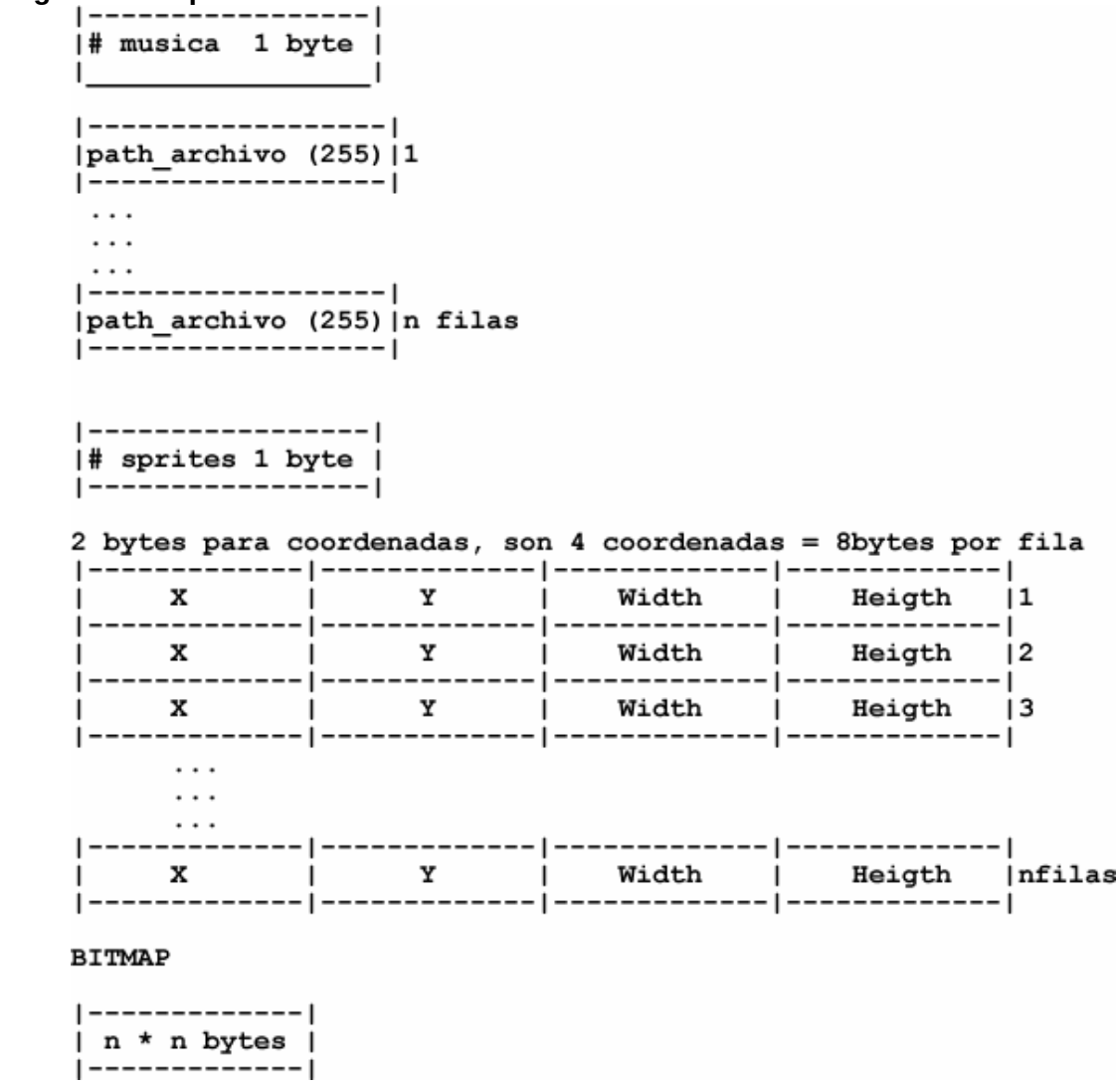
16.2.1. Especificación del archivo. Para cumplir con las especificaciones planteadas se ha definido la estructura del archivo como se ve en la Figura 86.

El archivo posee embebida una imagen tipo bitmap pero en su encabezado se han agregado varios campos adicionales:

- Música: este campo siempre ocupa un byte, aquí se encuentra la información del número de archivos de sonido asociados al actor.
- Path archivo: este campo ocupa 255 bytes por cada valor encontrado en # Música, ya que cada 255 bytes contiene la información relevante al nombre del archivo de sonido asociado al actor.
- Sprites: este campo ocupa un byte, aquí se encuentra la información del número de cuadros que posee la imagen en el archivo es decir máximo 225 cuadros, con esta información seguidamente se puede calcular la lectura para el campo descripción rectangular.

* GRI es la sigla de Gómez Ruiz Imagen, que es la extensión que se le dio al formato gráfico, Gómez, Ruiz son los apellidos de los integrantes de este proyecto.

Figura 86. Especificación del formato GRI



Fuente: Los Autores

Descripción rectangular, la cual ocupa 8 bytes por cada valor encontrado en Nro. De sprites, ya que cada ocho (8) bytes contiene la información relevante a cada cuadro dentro de la imagen es decir:

-posición inicial en el eje ‘x’ 2 byte
-posición inicial en el eje ‘y’ 2 byte
-dimensiones horizontal 2 bytes
-dimensión vertical 2 bytes

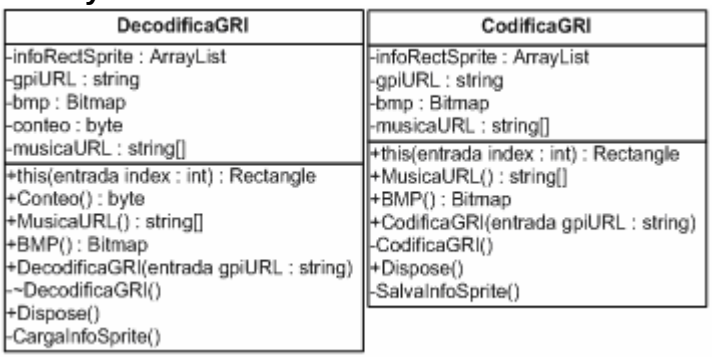
Esta estructura de datos se repite tantas veces como lo indique #Sprites.

Bitmap, El mapa de bytes que contiene la imagen, soporta cualquier formato pero lo más recomendable es el uso de imágenes en formato BMP.

16.2.2. Codificador y Decodificador GRI. Para lograr crear un archivo GRI es necesario tener una herramienta de software capaz de plasmar todas las características definidas para el tipo de archivo, por tal razón ser ha creado un módulo de codificación y una aplicación para tal fin, la aplicación adicionalmente requiere poder abrir un archivo de este tipo razón por la cual es necesario implementar de igual forma un decodificador de archivo.

El codificador y el decodificador poseen la estructura que se ve en la Figura 87.

Figura 87. Codificador y decodificador del formato GRI

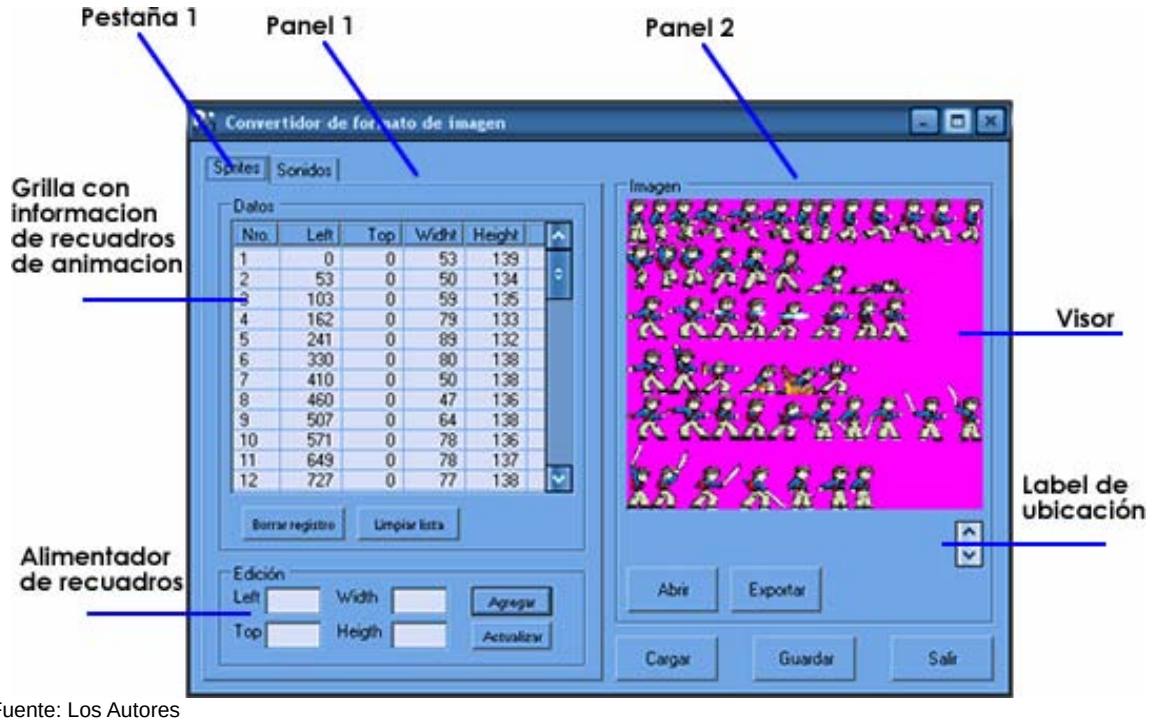


Fuente: Los Autores

16.2.3. Aplicación convertidor GRI. Esta aplicación es capaz de convertir un archivo de casi cualquier formato gráfico al formato GRI, la aplicación permite alimentar los diferentes conjuntos de datos que requiere el archivo.

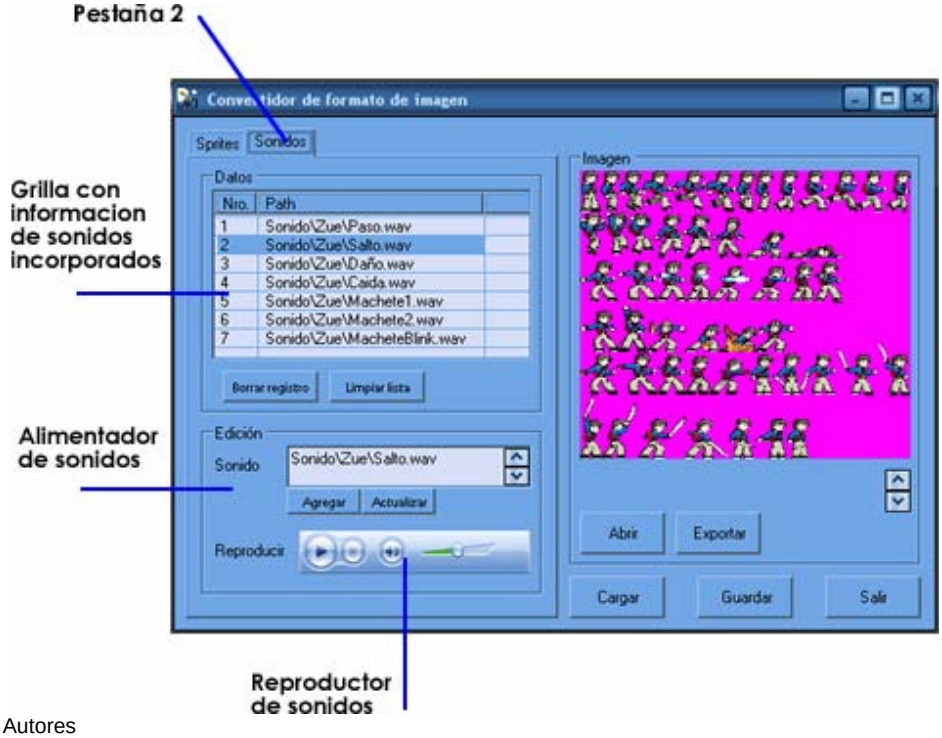
Consta de dos paneles, uno de ellos muestra el mapa de bits que se esta trabajando y permite cambiarlo o exportarlo a un formato gráfico comprensible, el otro posee dos pestañas la primera para alimentar los datos relevantes a cada cuadro de animación integrado con el dibujo y la segunda para adicionar información musical al archivo (figuras, 88,89).

Figura 88. Aplicación convertidor formato GRI, 1a pestaña



Fuente: Los Autores

Figura 89. Aplicación convertidor formato GRI, 2da pestaña



Fuente: Los Autores

La aplicación esta diseñada para permitir al artista incorporar la información de la imagen de la manera más rápida y eficiente posible, el reproductor musical incorporado permite testear los sonidos asociados a la imagen. El diagrama de clases de la aplicación Convertidor GRI, se puede apreciar en la Figura 90.

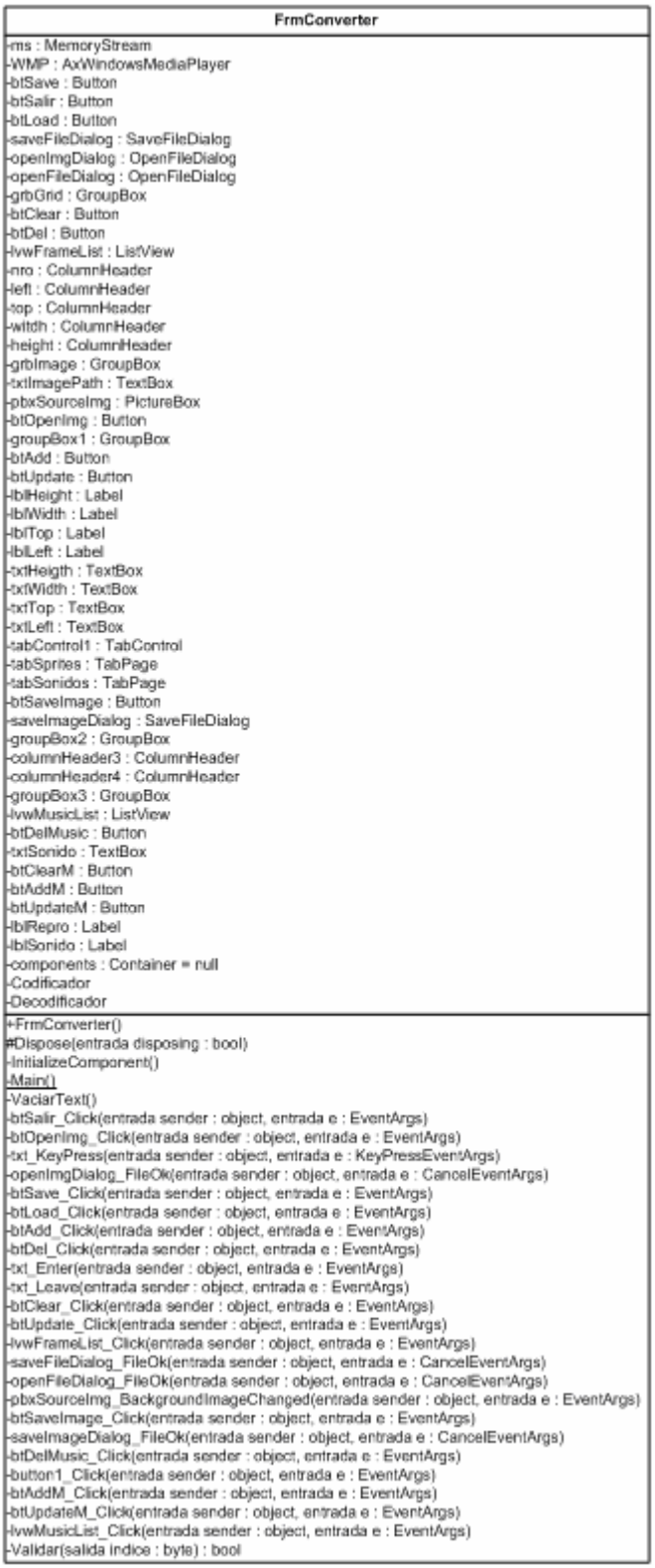
16.2.4. Funcionamiento aplicado al entorno de programación del juego. El codificador únicamente funciona en la aplicación del convertidor GRI, ya que desde ningún juego se utilizaría para crear archivos GRI. El codificador permite cargar el archivo gráfico desde el constructor y una vez allí permite disponer de los recursos asociados al archivo a través de indexadotes de objetos para los diferentes rectángulos que definen los sprites de la imagen y a través de un arreglo de cadenas para indicar la ubicación de los diferentes archivos gráficos asociados. Dicho de otra forma para acceder a cada uno de los cuadros de animación de un actor se haría así:

actor[1], actor[2], actor[3], actor[4]…

Para acceder a la música asociada al actor se haría así

Actor.MusicaURL[1],Actor.MusicaURL[2]…

Figura 90. Diagrama de clases convertidor GRI



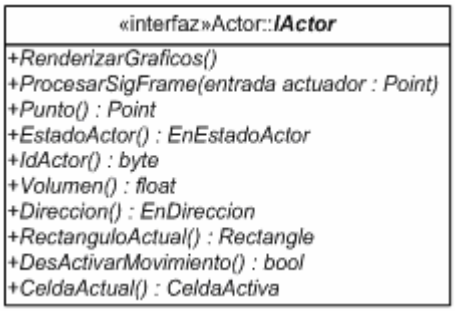
Fuente: Los Autores

16.3. DEFINICIÓN DE INTERFACES PARA ACTORES.

De acuerdo a los diferentes tipos de actores* que se han creado dos interfaces primarias con el fin de que sea a través de estas que el administrador de actores pueda cumplir su función.

Interfaz IActor. Esta interfaz define los métodos, atributos, propiedades etc. que debe poseer una clase para ser reconocida como un actor por el administrador de actores, ver Figura 91.

Figura 91. Interfaz IActor



Fuente: Los Autores

16.3.1.1. Propiedades

- CeldaActual: Retorna o Establece la celda actual sobre la cual se encuentra un personaje en pantalla.
- DesActivarMovimiento: Establece si se activa o desactiva el desplazamiento en pantalla de un personaje
- Direccion: Retorna la dirección del actor
- EstadoActor: Retorna o establece el estado IActor del actor
- IdActor: Retorna el ID del actor que usara el administrador de partidas
- Punto: Retorna o establece un Punto donde se encuentra el actor
- RectanguloActual: Retorna el rectángulo actual de dibujo
- Volumen: Retorna o establece el volumen del sonido del actor

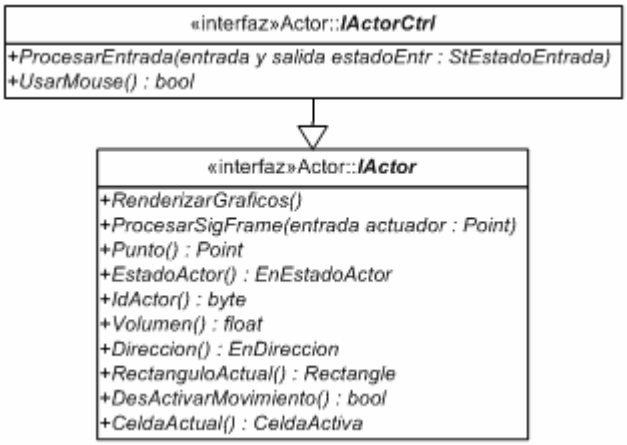
* Revisar Cáp. 16.1.3. Tipos de Actor en el presente documento

16.3.1.2. .Métodos

- ProcesarSigFrame: Método que lleva a cabo los cálculos relevantes al la manera en que se comporta un actor
- RenderizarGraficos Método que permite dibujar en al superficie secundaria de una clase gráfica.

16.3.2. Interfaz IActorCtrl. Esta interfaz define todos los miembros de Iactor, pero adicionalmente incorpora funcionalidades para el manejo de entrada y salida que debe poseer una clase para ser reconocida como un actor que interactúa de forma directa con el usuario y el administrador de actores, ver Figura 92.

Figura 92. Interfaz IActorCtrl



Fuente: Los Autores

16.3.3. Propiedades

- UsarMouse: Retorna o establece si se esta usando el mouse actualmente para generar los eventos del actor.

16.3.4. Métodos

- ProcesarEntrada: Realiza un llamado al método de captura de entrada de cada uno de los miembros de la colección de Actores que sea una instancia de tipo CSActorCtrl.

16.4. DEFINICIÓN DE CLASES BASE PARA ACTORES.

De acuerdo a los diferentes tipos de actores que se han creado cuatro clases base para la definición de actores.

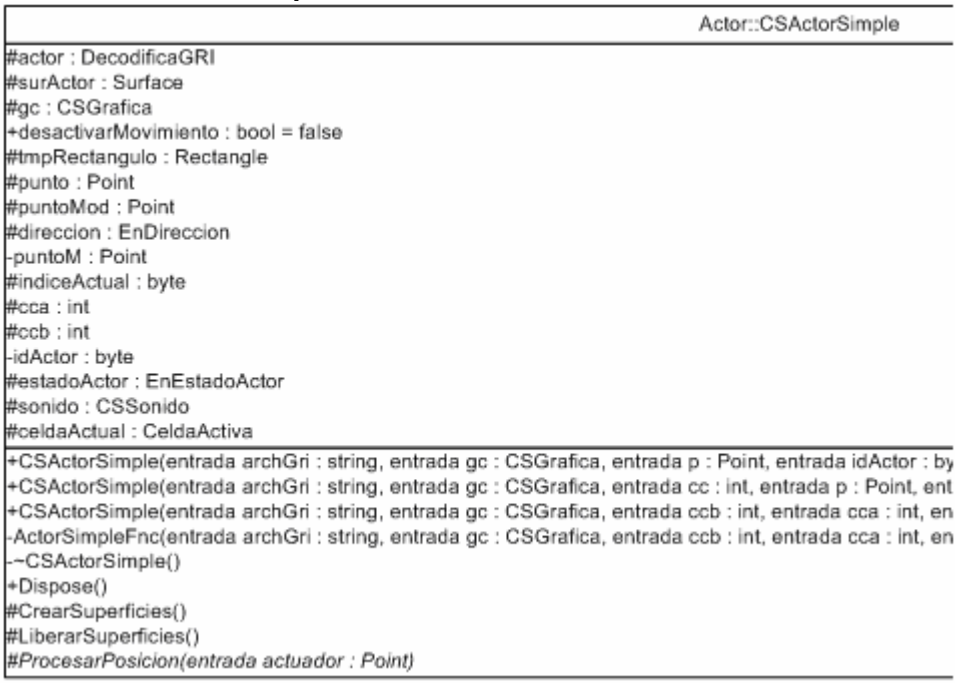
- CSActorSimple
- CSActorInv
- CSActorCtrlSimple
- CSActorCtrlInv

CSActorSimple y CSActorInv son clases que implementan las funcionalidades de la interfaz IActor, pero la única diferencia entre ellas es el manejo de recursos ya que la clase CSActorInv utiliza técnicas de reflexión de mapa de bits*.

CSActorCtrlSimple y CSActorCtrlInv poseen la misma diferencia que las dos anteriores pero cada una de ellas implementa la interfaz IactorCtrl.

Las estructuras que conforman estas clases se pueden ver en las figuras 93, 94, 95,96. Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

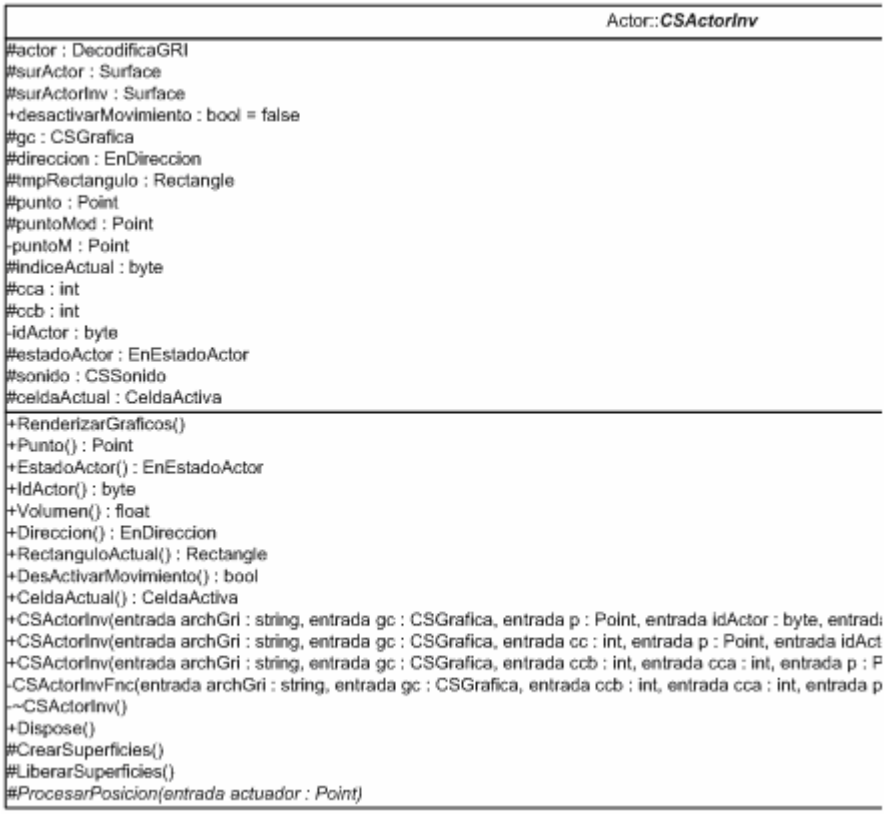
Figura 93. Clase CSActorSimple



Fuente: Los Autores

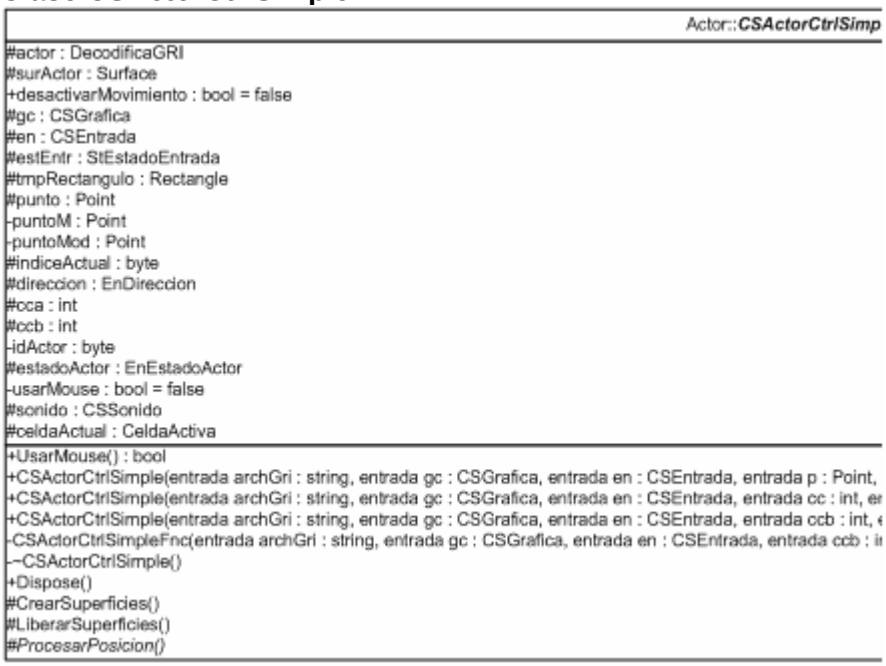
* Revisar Cáp. 16.1.3. Tipos de Actor del presente documento

Figura 94. Clase CSActorInv



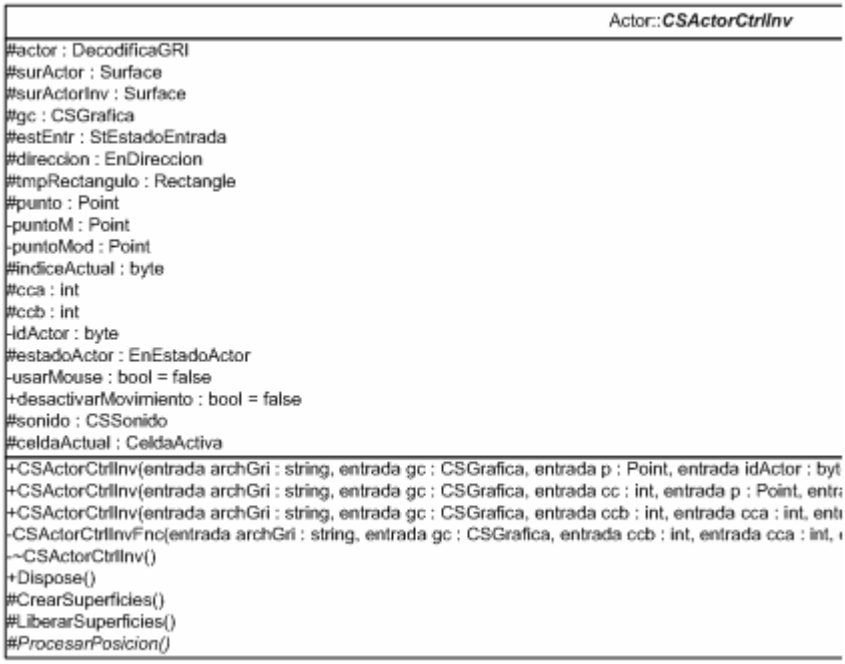
Fuente: Los Autores

Figura 95. Clase CSActorCtrlSimple



Fuente: Los Autores

Figura 96. Clase CSActorCtrlInv



Fuente: Los Autores

16.5. FUNCIONAMIENTO DEL MANEJADOR DE ACTORES.

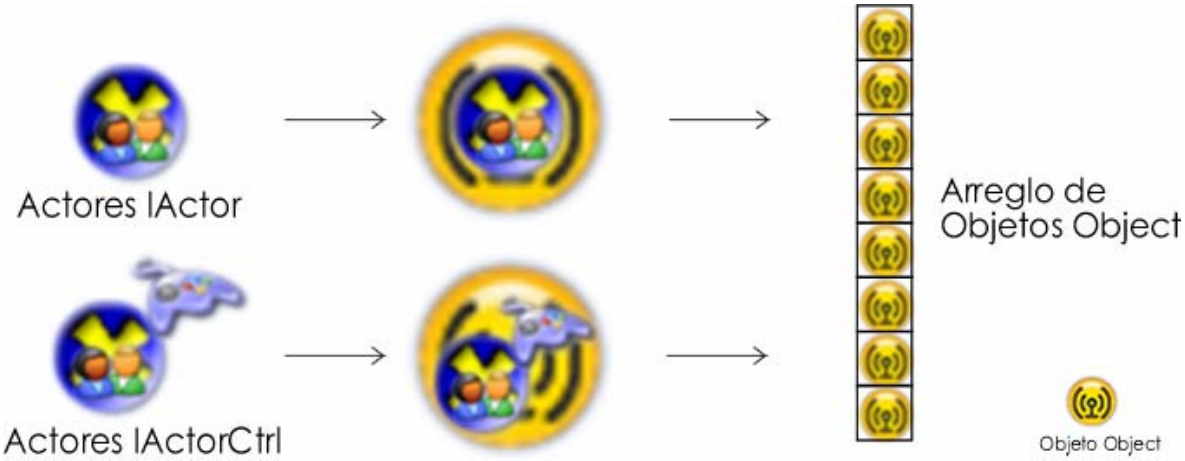
El manejador de actores posee un arreglo de objetos que implementan la interfaz IActor o bien IactorCtrl y administra de manera automática los llamados a los métodos de estos objetos, por esta razón el módulo incorpora métodos que permiten adicionar y retirar actores del administrador de actores, adicionalmente el administrador incorpora las mismas funcionalidades que se encuentran definidas en las interfaces IActor e IactorCtrl, pero sus implementaciones lo que hacen es realizar llamados a los actores que han sido incorporados y que cumplan con las exigencias de la interfaz.

16.5.1. Encajamiento y desencajamiento de objetos en el arreglo (boxing y unboxing). Debido a que el arreglo de actores que usa el administrador debe soportar múltiples objetos diferentes es necesario implementar o utilizar técnicas que permitan manejar ese conjunto de objetos de manera unificada, para tal fin se han creado las interfaces descritas en capítulos anteriores, ya que con ellas se simplifica en gran parte este problema y ahora solo es necesario pensar en dos objetos diferentes, los que implementan la interfaz IActor y los que implementan la interfaz IactorCtrl, de tal modo para agregar un actor cualquiera que sea su clase solo hace falce falta realizar un encajamiento en el arreglo del administrador de actores que por defecto es de tipo object [*] y para utilizar o tener acceso a los métodos del objeto solo es necesario hacer

[*] Se esta haciendo referencia a la clase object el lenguaje C#.NET

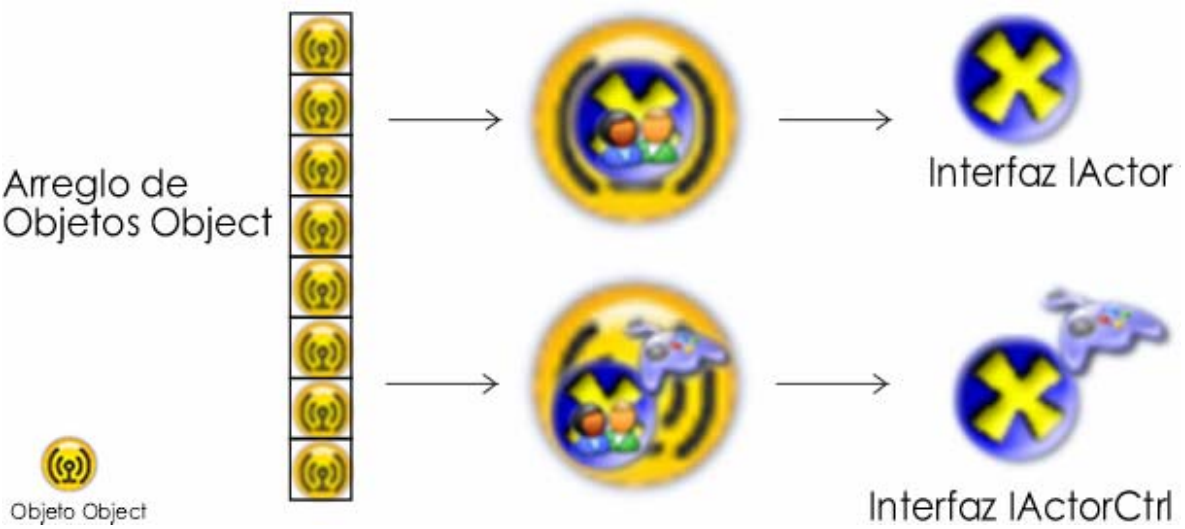
desencajamiento del mismo hacia un objeto que implemente IActor o IActorCtrl(figuras 97, 46).

Figura 97. Encajamiento (Boxing)



Fuente: Los Autores

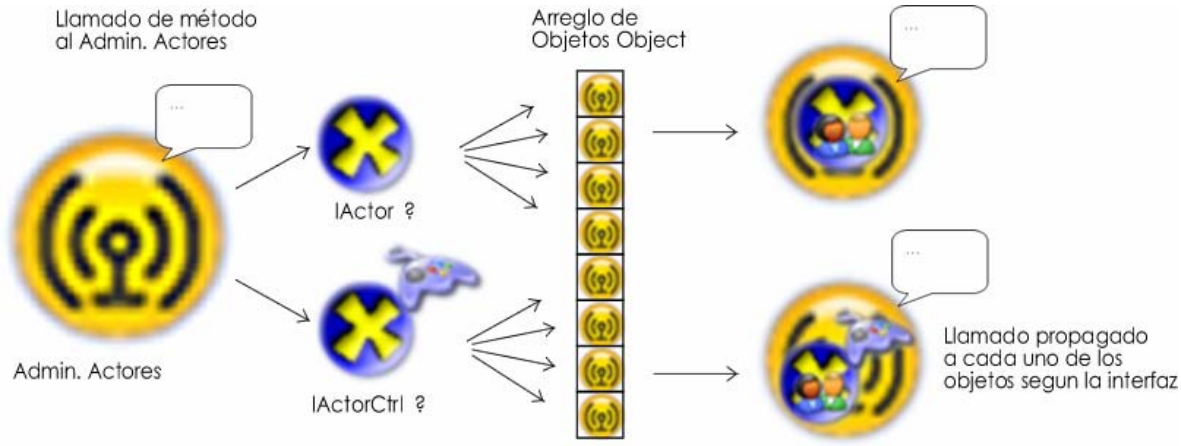
Figura 98. Desencajamiento (Unboxing)



Fuente: Los Autores

16.5.2. Propagación de llamados. El administrador de personajes posee algunas de las implementaciones o llamados de las interfaces IActor e IActorCtrl con el fin de que se use un llamado genérico a estas funciones, una vez que se realiza el llamado, el administrador de actores se encarga de propagar estas solicitudes de manera dinámica a cada uno de los objetos miembro del arreglo de objetos, el manejador diferencia entre las diferentes interfaces que implementa cada miembro del arreglo, para así realizar a cada uno solo las llamadas que le sean pertinentes de acuerdo al tipo de interfaz (Figura 99).

Figura 99. Propagación de llamados

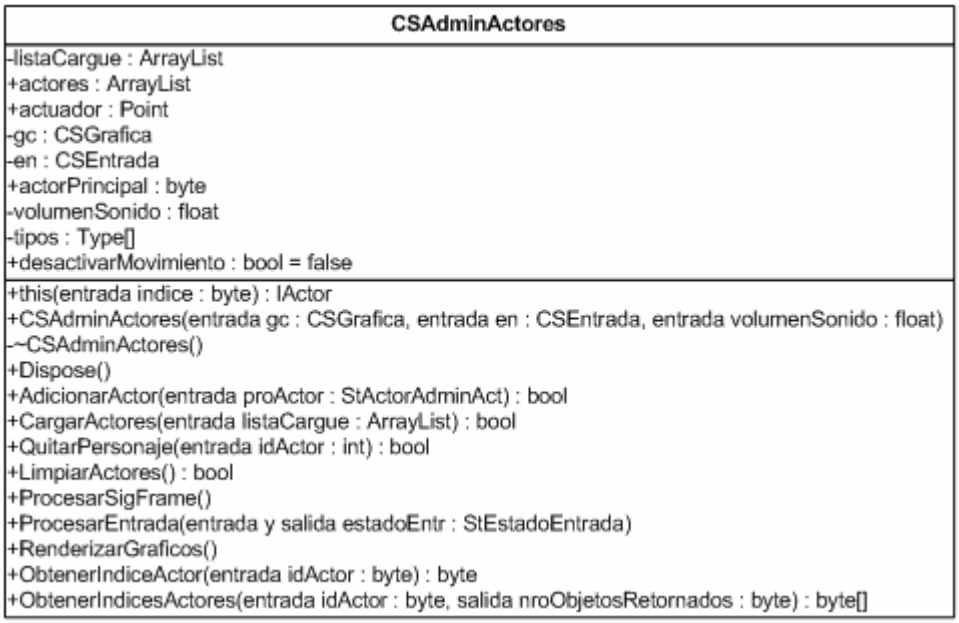


Fuente: Los Autores

16.6. DIAGRAMA DE CLASES

Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 100.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 100. Diagrama de clases del módulo manejador de actores



Fuente: Los Autores

17. MANEJADOR DE MENÚS

El módulo manejador de menús consta de una serie de clases que implementan los objetos de menú mas comúnmente usados dentro de un juego. Las diferentes implementaciones de objetos ofrecen funcionalidades comunes como:

- Botones
- Barras de desplazamiento
- Cajas de captura de texto
- Mensajes emergentes
- Implementaciones visuales de Mouse

Cada uno de los objetos enumerados debe trabajar en 'paralelo' con los demás y con el contenedor principal que es al que se le llama menú, por esta razón la implementación de un sistema que maneja este tipo de objetos se convierte en un sistema de comunicación por mensajes, es decir se debe desarrollar un módulo manejador de ventanas.

No se usan los componentes habituales como lo son los ActiveX, los MFC, ATL o los objetos de System.Windows del .NET Framework debido a que son incompatibles por la manera en que se dibujan en pantalla, ya que todos ellos hacen uso del propio sistema manejador de ventanas del sistema operativo el cual usa un motor gráfico GDI o GDI+ el cual no trabaja los dispositivos de la misma manera que DirectX, y aunque fuera compatibles, los sistemas GDI son demasiado lentos ya que cada ciclo de dibujo debe esperar al procesamiento de todos los mensajes de las ventanas en proceso y realiza el paso habitual a través de la API de Win32, DirectX y la mayoría de sus componentes se valen de una interfaz mas directa con el dispositivo evitando el llamado a las capas superiores de la API.

Al ser cada componente del manejador de menús un actor, estos hacen uso de las propiedades y objetos de la clase para desarrollar su comportamiento, así que ya cada uno posee métodos para dibujarse y para reproducir sonidos.

17.1. CONTENEDOR PRINCIPAL (MENÚ)

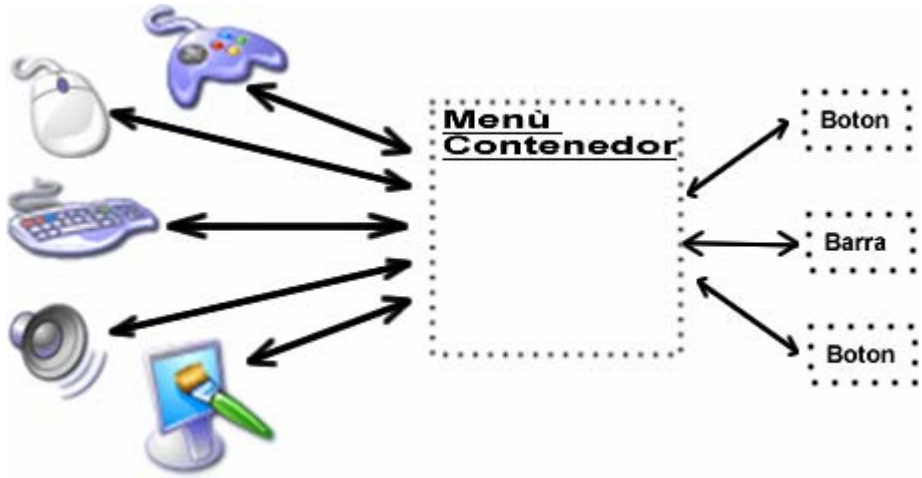
Un contenedor principal es un objeto que permite que le sean adicionados otros componentes propios de un sistema de ventanas, es en el donde se pueden agregar de manera dinámica y estática botones, barras, cajas de texto etc. Para lograr este efecto en el manejador de menús se recurrió a las funcionalidades que ya brinda el administrador de actores, entre las cuales se encuentra simplificar el uso de los elementos que poseen interfaces comunes derivadas de IActor y de IActorCtrl, este manejador de actores que trae incorporado el contenedor permite manejar el envió de mensajes entre los diferentes elementos que se carguen el menú. Cada vez que se adiciona un elemento a una pantalla

de menú, este elemento queda adicionado al manejador de actores y desde allí logra vincularse y sincronizarse con los diferentes eventos que ocurren dentro del contexto del menú* (Figura 101).

17.1.1. Navegación entre objetos. El contenedor principal posee una muy útil funcionalidad que es la de permitir la navegación entre los diferentes objetos a través del dispositivo del ratón o a través de la abstracción teclado - joystick**, todos los miembros del manejador de menús poseen o se integran con esta funcionalidad.

El manejador también permite habilitar o deshabilitar elementos del menú en un momento determinado, y se encarga de disparar o bloquear los eventos que se puedan generar a partir de un objeto bloqueado.

Figura 101. Propagación de mensajes en el sistema de menús



Fuente: Los Autores

17.1.2. Musicalización. El manejador de menús posee un reproductor de sonidos implementado haciendo uso de CSMusica (módulo del reproductor de música) y desde allí se administra la reproducción y efectos de desvanecimientos entre las diferentes pantallas de un menú o en las transiciones a otros estado de juego.

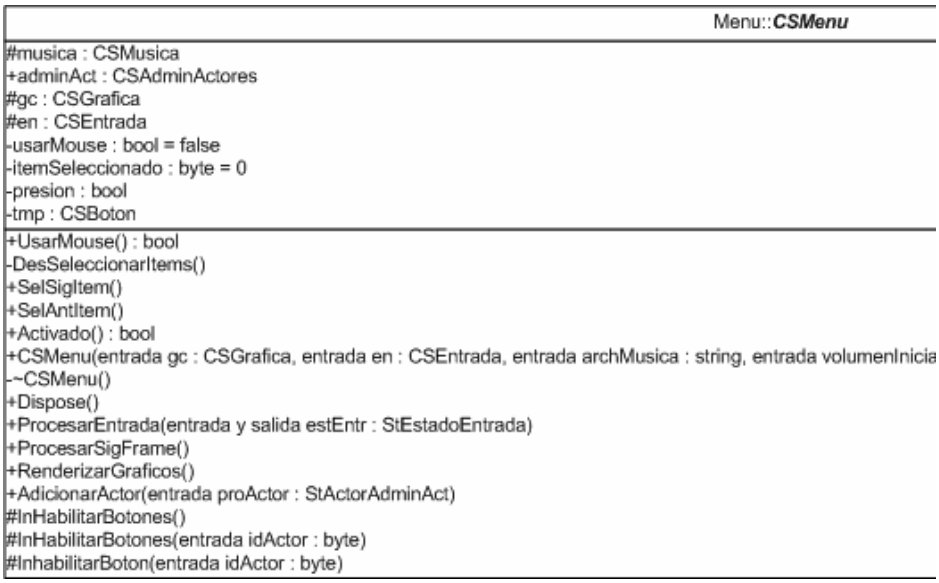
17.1.3. Diagrama de clases. Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 102.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

---

* Ver Cáp. 16.5.2. Propagación de llamados del presente documento

** Ver Cáp. 10.1. INTERFAZ UNIFICADA PARA TECLADO Y JOYSTICK del presente documento

Figura 102. Diagrama de clases del módulo manejador de menús



Fuente: Los Autores

17.2. IMPLEMENTACIÓN DEL MOUSE

Aunque el módulo de Entrada y Salida en conjunto con el sistema operativo proveen una serie de funcionalidades para acceder al dispositivo Mouse, estas funcionalidades no permiten su visualización en pantalla ya que se escapan del objetivo de dicho módulo, por esta razón las implementaciones donde se hace necesario que el Mouse sea visible requieren desarrollar estas funcionalidades.

El problema para la visualización del Mouse consiste en la virtualización de las coordenadas usada por el módulo gráfico para adecuarse a las necesidades de los diferentes modos de video[*], ya que ahora se suma un elemento adicional y es una nueva virtualización de coordenadas que se hace necesaria al momento de dibujar en modo ventana.

Esta implementación resulta muy conveniente ya que con solo implementarlo en el menú la funcionalidad se propaga a través del administrador de actores del menú a los demás miembros del mismo.

[*] Ver Cáp. 9.1.3. Modo pantalla completa y modo ventana. Del presente documento

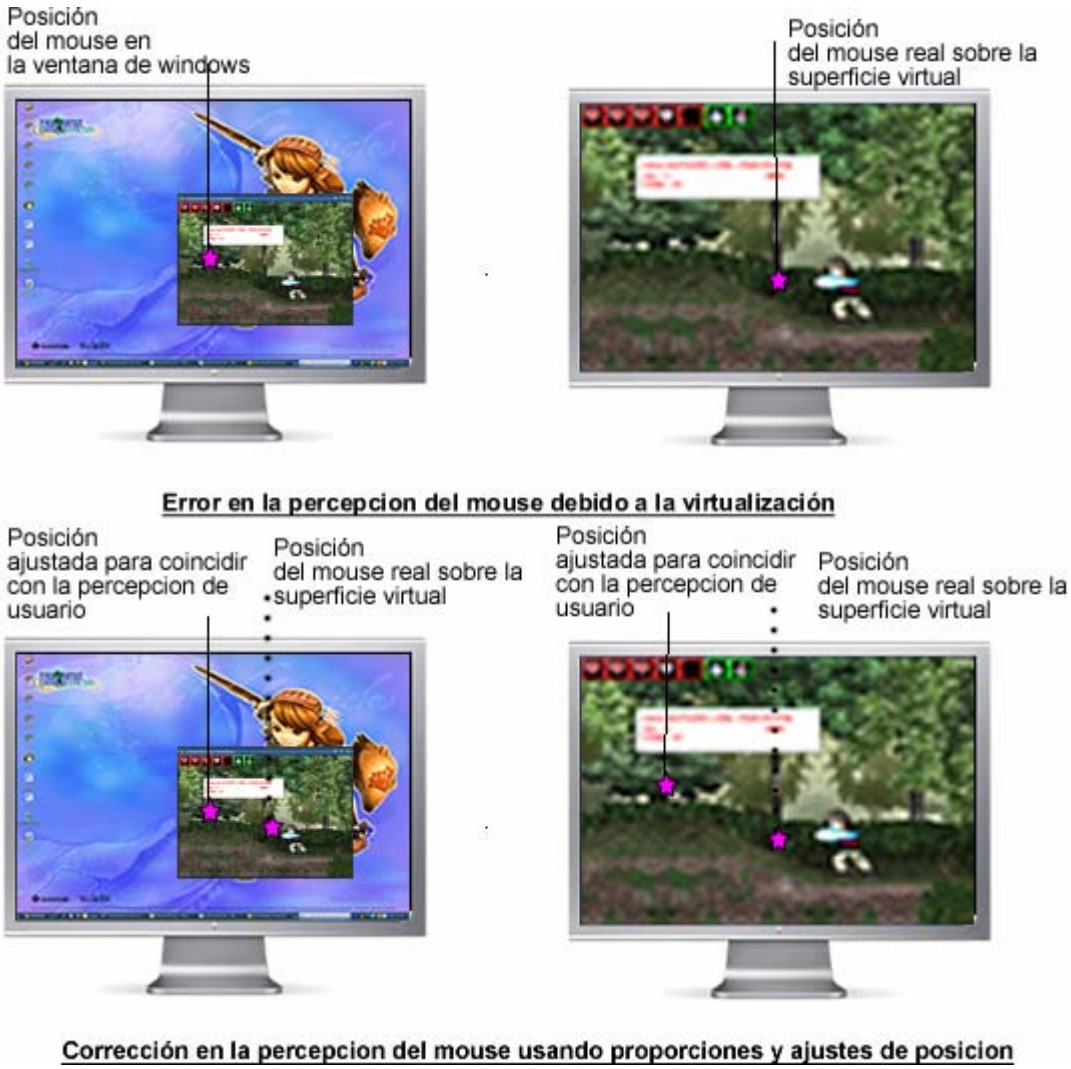
17.2.1. Cálculo de la ubicación virtual y real del Mouse. Dado que el modo ventana no tiene dimensiones constantes al momento de dibujar en la superficie principal y estas dimensiones son absolutas de acuerdo al tamaño de esta superficie (a diferencia de las demás superficies que sin importar el tamaño al que son dibujadas siempre poseen una escala virtual constante) resulta complicado dibujar el puntero del Mouse en la posición correcta, ya que para el usuario el Mouse debe moverse de acuerdo a como se mueve dentro de su sistema de ventanas tradicional, sin embargo la ventana de videojuego es una virtualización de escalas y de puntos por los cuales aunque el puntero teóricamente deba estar en una posición al realizar las operaciones de escalamiento y resposicionamiento este se dibujara en la posición que le correspondería en la pantalla virtual y no en la posición donde espera el usuario, peor aún el usuario creería estar activando un elemento del menú con ayuda el puntero del Mouse, cuando realmente las coordenadas del Mouse afectarían una posición diferente a la que el usuario ve para mayor claridad es recomendable observar, la Figura 103.

Figura 103. Error en la percepción del mouse

Fuente: Los Autores

Para solucionar este problema se hace necesario trabajar con un coeficiente de escala que es brindado por el módulo gráfico que se discrimina para cada una de las dimensiones de la ventana, este coeficiente es calculado en tiempo de ejecución por el motor y se halla a partir de la razón que surge de dividir el tamaño del buffer virtual entre el tamaño de la ventana en modo ventana (Figura 104).

Figura 104. Corrección de la percepción erada del mouse por virtualización



Fuente: Los Autores

17.2.2. Apariencia personalizable. Adicionalmente la clase Mouse permite personalizar la apariencia del puntero por cada instancia que sea generada ya que la clase Mouse hereda directamente de CSActorCtrlSimple.

17.2.3. Control de eventos del Mouse. Para que un objeto miembro del sistema de ventanas del manejador de menús sea notificado que el Mouse a ejercido alguna actividad en el, bien sea solo el moverse o el activar un botón del Mouse, la implementación requiere la funcionalidad de determinar si se encuentra sobre el área de dibujo del objeto que la ha invocado y debe ser capaz de informar si cuando estaba en el área de dibujo fue o no presionado uno de los botones del Mouse (Figura 105).

Figura 105. Fragmento corrección de la percepción del mouse

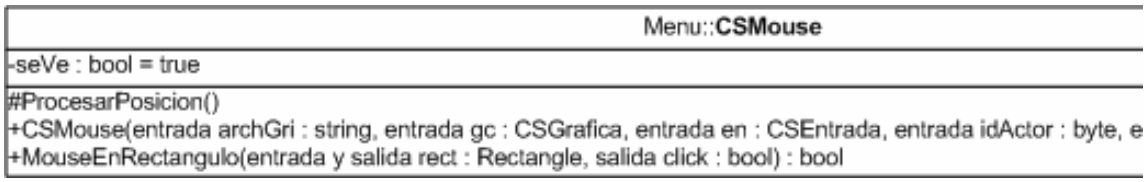
```
            punto = gc.forma.PointToClient(Control.MousePosition);
            punto.X = (int)(gc.Escala.Width * punto.X);
            punto.Y = (int)(gc.Escala.Height * punto.Y);

        Rectangle rectScr = gc.forma.RectangleToClient(rect);
        click =false;
        if((rectScr.X <= punto.X && rectScr.Y <= punto.Y) &&
            (rectScr.Right >= punto.X && rectScr.Bottom >= punto.Y))
        {
            click =(estEntr.Mouse_Button_R)? true:false;
            return true;
        }
        else
        {
            return false;
        }
```

Fuente: Los Autores

17.2.4. Diagrama de clases. Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 106.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 106. Diagrama de clases manejador de ratón en menús



Fuente: Los Autores

17.3. BOTÓN

El botón hereda directamente de la clase CSActorCtrlSimple[*] por lo cual comparte todas las funcionalidades de cualquier actor controlable y puede ser controlado por el administrador de actores contenido en el manejador de menús.

17.3.1. Estados de un botón. Un botón es un actor que alterna entre sus diferentes estados por acción del Mouse o de la abstracción teclado – joystick todas ellas filtradas y procesadas por el manejador de menús, de este modo un botón siempre tendrá al menos 2 estados:

- Presionado
- No presionado

Sin embargo de acuerdo a sus propiedades gráficas y a las facultades del manejador de menús es posible definirle a un botón estados como:

- Seleccionado
- Bloqueado

No todos los botones reaccionan igual ante cada estado que asuman, según las necesidades del desarrollo y las necesidades desde la parte artística. Así que existen tres tipos de botones (Figura 107):

- De dos estados: Botones que reflejan cambios gráficos al estar presionados o no presionados
- De tres estados: Botones que reflejan cambios gráficos al estar presionados, no presionados o seleccionados, se habla de un botón seleccionado cuando el Mouse esta sobre el, o cuando este asume el foco como consecuencia de una acción de la abstracción de teclado – joystick.
- De cuatro estados: Botones que reflejan cambios gráficos al estar presionados, no presionados, seleccionados o bloqueados, se habla de un botón bloqueado cuando el manejador de menús cambia su estado a bloqueado y así queda inhabilitado para responder a los eventos del Mouse o de la abstracción de teclado – joystick.

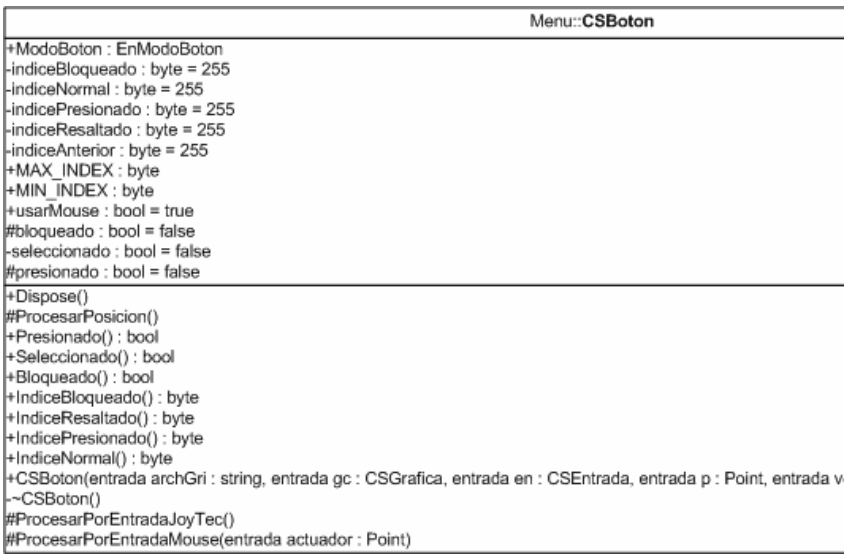
[*] Ver Cáp. 16.3. DEFINICIÓN DE INTERFACES PARA ACTORES. Del presente documento

Figura 107. Tipos de botón según el número de estados

Fuente: Los Autores

17.3.2. Diagrama de clases. Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 108.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 108. Diagrama de clases módulo Botón



Fuente: Los Autores

17.4. BOTÓN MOVIBLE

El botón movible hereda todas las características de la clase CSBoton por lo cual comparte todas sus funcionalidades puede ser controlado por el administrador de actores contenido en el manejador de menús.

17.4.1. Estados de un botón movible. Este tipo de botón maneja los mismos tipos de estado que un botón normal, pero sus métodos de administración de entrada y salida para alternar entre estados han sido extendidos para depender y variar de acuerdo a la posición o variación de la posición del actuador, que bien puede ser el Mouse o bien la abstracción de teclado – joystick (Figura 109).

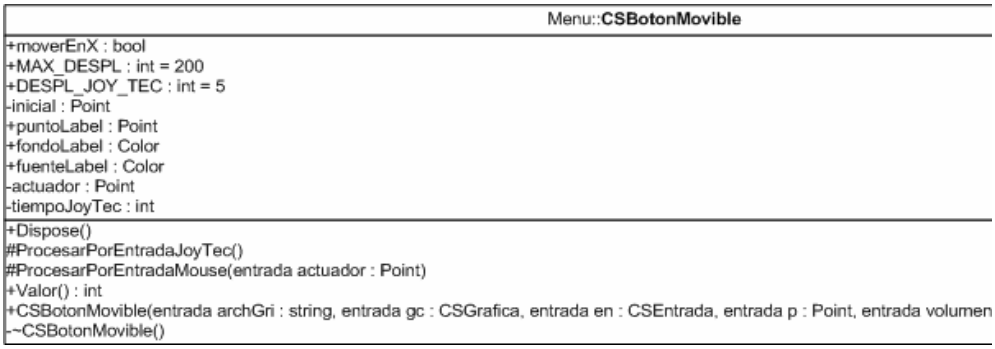
17.4.2. Posición del botón movible. El botón movible posee un pequeño aviso que hace las veces de indicador de posición (Figura 109).

Figura 109. Botón movible (barra de desplazamiento)

Fuente: Los Autores

17.4.3. Diagrama de clases. Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 110.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) en los capítulos 5 y 6.

Figura 110. Diagrama de clases CSBotonMovible



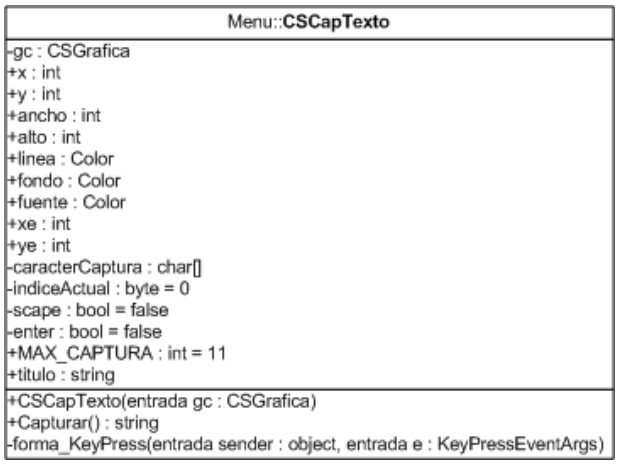
Fuente: Los Autores

17.5. CAJA DE TEXTO

La caja de texto es una clase que se dibuja totalmente haciendo uso de los métodos que provee el módulo gráfico, es decir no se deriva de un mapa de bytes sino de una serie de trazados en tiempo real. Esta implementación detiene el paso de mensajes dentro del sistema de menús en tanto no sea activada la acción desbloquearte del objeto, la cual sucede cuando se inserta un texto en la caja y se da este texto como aceptado (Presionar Enter).

17.5.1. Diagrama de clases. Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 111.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) en los capítulos 5 y 6.

Figura 111. Diagrama de clases CSCapTexto



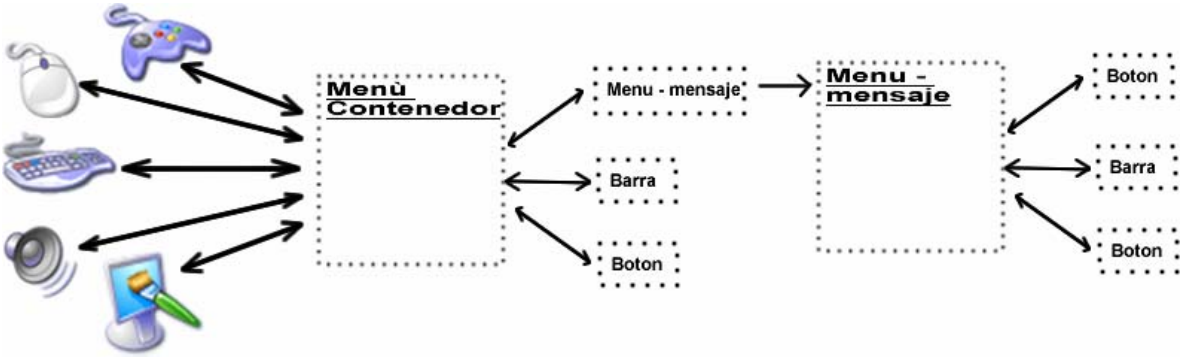
Fuente: Los Autores

17.6. VENTANA DE MENSAJES EMERGENTES

Este objeto se caracteriza por iniciar su propio proceso de paso de mensajes entre objetos y suspender la acción del menú que lo invoca hasta que no se obtiene una confirmación de leído que sucede al presionar el botón de salida del diálogo.

El objeto hereda de la clase CSMenu por lo cual todas las características propias de un menú son también implementadas por la ventana emergente, lo cual lo lleva a ser un contenedor de segundo nivel, que es creado también para poder almacenar objetos al igual que un menú* (Figura 112).

Figura 112. Mensaje funcionando como un menú anidado.

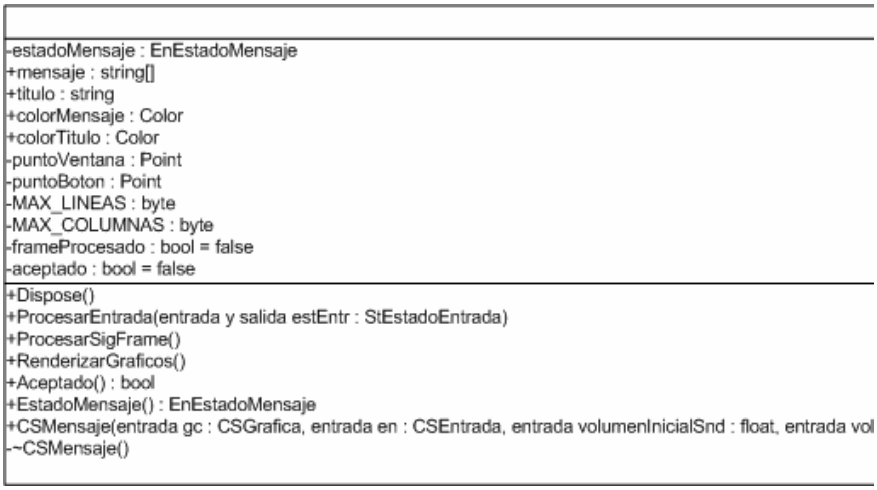


Fuente: Los Autores

17.6.1. Diagrama de clases. Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 113.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) en los capítulos 5 y 6.

* Ver Cáp. 17.1. CONTENEDOR PRINCIPAL (MENÚ) del presente documento

Figura 113. Diagrama de clases CSMensaje



Fuente: Los Autores.

17.6.2. Menús soportados bajo técnicas de tiles. Se utilizo la técnica de tiles (celdas) para construir interfaces en la que el usuario podría escoger una acción especifica. Las técnicas de tiles consisten en dividir la imagen mostrada en cuadros de menor tamaño y comenzar a dibujarlos en el orden correcto para lograr la imagen deseada.

Figura 114. Menú en Tiles

Fuente: Los Autores

En las pantallas de menús se utilizó esta técnica con el ánimo de evaluar la mejor opción para este tipo de interfaces, obteniendo como resultado que son algo restrictivas, por que al estar restringidas por la celdas no permiten gran manipulación, son mejores opciones las mencionadas anteriormente ya que permiten mas interactividad.

Las técnicas de tiles (celdas) son mas adecuadas en la generación de escenas para el mundo del un videojuego.

18. MANEJADOR DE ESCENAS

Este módulo es el que gestiona la operatividad de las escenas dentro del juego, es el administrador de acciones necesarias para el dibujado e interacción de las escenas dentro del videojuego.

Un videojuego de aventuras por lo general cuenta con una o varias escenas en las que el personaje debe ir avanzando para superar los retos que se le presentan. Las escenas son donde toma lugar el videojuego, en la descripción de guión se definen como van a estar compuestas y que retos tendrán, el manejador de escenas es la herramienta que permite que las escenas sean generadas y produzcan eventos que afecten al personaje principal y a los personajes secundarios.

Los requerimientos para un manejador de escenas por lo general serían los siguientes:

- Instanciar la escena
- Efectuar el cargue de los recursos necesarios de la escena
- Imágenes
- Sonidos
- Personajes
- Diálogos
- Mostrar un interfaz que indique el estado de cargue de los recursos
- Iniciar el dibujado de la escena
- Capturar los eventos generados en la escena para comunicarse con el manejador de personajes
- Terminar la escena y solicitar al motor la devolución de los recursos del sistema.

18.1. IMPLEMENTACIÓN DEL MANEJADOR DE ESCENAS

Para Fantasía Mitológica en el manejador de escenas se desarrollo las funcionalidades propias del mismo. Permitiéndole al juego poder controlar más de una escena y cumplir con las funcionalidades para poder interactuar con el manejador de personajes.

Para poder cumplir con los objetivos de un manejador de escenas, se construyó una clase en donde sus métodos invocaran los métodos de dibujado de las escenas y sus capas, también se desarrollo el menú de escenas en el cual el jugador elije la escena que desea jugar, en ese momento el manejador de escenas toma la referencia y la busca en sus escenas y inicia el cargue de los recursos vinculados por la escena. Anterior a esta acción se inicia la barra de cargue para indicarle al jugador la progresión en la lectura de los recursos, cuando a terminado el cargue de los recursos de la escena, comienza con el cargue de los personajes de la escena elegida, en este punto se comunica con el manejador de personajes y le indica cual es el personaje a ser instanciado y en que punto

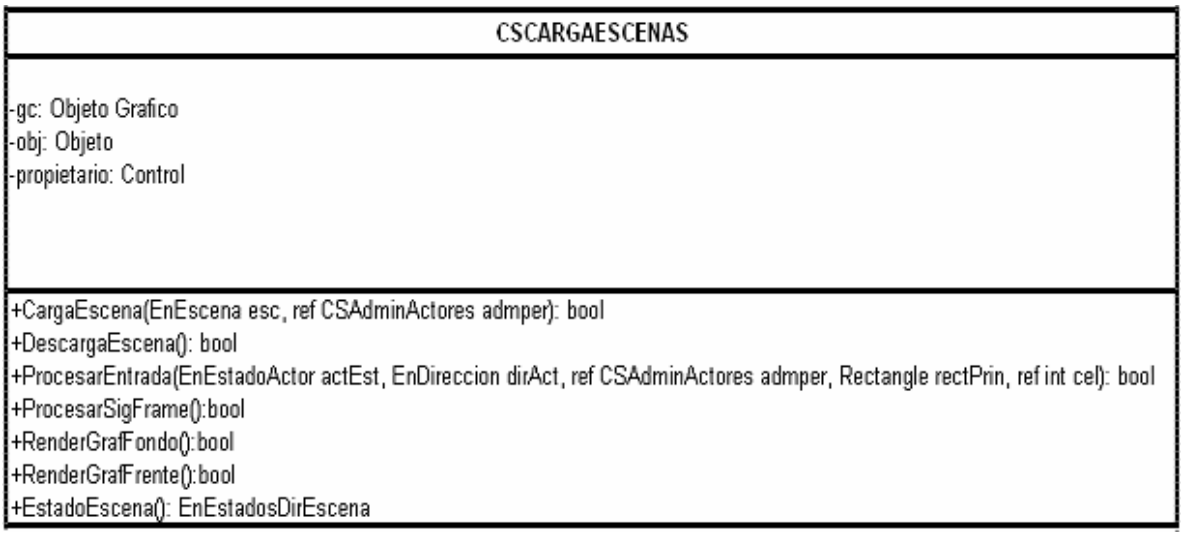
debe aparecer. Se dispone ahora a dibujar la escena y cada vez que el ciclo principal del juego le indica ordena recalcular las posiciones de la escena de acuerdo a la acción tomada por el personaje principal.

Figura 115. Pantallas de Escenas

Fuente: Los Autores

18.1.1. Diagrama de clases. Teniendo en cuenta las necesidades en un manejador de escenas, se ha diseñado el diagrama de clases de la Figura 116. Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A [documentación de desarrollo].

Figura 116. Diagrama de clases del módulo manejador de escenas

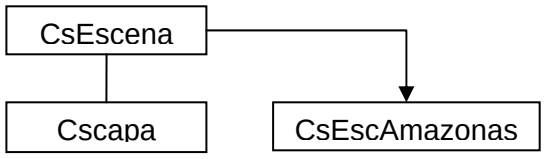


Fuente: Los Autores

18.2. ÁRBOL DE HERENCIA ESCENAS

Para el manejo de las escenas por parte videojuego se creo las siguientes definiciones de clases:

Figura 117. Diagrama de herencia de escenas



Fuente: Los Autores

18.3. IMPLEMENTACIÓN DE ESCENA

Las escenas en los videojuegos son el mundo donde se desarrolla la acción, son parte importante para poder colocar el juego dentro del contexto definido dentro del guión del juego.

Para los juegos de aventuras las escenas deben trasladar al jugador a un ambiente en donde se coloque los retos que deben superar.

Existen varias técnicas para poder construir un escena dentro de un videojuego, algunas de ellas utilizan un sola imagen en fondo que no cambia jamás y no tiene ningún movimiento, por otro lado existen los ambientes en donde el personaje es colocado en un mundo donde la escena tiene movimiento y da la sensación de estarse moviendo a través de ella, también existe un modelo en el cual se construye un escena capaz de interactuar con el jugador y dar un ambiente realista y encantador.

Los desarrolladores de videojuegos deben decidir cuán importante van a ser las escenas dentro de juego y entre mas estén vinculadas con el objetivo debe procurarse realizarlas lo mas detalladas e interactivas posibles. Pero esto tiene un inconveniente entre más detallado se ha el mundo a describir se necesitaran más recursos de la computadora. Es aquí donde se necesita una técnica que resuelva este inconveniente. La técnica de celdas o tiles ha sido ampliamente usada para esta misión.

18.3.1. Técnica de tiles. Esta técnica que es utilizada por algunos de juegos más famosos como Mario Bross, consiste en construir un escena a través de tiles (celdas), es decir, se divide la imagen que describe la escena en una cuadricula de tamaño pequeño y se realiza un proceso de búsqueda en donde se eliminan las celdas repetidas con el ánimo de hacer cada vez la imagen mas liviana y más fácil de manejar por parte de la computadora.

Figura 118. Escenas Mario Bross y Fantasía Mitológica Colombiana

Fuente: Los Autores

Aquí se ve un ejemplo de cómo se veía talvez el juego más famoso de todos los tiempos, Súper Mario Bros, al lado vemos una imagen extraída de el videojuego desarrollado en este proyecto Fantasía Mitológica Colombiana, en la imagen de Mario se puede ver claramente las división entre celdas y la repetición de las mismas sobre la escena, en la de fantasía mitológica no es posible apreciar esos detalles aunque están basadas en el mismo principio, algo adicional es que esta técnica puede ser aplicada en la modalidad de capas y con un buen manejo de las imágenes producir la sensación de realismo y profundidad. Veamos como es algorítmicamente la resolución de esta técnica.

18.3.1.1. El mapa. Si se presta un poco de atención, se puede notar que el mapa esta compuesto por pequeños bloquecitos cuadrados uno detrás de otro; se Necesita conseguir un método de mantener ese mapa en memoria y manejarlo a la hora de mostrarlo.

Una forma fácil es usar un arreglo de caracteres para guardar el mismo, a continuación se puede ver un mapa de ejemplo:

Figura 119. Mapa capa de tiles

```
Mapa[0] = "T                                                    EF                T"
Mapa[1] = "T                                                                      NL"
Mapa[2] = "T                  EF                  EF                      EF      T"
Mapa[3] = "T      EF                                                              T"
Mapa[4] = "T                                                            NO        NL"
Mapa[5] = "T           VWX                              VWX             NLU       T"
Mapa[6] = "T           YZa                              YZa   B         NLLU      T"
Mapa[7] = "T GHI        c                  NMO            c   ADC      NLLLLO     NL"
Mapa[8] = "TMMMMMO     NMMMMMMMO                      GHINMMMMMMMMMMMMMLLLLLLO     T"
Mapa[9] = "TLLLLLU     TLLLLLLLLO                     NMMLLLLLLLLLLLLLLLLLLLO      T"
Mapa[10]= "TLLLLLRPPPPPQLLLLLLLLLO    NMMO          NMMMMLLLLLLLLLLLLLLLLLLLLLLLO GHI T"
Mapa[11]= "TLLLLLeSSSSSSdLLLLLLLLU   NLLLRPPPPPPPPQLLLLLLLLLLLLLLLLLLLLLLLLLLLMMMMML"
Mapa[12]= "TLLLLLeSSSSSSdLLLLLLL     TLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL"
Mapa[13]= "TLLLLLeSSSSSSdLLLLLLLPPPPPQLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL"
Mapa[14]= "TLLLLLeSSSSSSdLLLLLLLSSSSSSdLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL"
Mapa[15]= "TLLLLLeSSSSSSdLLLLLLLSSSSSSdLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL"
```

Fuente: Los Autores

Cada carácter del arreglo, es un bloque en el mapa, así, dependiendo del carácter que sea se relacionará con el bloque y se usará una imagen asociada para representarlo. Por el momento hay tener en cuenta que un carácter vacío (“ “), es un espacio libre en el mapa.

18.3.1.2. Dibujando el mapa. Para dibujar el mapa se tiene que ir barriendo el arreglo, para detectar que caracteres hay, e ir dibujando según corresponda, para leer un carácter en la posición X e Y del mapa.

```
Char CH;
CH = encontrarCaracter[Mapa[Y],X,1]
```

Entonces, para dibujar el mapa, se barren los caracteres que se pueden ver por que solo la cámara activa es la que el jugador logrará ver en la pantalla.

```
for(Y=0;Y<=14;Y++) {
for(X=(X+ColMap); (X+ColMap)<= (20+ColMap); X++) {
Bloq = encontrarCaracter[Mapa[Y],X,1]
If(Bloq <> " ") {
posX = ((X) * 16) – (ColMap * 16)
posY = (Y * 16)
GraficarBloque(PosX, PosY, Bloq)
        }
    }
}
```

Donde:
En el ciclo anterior se comienza a graficar el mapa desde la posición 0,0 y se avanza hasta llegar al límite derecho donde se comienza con la siguiente línea, FilMap y ColMap son las coordenadas del mapa a partir de las cuales se empieza a leer el arreglo, serian algo así como la posición de la cámara en X e Y, GraficarBloque es una función que graficará al bloque, en la posición PosX, PosY de la pantalla, en este ejemplo los bloques tienen 16x16 píxeles de tamaño.

Es de notar el porque se resta (ColMap * 16)
Si estas restas no están, y la cámara esta en ColMap=40,
Entonces cuando grafiquemos el bloque situado en X=5 e Y=7 (ver el for), lo cual correspondería realmente al bloque X=ColMap+5 e Y=7 :

```
posX = [X] * 16]
posY = [Y] * 16]
```

Que valor tiene PosX y PosY ahora:
PosX = [ColMap+5] *16 = 720
PosY = Y *16 = 112

Estaríamos graficando el bloque en la posición [720,112] -en píxeles- en una resolución de 320x240. Evidentemente no veríamos nada, de todas formas, eso se aplica solo a la coordenada X, ya que en las Y solo tenemos la cantidad de bloques necesaria para llenar la pantalla.

La posición donde deberíamos graficar seria en realidad de

X=5*16=80
Y=7*16=122

Ese es el resultado que se obtiene con la resta.

En resumen, podríamos decir que la misma hace que los bloques que veamos, o lo que tenemos que ver, se centren en la pantalla.

18.3.1.3. Eligiendo la celda para dibujar el bloque. Antes una aclaración, se necesita un archivo donde están todos los sprites o celdas que componen la imagen, formando una grilla, este BMP se carga en una superficie directX y de allí se van sacando las imágenes usando un tipo rectángulo para seleccionar el sprite, el rectángulo es cuadrado (de 16x16), y lo que se hace es ir desplazándolo por la grilla, para sacar la celda que queramos.

Dijimos que dependiendo del carácter que haya en el arreglo, será el gráfico que tendrá el bloque, entonces, tendremos que variar la posición del RECT de origen, como se dijo antes, para sacar la imagen correspondiente.

Figura 120. Tiles

Fuente: Los Autores

Se va a ver como sacar una celda que necesitamos con el RECT la celda que se requiere es la imagen de un fragmento de los troncos.

```
RECT celda;
celda.Top = 1 + (17 * (2 – 1))
celda.Bottom = celda.Top + 16
celda.Left = 1 + (17 * (10 – 1))
celda.Right = celda.Left + 16
```

La fórmula se complica porque se tiene que saltar la línea divisoria entre las celdas, sin ella sería mucho más fácil, pero un poco más difícil para el diseñador, Ahora podemos hacer una formula general para extraer celdas de la grilla:

```
celda.Top = 1 + (17 * (Fila – 1))
celda.Left = 1 + (17 * (Columna – 1)
celda.Bottom = 16 + celda.Top
celda.Right = 16 + celda.Left
```

volviendo a lo de elegir la celda según el carácter, lo que se tiene que hacer es modificar el valor de Columna y Fila segur el carácter, y después modificar los valores del RECT.

Aquí hay una porción del código que hace esto:

```
Bloq = encontrarCaracter[Mapa[Y],X,1]
If (Bloq != " ")
If (Bloq = "A") { Fila = 2; Columna = 1}
If (Bloq = "B") {Then Fila = 2; Columna = 2}
..
 [y así la cantidad de veces necesarias]
```

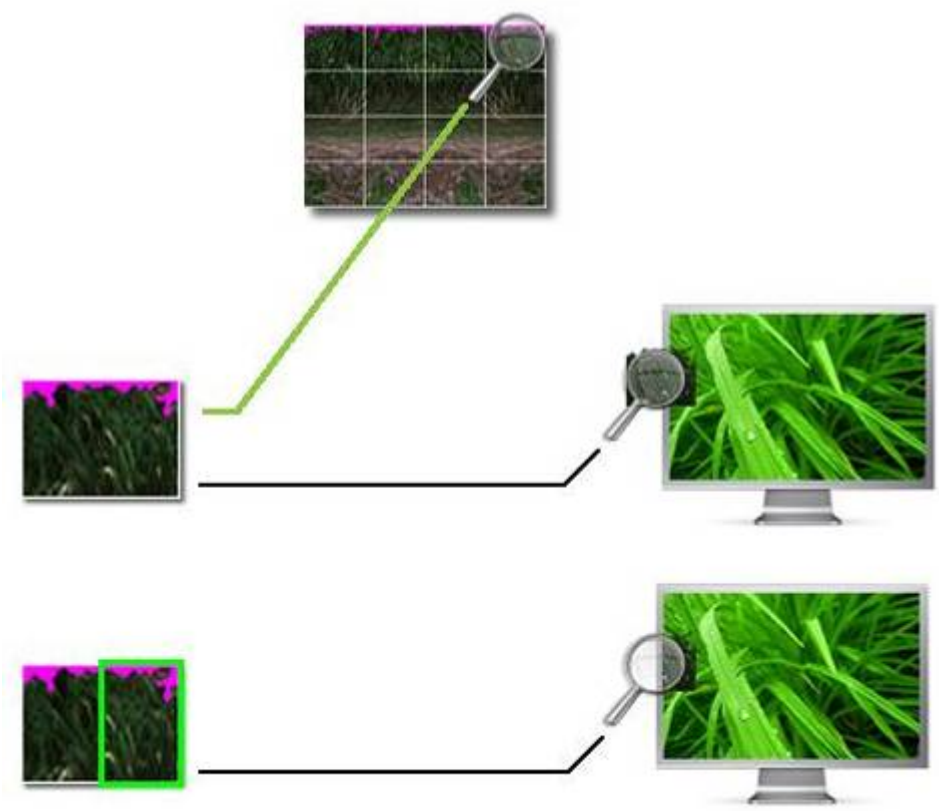
Ahora lo único que queda por hacer es ver como graficar el bloque en la pantalla en posX y posY las coordenadas en la pantalla

18.3.1.4. Clipping. Se denomina a la acción de eliminar algo que no puede ver, hay varias formas de hacer esto, se puede recortar el RECT destino, para que no se vaya fuera de los límites de la superficie, Entonces, lo que se hace es tomar un RECT mas pequeño, de modo que cuando lo copiemos a la pantalla no salga fuera de esta.
Un ejemplo gráfico de nuestro problema, y la solución son:

```
/*////////////////// clipping izquierdo //////////////////*/

If (posX <= 0) {
 celda.Left = celda.Left + (Abs(posX))
posX = 0
}
```

Figura 121. Clipping



Fuente: Los Autores

```
/*////////////////// clipping derecho /////////////////*/

If pos1X >= 319 {
 celdas.Right = celdas.Right – (pos1X – 319)
}
```

Donde:

pos1X es la coordenada X de la Derecha en la pantalla
posX es la coordenada X de la Izquierda en la pantalla

18.3.1.5. Parallax scrolling. El Scroll o movimiento que se tiene hasta ahora es muy brusco, ya que la cámara avanza por bloques (que serian 16 píxeles recordar ColMap), lo cual es extremadamente poco atractivo, se tiene que hacer que el Scroll sea de a 1 píxel, para lograr un movimiento suave y agradable, para esto, debemos modificar la sección de código donde se dibuja el mapa:

```
for(Y=0;Y<=14;Y++) {
for(X=(X+ColMap); (X+ColMap)<= (20+ColMap); X++) {
Bloq = encontrarCaracter[Mapa[Y],X,1]
If(Bloq <> " ") {
posX = [[X] * 16] - [ColMap * 16] – MapDspCol
posY = (Y * 16)
GraficarBloque(PosX, PosY, Bloq)
        }
    }
}
```

Que es igual que la anterior, solo que se resta MapDspCol (Mapa Desplazamiento Columna). Ahora, lo único que se tiene que hacer es manejar MapDspCol y ColMap de manera que se tenga un buen Smooth Scroll o movimiento suave.

ColMap es la posición de la cámara, en bloques, o sea que si se incrementa [o decrementa] su valor, se tiene un Scroll de a 16 píxeles [que es el tamaño de los bloques]. MapDspCol proporciona un desplazamiento del mapa en la pantalla, si se modifica este valor, se va a ver que todo el mapa se mueve de a píxeles, pero, no tiene la misma función que ColMap, MapDspCol puede ser visto como el control de posición horizontal del mapa en la pantalla [algo así como el potenciómetro del monitor, pero del mapa].

Ahora la implementación del Parallax Scroll: son dos piezas de código, una para cuando se detiene a la izquierda, y otra cuando se detiene a la derecha.

```
/*////////// Muevo la "cámara" a la derecha ////////////////*/
if (ColMap <= 58) {
MapDspCol = MapDspCol + 1
If (MapDspCol > 15) {
MapDspCol = 0
ColMap = ColMap + 1
}
}

/*////////// Muevo la "cámara" a la izquierda //////////////*/
If (ColMap > 1) {
MapDspCol = MapDspCol - 1
If (MapDspCol < 0) {
MapDspCol = 15
ColMap = ColMap - 1
}
}
```

Y con esto se logra un movimiento de la escena suave y armónico, se le pueden hacer mejoras a este ejemplo para que se mueva arriba y abajo o hacia las diagonales.

18.4. IMPLEMENTACIÓN DE CAPA

Figura 122. Escenas Fantasía Mitológica

Fuente: Los Autores

Una gran mejora visual de la técnica de tiles se logra cuando se aplica a varias capas en la misma escena, para ello será necesario solicitar al diseñador gráfico que construya las celdas para un paisaje en donde se aprecie la diferencia entre las capas.

Lo anterior significa que debe crear un ambiente en donde parezca que la capa que esta más en el fondo sea algo lejana y la que está más cercana al personaje se muy clara, todo esto combinado con un movimiento armónicamente adecuado producirá un efecto realista del ambiente que rodea el juego. Eso se puede apreciar en las imágenes de la figura 122:

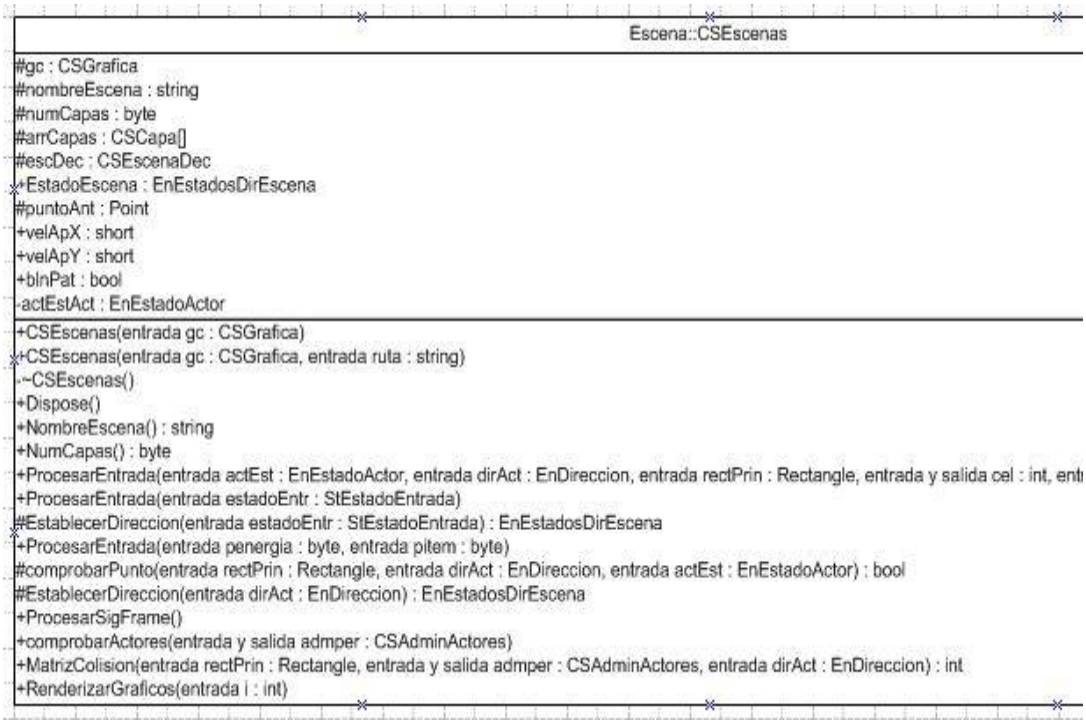
En ambas se utilizaron 4 capas para la escena cada cual tiene su propia velocidad para lograr el mismo efecto de cuando alguien mira a través de la ventana desde un auto en movimiento, los objetos que pasen muy cerca de el se moverán muy rápido en cambio los objetos que están a la distancia parecerán casi estáticos se moverán muy lentamente.

Para lograr esto es necesario utilizar la técnica de tiles con algunos cambios, es necesario poseer el mapa que describa a cada capa el cual debe ser cuidadosamente diseñado, para que las capas delanteras permitan que se vean las capas del fondo, además debe existir un arreglo de capas en el cual se debe iterar para dibujarlas en el orden preciso y lograr una escena realista.

18.5. DIAGRAMA DE CLASES

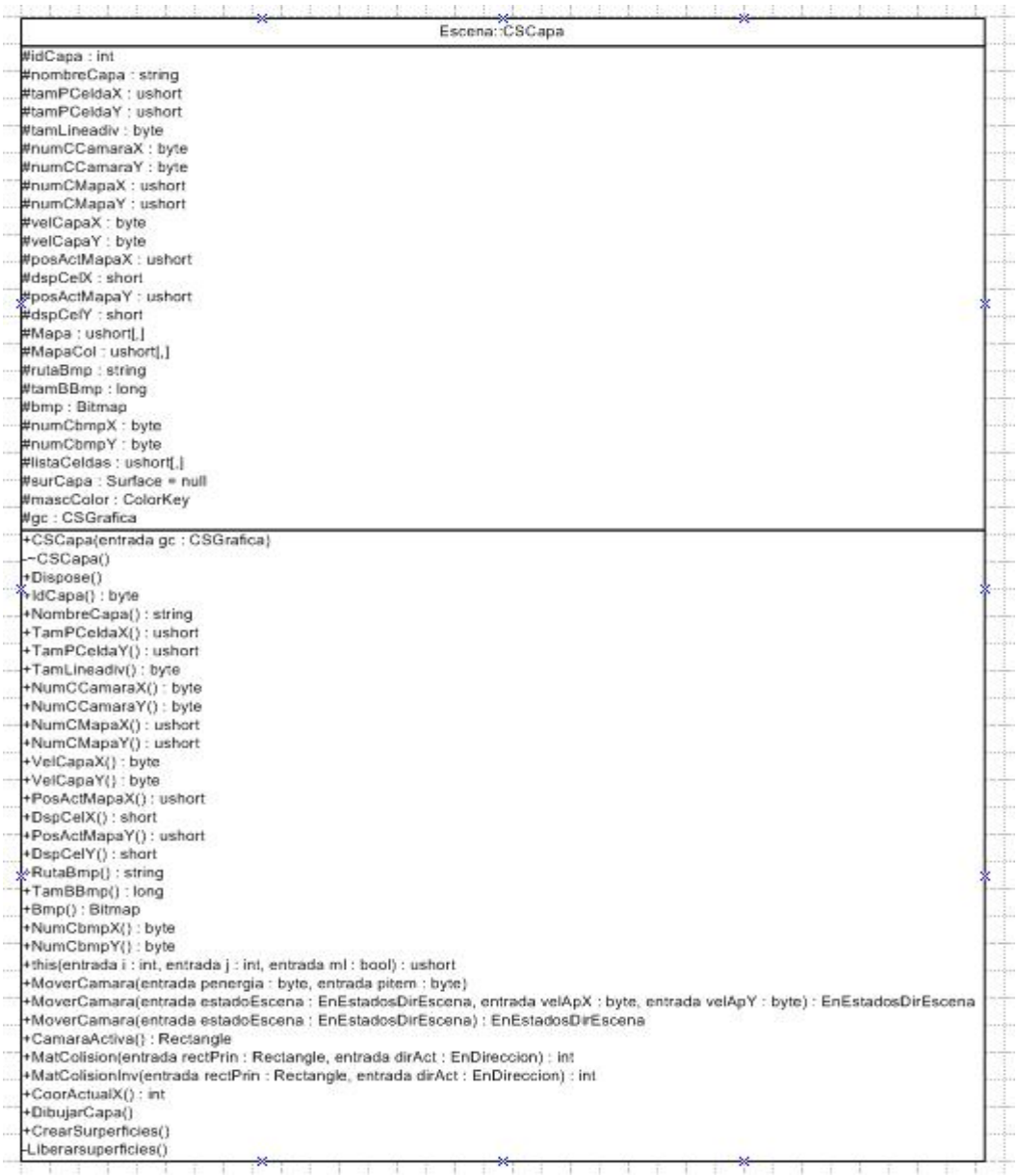
Teniendo en cuenta las necesidades de la escena y sus capas se ha diseñado el diagrama de clases de la Figura 123. Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A [documentación de desarrollo] y en los capítulos 5 y 6.

Figura 123. Diagrama de clases para escena



Fuente: Los Autores

Figura 124. Diagrama de clases para capa



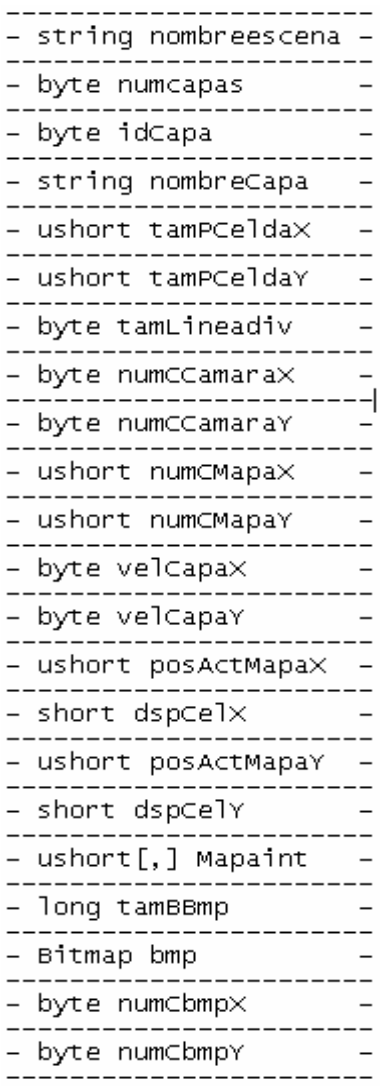
Fuente: Los Autores

18.6. FORMATO GRÁFICO GRE

En una escena que maneje capas y la cual sea dibujada en pantalla bajo la técnica de tiles(celdas), es necesario poseer toda la información requerida para aplicar el algoritmo, como el tamaño de las celdas, numero de celdas a lo largo y ancho de la escena, numero de capas, los mapas de cada una de las capas etc.

Para poder manejar esta cantidad de datos y con el ánimo de construir un escena reutilizable y parametrizada, se creo el formato en el cual se coloca la información referente a todos los detalles necesarios de una escena, la siguiente es la lista de conceptos que maneja el formato:

Figura 125. Especificación formato GRE



Fuente: Los Autores

- Nombre escena
- Numero de capas
- Lo siguiente se repite por cada capa dentro de la escena
- Identificador de la capa
- Nombre de la capa
- Tamaño en píxeles del ancho(X) de las Celdas
- Tamaño en píxeles del alto(Y) de las Celdas
- Tamaño línea divisoria entre las celdas en el bmp
- Numero de Celdas de la Cámara a lo ancho(X)
- Numero de Celdas de la Cámara a lo alto(Y)
- Numero de Celdas del Mapa a lo ancho(X)
- Numero de Celdas del Mapa a lo alto(Y)
- Velocidad de la capa en X
- Velocidad de la capa en Y
- Posición actual esquina superior izquierda en el mapa en X
- Desplazamiento de celda dentro de la cámara en X
- Posición actual esquina superior izquierda en el mapa en Y
- Desplazamiento de celda dentro de la cámara en Y
- Mapa de la capa, matriz de numCMapaX * numCMapaY
- Tamaño en bytes del bmp
- bmp de la capa
- Número celdas del bmp a lo ancho(X)
- Número celdas del bmp a lo alto(Y)

18.7. CODIFICADOR GRE

El formato GRE creado para poder manejar la información de la escena y sus capas, tiene un objetivo extra, el ocultar la información del archivo al jugador; es decir, que el jugador no pudiese entrar al archivo y cambiar los datos allí contenidos de manera simple. Además se debía tener una manera de generar los archivos de manera eficiente y más gráfica.

Para ello se diseño y construyo un subproyecto. El cual es un aplicativo especializado en crear archivos con la extensión gre y cumpliendo todos los requerimientos de la técnica de tiles.

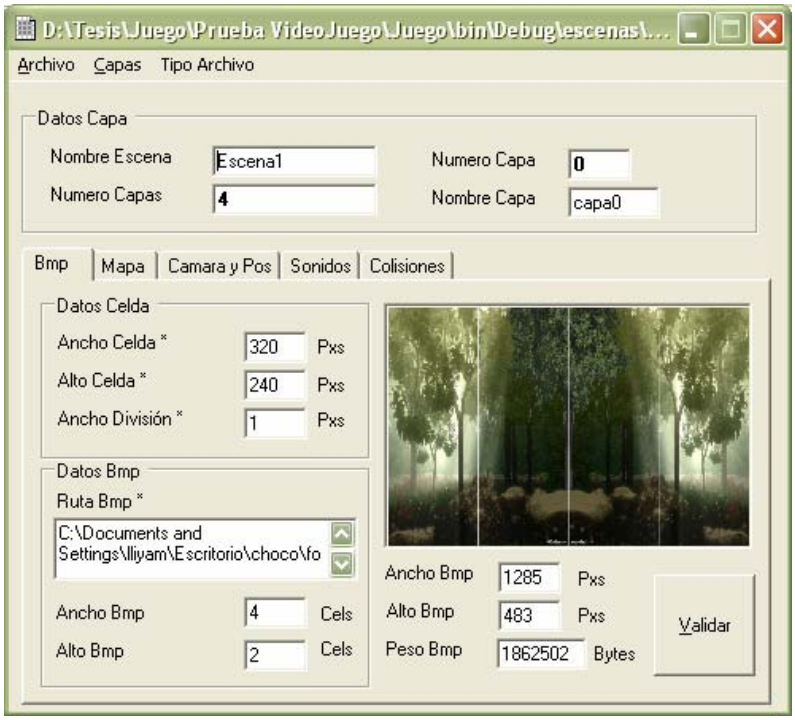
La primera versión del formato para una escena se encontraba embebida dentro del código del videojuego algo que no era muy genérico, ya que para cada escena se debía manejar sus propios atributos, de este punto se salto a la versión del formato en el que las cosas se manejaban con datos en texto plano y debían ser construidos en un editor; en donde cada línea en orden estricto era un atributo.

Con ello llegamos a un punto en donde se descubrió que dejar abierta esta información podría ser peligroso ya que el usuario podría cambiar las características de escena y

dañar su estructura. En este punto se inicio la codificación de los atributos de escena en forma binaria fue así como llegamos al formato gre.

El aplicativo codificador gre esta desarrollado sobre la plataforma .net en el lenguaje C# se ve de la siguiente forma:

Figura 126. Aplicativo para formato de escena

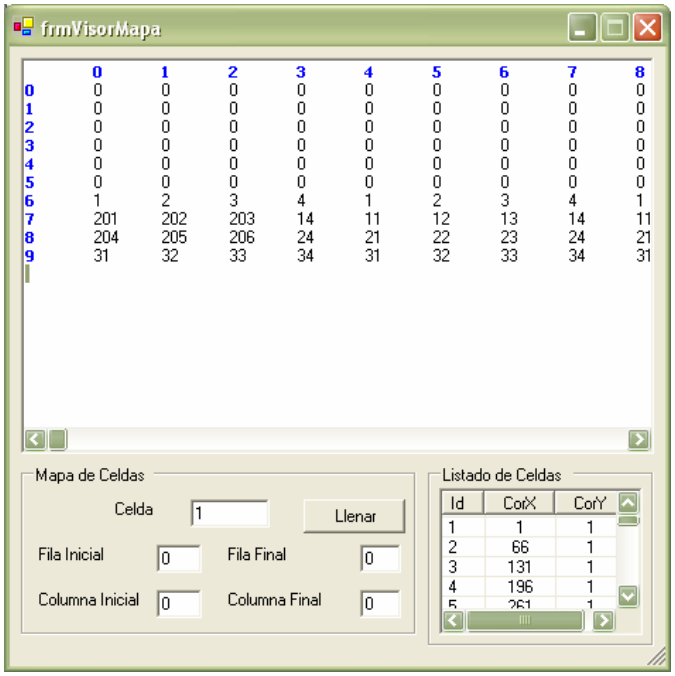


Fuente: Los Autores

En el están las funcionalidades necesaria para construir las escenas con capas.
Es de gran ayuda al construir un escena que tiene ya un tamaño considerable, por que para una escena promedio de Fantasía Mitológica Colombiana el mapa tiene una dimensión de 250 celdas de ancho por 8 de alto, eso no da un número de celdas igual a 2000 celdas que hay que llenar una por una.

Para tener mejor control sobre las celdas se diseño una vista en donde el diseñador de escenas tiene la posibilidad de ver el mapa y llenarlo al mismo tiempo.

Figura 127. Manejo de Mapa para Escena



Fuente: Los Autores

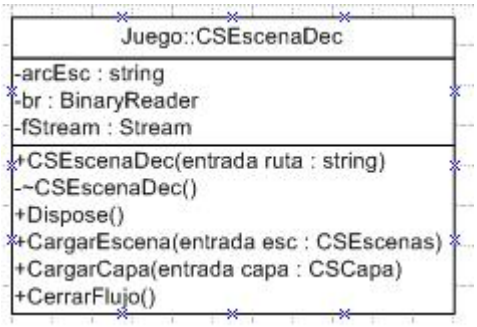
El aplicativo tiene las funcionalidades para adicionar o eliminar capas y para crear una versión tipo texto del archivo y así verificar que todo esta quedando como se ha diseñado.

18.8. DECODIFICADOR GRE

El formato GRE ya cumplía con sus requerimientos, ahora era necesario hacer que el motor de videojuego fuese capaz de entender el formato, se construyó un clase contenida dentro de moto para esta función,

La clase toma como inicio el dato del nombre de escena y el número de capas que contiene en ese momento comienza a iterar cargando los datos de cada capa.

Figura 128. Diagrama de clases para decodificador gre



Fuente: Los Autores

19. SISTEMA DE COLISIONES - PERSONAJE/ESCENA

19.1. COLISIONES EN 2D.

¿Qué sería de un videojuego sin colisiones?, es decir. ¿Si ningún objeto de los que actúa dentro del juego se enterara de que acaba de chocar contra algún enemigo, balón, árbol, proyectil, etc.?, seria un juego algo abstracto.
La base de la detección de colisiones es hacer que cuando los diferentes objetos que actúan en la pantalla se juntan lo suficiente entre sí parezca que están chocando y puedan reaccionar de alguna manera. La forma en la que van a reaccionar y cómo manejarlo depende mucho de qué objetos se traten y cómo sea el juego que se esté haciendo.
Los juegos en 2D tienen la característica de que la perspectiva desde la cual se desarrolla todo el juego siempre es desde arriba o desde un lado.

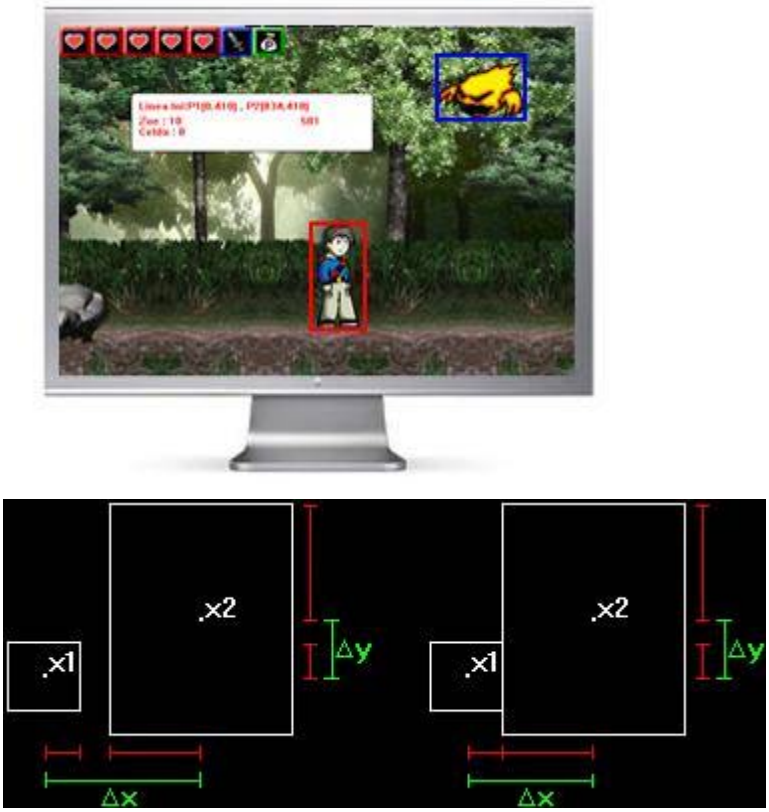
19.2. ESCENA-PERSONAJE

La relación entre la escena y el personaje principal es muy estrecha, ambos son afectados por las acciones y reacciones del uno frente al otro. La escena se mueve de acuerdo al avance del personaje y el personaje debe avanzar evitando los obstáculos que la escena le presenta, en cada momento se debe revisar esta relación, se debe revisar la posición del personaje principal en donde se encuentra parado y que acción esta realizando, si afecta a la escena o si por el contrario se afecta al personaje.
Un ejemplo clásico de detección de colisiones es cuando el personaje simplemente camina por el suelo de la escena, debe encontrarse en la posición del piso no podría estar caminado por encima de él porque parecería que estuviera flotando y no puede estar tampoco debajo de él porque parecería que esta por debajo del suelo, es aquí donde se entra a aplicar algo bastante interesante la física en los videojuegos.
La física en los videojuegos es un tema algo extenso y depende del juego mismo de donde se desarrolle y que capacidades u objetos se manejen, como ya lo hemos comentado para un videojuego se construye un mundo en el que hay que definir las reglas físicas que lo rigen y solo la imaginación del guionista pone límites a las reglas aplicables.
En un juego donde el personaje principal es un superhéroe que tiene la habilidad de volar, el no obedecerá a la ley de gravedad, pero en otro en donde el mundo esta basado en algo mas realista el personaje principal no debe sobrepasar los límites racionales de la física, en muchos juegos se acomodan estas reglas para hacerlo más jugable, pero algunos juegos han ganado renombre por simular fielmente la realidad.

19.3. DETECCIÓN DE COLISIONES

La manera de detectar las colisiones en un videojuego, varía de acuerdo a las reglas definidas en la física del mismo, pero en general se trata de calcular la posición actual del objeto y su área ocupada, esto para cada uno de los objetos después solicitar la escena que pasa en esa posición específica y tomar alguna decisión.

Figura 129. Manejo de colisiones



Fuente: Los Autores

La forma de encontrar la posición exacta dentro de la escena es basada en el mapa de celdas, se pregunta en dónde se esta dibujando el personaje y se devuelve las características de las celdas contiguas al personaje.

20. DISEÑO Y CONSTRUCCIÓN DE ELEMENTOS ARTÍSTICOS PARA FMC

20.1. PERSONAJES

20.1.1. Personaje Principal. Con base en la historia y en los bocetos se empieza a diseñar las secuencias de animación para el personaje principal, tales como caminar y/o correr, saltar, atacar, recibir daño y morir.

Para realizar cada animación del personaje principal se ejecutaron los siguientes pasos:

- Evaluar la cantidad de fotogramas necesarios para la secuencia de animación
- Realizar en papel la animación básica de el "esqueleto" del personaje (puntos guía unidos por líneas).
- Digitalizar la secuencia de animación del esqueleto,
- Convertir la secuencia en fotogramas por medio de Macromedia Fireworks.
- Probar la secuencia de fotogramas en Fireworks con intervalos de tiempo distintos para poder optimizar la animación.
- Detectar problemas en la secuencia de fotogramas y corregir el (los) fotograma(s) erróneos.
- Aprobar los fotogramas del esqueleto del personaje
- Redibujar en papel la secuencia de movimiento aplicándole texturas y volumen (rostros, cuerpo, ropaje, aditamentos, etc.) utilizando el esqueleto como guía de animación.
- Digitalizar las secuencias de animación.
- Redimensionar las secuencia de animación.
- Elaborar Fotogramas del Personaje utilizando las secuencias de animación.
- Probar Fotogramas en Fireworks.
- Detectar y corregir errores en la animación de los fotogramas.
- Agrupar los fotogramas en un solo archivo.
- Aplicar Colores en Photoshop CS utilizando técnica de capas y sombras con pincel en formato aerógrafo.
- Editar los bordes de las imágenes para eliminar colores con canal alpha.
- Delinear bordes de las imágenes con color negro.
- Exportar documento en archivo PNG y BMP.

Los gráficos del personaje principal se modificaron a lo largo del proyecto de acuerdo a las necesidades que exigía el engine en cuestión al formato de las imágenes.

Figura 130. Secuencia Ataque de Fuego

Fuente: Los Autores

20.1.2. Personajes Secundarios. Sus características tales como raza y vestido son determinados por la región donde se van a ubicar según la historia, además de esto la complejidad de movimientos es mucho menor, debido a que se reduce a 2 fotogramas en la mayoría de los casos. Estos personajes cumplen una función de informantes debido a que con la información que entregan a lo largo del juego el usuario podrá recibir pistas que le ayuden a avanzar dentro del juego.

Para realizar la animación de los personajes se siguieron los siguientes pasos, pertenecientes al desarrollo del personaje principal:

- Dibujar en papel la secuencia de movimiento aplicándole texturas y volumen (rostros, cuerpo, ropaje, aditamentos, etc.)
- Digitalizar las secuencias de animación
- Aplicar Colores en Photoshop CS utilizando técnica de capas y pincel en formato aerógrafo.
- Editar los bordes de las imágenes para eliminar colores con canal alpha.
- Delinear bordes de las imágenes con color negro.
- Agrupar los fotogramas con color en un solo documento en Fireworks.
- Exportar documento en archivo PNG y BMP.

Ver J. Anexo personajes secundarios

Figura 131. Personaje Secundario

Fuente: Los Autores

20.1.3. Enemigos Principales

Basados en los seres mitológicos más reconocidos de nuestro país, hacen su aparición dentro de la historia del juego, el Mohan, la Patasola, la Madre Monte, la Madre de Agua y el Bufeo.

Estos personajes tienen secuencias en común con el personaje principal (ataque, salto, daño, muerte) aunque en una proporción menor de fotogramas.

Para crear dichas secuencias se utilizan los mismos pasos descritos para las secuencias de animación y fotogramas del personaje principal. Ver Anexo K. Animaciones Enemigos.

Figura 132. Enemigos Principales

Fuente: Los Autores

20.1.4. Enemigos Secundarios

En esta clase de enemigos se encuentran seres mitológicos menores y animales. Para ambos tipos se utilizan secuencias de animación comunes (Movimiento, Ataque, daño y muerte).

Su aparición dentro del juego esta determinada al igual que los personajes secundarios por la región donde se ubican en la historia.

Otra característica común son los pasos a seguir en el desarrollo de secuencias de animación y fotogramas, salvo ciertas excepciones como los son el Jaguar y el duende, que necesitaron también de la creación de esqueletos las secuencias de animación más complejas. Ver Anexo L

Figura 132. Enemigos Secundarios

Fuente: Los Autores

20.2. ESCENARIOS

Para crear los escenarios, se decidió implementar dos técnicas para creación de escenarios, la técnica de tiles y la técnica de capas.

20.2.1. Técnica de Tiles. Esta técnica permite construir escenarios utilizando imágenes de distintas texturas que, poniéndolas de forma adyacente (unas al lado de otras), empalman a la perfección dando la ilusión óptica de que se esta viendo una sola imagen.

Los "tiles" permiten construir escenarios o mapas mediante la utilización inteligente de pequeñas gráficas que se identifican por sus diferentes texturas.

Con esto se puede generar una escena de grandes proporciones que simule un relieve con grama o roca, hasta texturas de agua y fuego, mediante la unión de "tiles" que tengan la misma textura. La utilización de "tiles" ha sido muy provechosa para la creación de escenarios en juegos ya que permiten a los programadores disponer de mapas fabricados a partir de matrices que toman diferentes valores dependiendo del "tile" que se representan en pantalla.

Esta técnica se empezó a utilizar en los sistemas de videojuego de 8 Bit, su funcionalidad era ahorrar todo el espacio en memoria posible debido a las limitantes existentes en el momento, este concepto se siguió implementando en los sistemas superiores de 16, 32 y 64 bit.

20.2.2. Técnica de Capas. Esta técnica se utiliza en los juegos 2D para simular profundidad y así dar un aspecto 3D. Consiste en definir varias capas con características diferentes y se acostumbra a utilizar entre 3 y 5 capas.

La velocidad de desplazamiento de las capas tanto en el eje X como en el eje Y depende de su nivel de profundidad, por ejemplo tenemos la capa mas lejana que es un paisaje que se denomina “Fondo”, donde se ubican elementos de paisajes tales como las montañas, ríos, valles así como bosques, el cielo, nubes etc., su movimiento debe ser el mas lento de todas las capas.

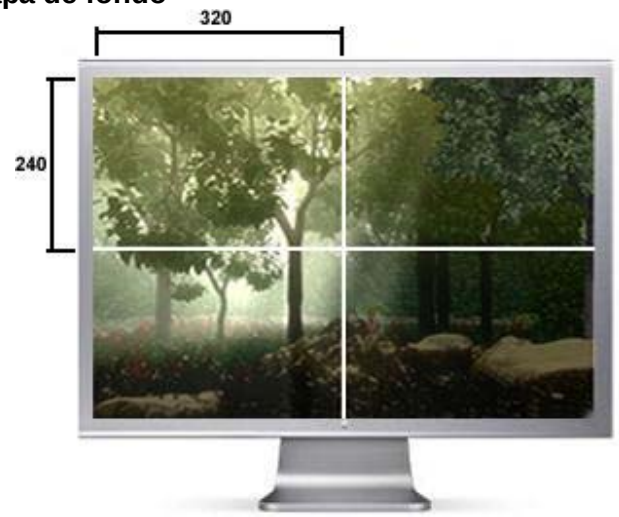
Retomando la información del tablero de historia, se procede a elaborar los tiles para los gráficos de cada capa.

20.2.2.1. Capa Fondo. Se utilizaron imágenes de paisajes representativos para cada escena, en el caso de amazonas y meta se optó por una imagen de un bosque denso y un atardecer respectivamente.

Se editaron los fondos utilizando Photoshop Cs y Fireworks para reflejar cada imagen y empalmarla con su reflejo, mediante técnicas de transparencia aplicada a los bordes de cada imagen se logro dicha fusión.

Posteriormente se crearon tiles de 320*240 para conformar una imagen de 1024 * 480.

Figura 133. Tiles capa de fondo



Fuente: Los Autores

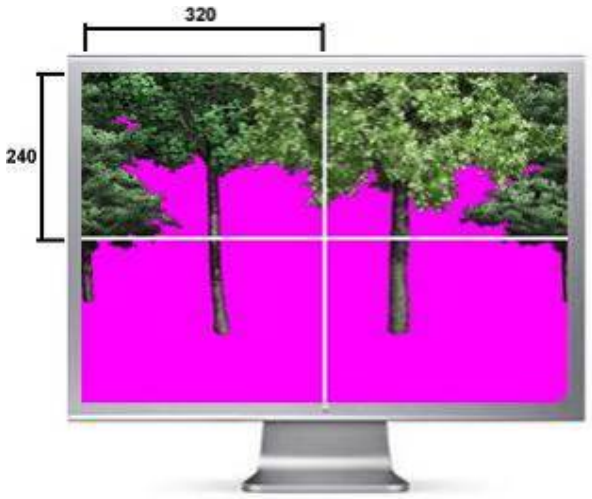
20.2.2.2. Capa Escenografía. En esta capa se incorporan elementos de escenografía tales como árboles, edificios, postes de luz, automóviles, buses, grupos de personajes no interactivos y relieve.

En la aplicación Maya se diseñaron distintos tipos de árbol para ser utilizados en esta capa, se ensamblaron todos en una sola imagen para conformar un bosque. Posteriormente se editaron los bordes de los árboles para eliminar colores con canal alpha.

Acabado esto las zonas vacías de la capa fueron determinadas utilizando un color de transparencia estándar Rojo: 255 Verde: 0 Azul 255 o #FF00FF en hexagesimal.

Utilizando la imagen del bosque se crearon tiles de tamaño 320 * 240 para conformar una imagen de 1024 * 480 que permita el desplazamiento de los objetos inmersos en la capa cuando el personaje principal avance o retroceda.

Figura 134. Tiles capa frontal



Fuente: Los Autores

20.2.2.3. Capa Relieve. Esta es la capa tiene una importancia más alta que las demás, debido a que es en ella donde se sitúan tanto el personaje principal como los personajes secundarios.

Contiene el camino que debe recorrer el personaje principal con sus respectivos accidentes geográficos como lo son pequeñas elevaciones, hoyos en el piso (vacíos, con agua, lava, púas, etc.).

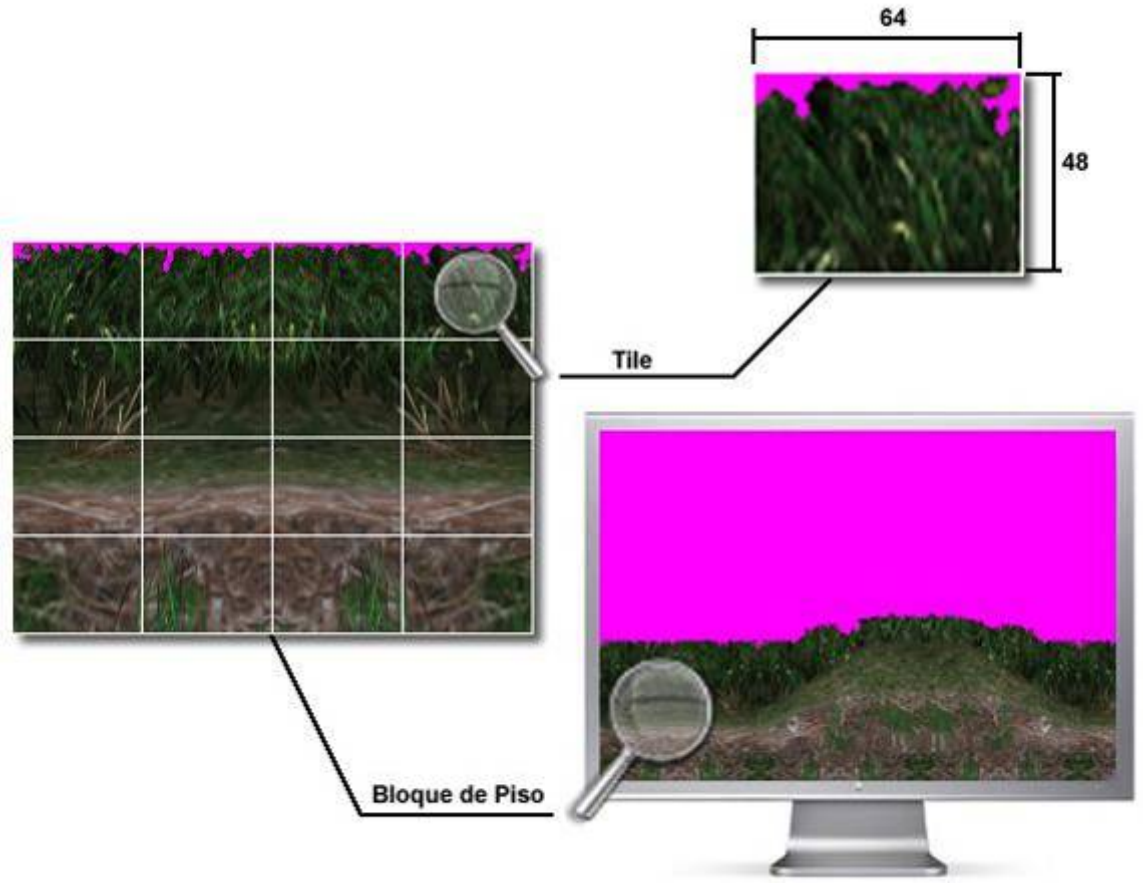
También en esta capa se incorporan elementos que interactúan con el personaje principal como trampas, objetos escenográficos (casas, tiendas, ítems, cajas, cofres).

Las dimensiones de cada tile de esta capa es 64 * 48 píxeles y las zonas vacías de la capa fueron determinadas utilizando un color de transparencia estándar Rojo: 255 Verde: 0 Azul 255 o #FF00FF en hexagesimal.

Los siguientes son los bloques de tiles que conforman la capa de relieve:

- Bloque de piso: Esta conformado por 16 tiles con texturas de tierra y de pequeñas ramas además de maleza alta. Posee 4 Tiles comodín que pueden ser reemplazados con tiles con objetos de púas.
- Bloque de Huecos: Con el mismo estilo de texturas de la sección de piso se generan los tiles de los hoyos. Se utilizan 24 tiles para elaborar un bloque, de los cuales 8 actúan de comodines ya que pueden se reemplazados con tiles cuyas texturas sean agua, púas y hasta fuego o lava.
- Elevación: Para diseñarla se hizo necesario crear 30 tiles de los cuales 5 se reutilizan para completar la estructura de la elevación.

Figura 135. Detalle de Tile Capa Relieve



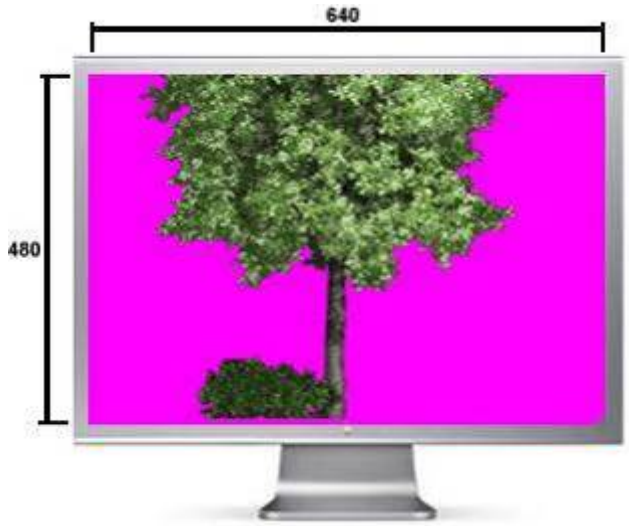
Fuente: Los Autores

- Bloque Cascada: Surge como una modificación del bloque de hueco, se creó una textura de caída de agua que luego se fusiono con el hueco, para así crear nuevos 16 tiles que pueden ser empalmados en el bloque de hueco y crear una cascada.
- Choza: Este objeto fue modelado y renderizado en Maya, se modela moviendo los vértices, se le aplican texturas hechas en photoshop y se renderiza con diferentes cámaras según la toma, luego se le aplican efectos de luces según el escenario.
- Su tamaño fue redimensionado para adaptarlo a la escala de la capa luego se fusionó con 2 bloques de piso y se dividió en 64 tiles.
- Roca, Troncos: Se fusionaron imágenes de roca y de troncos con el bloque del piso, se diseñaron 6 tiles para cada uno. Ver Anexo G Tiles Escenas

20.2.2.4. Capa Frontal. Estas capas están por delante de la capa de relieve, dependiendo de su función puede tener o no movimiento. Este tipo de capa posee varias formas de utilizarse:

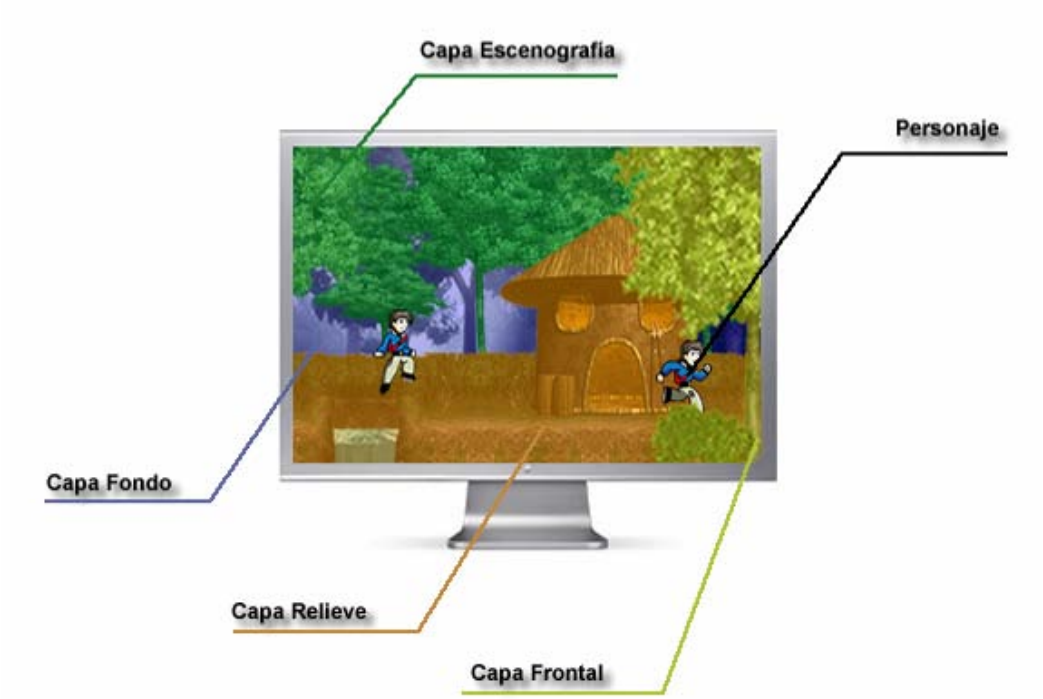
- Para ubicar árboles, maleza, postes de luz, o partes de la infraestructura del interior de una casa o tienda tales como sillas, mesas, materas, columnas etc.
- Situar en ella los ítems equipados por el usuario para que sean utilizados por personaje principal además de la barra de energía.
- Mostrar los diálogos entre los personajes, mensajes de introducción a escenas, mensajes de alertas, etc.

Figura 136. Capa frontal



Fuente: Los Autores

Figura 137. Ubicación de las Capas



Fuente: Los Autores

20.3. ELEMENTOS MULTIMEDIA

20.3.1. Videos

20.3.1.1. Presentación de Grupo. Se diseño un logo para representar al grupo de desarrollo del trabajo de grado, G.R. Media, por las siglas de los apellidos de los integrantes.

Se elaboro una animación en Flash utilizando el logo del grupo, una imagen de Zue (El personaje principal del juego) y un fondo musical.

Figura 138. Logo del grupo

Fuente: Los Autores

20.3.1.2. Presentación Del Juego. Utilizando el logo del título elaborado para el juego, los gráficos representativos de cada personaje e incorporando un fondo musical se realizó una presentación en flash para crear el video de presentación del juego.

Figura 139. Video inicial 1

Fuente: Los Autores

Conversión a Formato Avi. Macromedia Flash permite exportar las animaciones en un formato de video AVI para Windows o QuickTime para MacOS.

Los archivos fueron exportados a formato AVI con el Plugin XVID porque fue este el que otorgó los mejores resultados en calidad de audio y video.

20.3.1.3. Sonidos

Los sonidos utilizados en el juego se clasifican en distintas categorías:

- Animales: A lo largo del juego el personaje principal se encontrará con varios animales como el jaguar, monos, aves, mosquitos, perros, gatos etc. y cada uno de estos dispondrá de su sonido representativo.
- Efectos Especiales: Este tipo de sonido se utiliza en los ataques mágicos del personaje principal, los pasos y en el arma que utiliza.
- Escenográficos: Son los sonidos que dan ambientación a las escenas tales como cascada, fuente de agua, rayo, trueno, etc.
- Menús: En los diferentes menús pertenecientes a la interfaz gráfica del juego se utilizaron sonidos para eventos de los botones.
- Música: Se utilizaron distintos fondos musicales para cada menú, además de incorporar también música por cada escena.
- Personajes: Se utilizaron diversos sonidos para representar acciones específicas del personaje principal como lo son el salto, el ataque con el arma, recibir daño y caída.

La interfaz gráfica fue diseñada utilizando Photoshop CS para el marco con sombra alrededor de la imagen de fondo, los títulos y los botones con sus respectivos estados fueron diseñados en Fireworks , en esta herramienta se le incorporó el borde de madera a los botones y el efecto de relieve elevado y hundido a cada una de las etiquetas.

20.3.2. Menú Principal. Este es el Menú Inicial del juego, se carga cuando finalizan los videos de introducción del juego. Dispone de 3 botones, los 2 primeros permiten acceder al administrador de partidas y al menú de configuración, el último botón permite salir de la aplicación.

Figura 140. Menú Principal

Fuente: Los Autores

20.3.3. Menú de Configuración. Accediendo desde la segunda opción del menú principal se encuentra menú de opciones, esta divido en 2 secciones, la primera permite personalizar las siguientes características:

- Resolución en pantalla (640*480, 800*600, 1024* 768)
- Profundidad de color (16 bits, 24 bits, 32 bits)
- Volumen de los efectos de sonido (SFX) entre valores de 0 a 100.
- Volumen de la música entre valores de 0 a 100.

Los botones utilizados para las opciones de cambio de resolución y profundidad de color poseen un cuarto estado que es el estado seleccionado.

Utilizando la propiedad marco de madera en Firewoks se diseñó el botón utilizado para el control deslizante del volumen de la música y de SFX.

A las imágenes de accesorio (Monitores, parlante, teclado, Mouse, Joystick) y a las etiquetas presentes en cada una de las secciones, se les aplicó el efecto de sombra desde Fireworks.

.

Figura 141. Menú de Configuración: Sección 1

Fuente: Los Autores

La segunda sección permite configurar las siguientes opciones de modo de juego:

- Windowed (Modo ventana) o Modo FullScreen (Pantalla Completa).
- Dispositivo de Entrada, Teclado o Joystick
- Utilización del Mouse en los menús.

Todas las opciones personalizadas se guardan en el registro para que cuando el usuario ingrese de nuevo al juego no tenga que volver a configurar la aplicación.

Figura 142. Menú de Configuración: Sección 2

Fuente: Los Autores

20.3.4. Menú de Partidas. Este es el administrador de partidas, aquí se pueden crear o eliminar un máximo de 4 partidas.

El menú dispone de 7 botones de los cuales 4 están rotulados como “archivo” poseen un cuarto estado, dicho estado es bloqueo.

Cuando no se ha iniciado ninguna partida nueva, este será el estado por defecto de los botones.

Figura 143. Menú de Partidas

Fuente: Los Autores

20.3.5. Menú Selección de Escena. El diseño de este menú es distinto de los demás anteriores debido a que su implementación se hace por medio de la técnica de tiles y la técnica de capas.

Se diseño el menú de una forma integral (marco, fondo, botones) en una sola imagen, luego se procede a montar una cuadricula guía, para segmentar la imagen en tiles.

Los rollover de los botones se hacen fusionando la imagen del puntero (mariposa) con una copia de los tiles que contienen los botones. Esto con el fin de que cuando se este sobre un tile-botón dicho tile sea reemplazado en una capa superior por el tile con el efecto de rollover.

Figura 144. Menú Selección de Escenas

Fuente: Los Autores

20.3.6. Pantalla de Carga. Su diseño al igual que el selector de escenas se hace integral y luego se fracciona en tiles. El rectángulo central se hizo utilizando el efecto marco madera y al contenido de ese marco se le dio un efecto de transparencia al 40% en Fireworks .

Figura 145. Pantalla de Carga

Fuente: Los Autores

21. IMPLEMENTACIÓN DE ESCENAS PARA FANTASÍA MITOLÓGICA

21.1. ESCENA AMAZONAS

La escena de amazonas fue relacionada en guión del juego como la primera en la que el jugador se adentrara en la aventura, en ella Zue se encontrará con retos conocidos del ambiente selvático de la zona, el diseño de la escena es basado en su biodiversidad vegetal, animal y cultural.

En el transcurso de la escena Zue se encuentra con personajes típicos de selva como lo son habitantes de tribus y animales agresivos, los clásicos mosquitos que son constantemente molestos y el más grande depredador de la zona como lo es el jaguar al que se le atribuyen grandes misterios y se le tiene respeto por su voracidad.

21.1.1. Capas. Para el diseño de la escena se utilizaron 4 capas en donde se plaza el ambiente selvático de la zona, el verde y el color hojarasca se mezclan, las capas utilizadas fueron:

- Bosque Profundo
- Árboles
- Caminos y Poblados Indio
- Árboles

21.1.2. Obstáculos. Los obstáculos presentes en la escena son los siguientes:

- Chuzos
- Abismos
- Trampas
- Ríos
- Cascadas

21.1.3. Características Típicas. Las características típicas que se podrán apreciar son:

- Habitantes
- Caserío Indio
- Leyenda del Bufeo
- Zona Selvática

21.2. ESCENA META

La escena del meta es donde Zue comenzará a descubrir pistas sobre el misterio de la desaparición de su madre, esta zona colombiana en la que se puede encontrar grandes planicies combinadas con una topografía con grandes mesetas y los atardeceres en donde el sol parece no quisiera ir a descansar, en ella el jugador avanzará por un abrazante sol y la fauna local.

21.2.1. Capas. Para el diseño de la escena se utilizaron 4 capas en donde se plaza el ambiente selvático de la zona, los colores marrones y el color hojarasca seca se mezclan, las capas utilizadas fueron:

- Atardecer
- bosque
- Caminos y Poblados Rural
- Árboles

21.2.2. Obstáculos. Los obstáculos presentes en la escena son los siguientes:

- Chuzos
- Abismos
- Trampas
- Lagunas

21.2.3. Características Típicas. Las características típicas que se podrán apreciar son:

- Habitantes
- Caserío Rural
- Leyenda de la patasola
- Zona Selvática

22. IMPLEMENTACIÓN DE ELEMENTOS COMPLEMENTARIOS PARA FMC

Los elementos complementarios de Fantasía Mitológica Colombiana son los que hacen del juego más agradable, estos elementos son los siguientes:

- Menús
- Videos
- Selector de Escenas
- Barra de Estado del Personaje
- Barra de Cargue

Vamos a describir cada uno de ellos, para ver su objetivo y la forma en la que fueron realizados.

22.1. SELECTOR DE ESCENAS

Un selector de escenas para un videojuego es el menú de destinos al que el jugador puede ingresar, en algunos es un mapa el cual se sigue en un estricto orden y muestra un camino a seguir en otros casos no el jugador no tiene interacción con esta pantalla en juego solo la muestra para que el jugador sepa en que nivel de avance se encuentra.
Fantasía Mitológica colombiana se desarrolla en Colombia, nuestro país, es por ello que el selector de escenas para el juego, muestra el mapa de Colombia y las zonas elegidas en guión por ser tener más historias de mitología.

Figura 146. Selector de Escenas

Fuente: Los Autores

Para el desarrollo se utilizó la misma técnica que para el desarrollo de las escenas, con algunas diferencias, como evitar el movimiento y solo cuenta con 2 capas, una en donde se coloca el fondo y la otra en la cual se mueve la mariposa la cual sirve como apuntador.

22.2. INTERACCIÓN BARRA DE ESTADO/PERSONAJE

La barra de estado es un elemento que le da realito al juego; le permite al jugador saber cual es su condición actual y varios datos de interés como que arma activa, que magia o objeto esta activo y en muchos juegos se coloca el puntaje que va acumulando o la cantidad de dinero que a conseguido, dependiendo el juego la barra de estado debe contener los datos relevantes que le permitan conocer el estado del juego al personaje.

Para Fantasía Mitológica colombiana se diseñó una barra de estado en la que el jugador podrá conocer número de corazones o vida que tiene, el arma que esta utilizando y el objeto que tiene en este momento activo, la barra luce de la siguiente manera:

Figura 147. Barra de Ítems

Fuente: Los Autores

La ubicación de la barra depende del diseño de las escenas, esta debe ser ubicada en donde no interfiera con el desarrollo de la acción, y su tamaño debe ser también importante.

Como ya lo comentamos la barra maneja 3 aspectos cada uno tiene relación con el personaje principal:

Nivel de Vida
La vida para Zue es un numero limitado, cada vez que sea atacado perderá a razón de medio corazón por golpe, al llegar a cero morirá.

Arma Actual
El arma actual es como su nombre lo dice el arma con la que atacara, las opciones que tiene Zue son machete, garra, poder, y fuego. Y tendrá la habilidad a través del joystick o interfaz de comando de irlas cambiando una a una.

Objeto Actual
Este es el objeto que utilizará Zue en determinadas ocasiones, cuando ingresa a una tienda podría utilizar el dinero, o en alguna clave o secreto puede utilizar un gema u objeto maravilloso.

Para el desarrollo se utilizó la misma técnica que para el desarrollo de las escenas, con algunas diferencias, solo utiliza una capa y en ella cada objeto tiene las mismas dimensiones.

22.3. BARRA DE CARGUE

La barra de cargue es un objeto que tiene como misión mostrar el avance del cargue de recursos de escena y que el jugador no vea un imagen bloqueada del selector de escenas, este un elemento se introdujo cuando los juegos comenzaron a tener grandes necesidades de recursos.

Su apariencia es la siguiente:

Figura 148. Barra de Cargue

Fuente: Los Autores

Para el desarrollo se utilizó la misma técnica que para el desarrollo de las escenas, con algunas diferencias, utiliza dos capas la de fondo y la capa de bloques de avance.

22.4. BUCLE PRINCIPAL

El bucle principal en un videojuego es la parte donde se desarrolla toda la acción. La necesidad de tener un ciclo que se repita continuamente es por que en un juego no se puede esperar hasta que el usuario de una orden para generar una respuesta.

Como se explicó en la sección 3.4.6. Ciclo básico en un videojuego, un videojuego tiene un ciclo básico que se cumple casi para todos los juegos:

- 1: Inicialización
- 2: Ciclo de Juego
- + Entrada
- +Procesamiento
- +Salida
- 3: Finalización

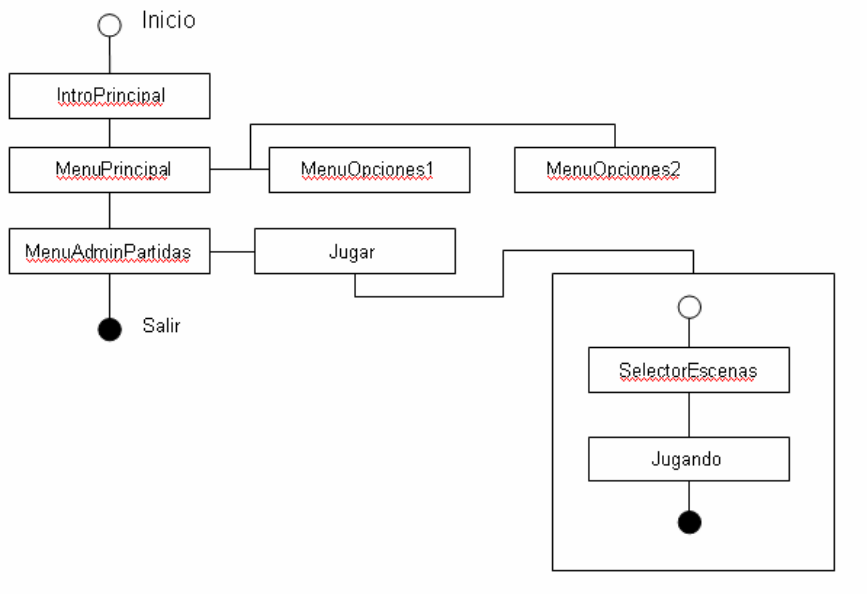
Para fantasía mitológica Colombiana el ciclo es algo más extenso debido a que en él se pasan por las diferentes interfaces que tiene como los son en su orden:

- Videos Introductorias
- Pantalla Grupo GR Media
- Pantalla Inicial
- Pantalla de Configuración
- Pantalla de Administración de Partidas
- Pantalla de Selección de Escenas
- Pantalla de Juego

Para poderlas manejar se manejaron los siguientes estados dependiendo del lugar donde se encontrara y el destino al cual deseaba llegar

- Inicio
- MenuPrincipal
- MenuAdminPartidas
- IntroPrincipal
- MenuOpciones1
- MenuOpciones2
- Jugar
- SelectorEscenas
- Jugando
- MenuPrincipal
- Salir

Figura 149. Estados para Fantasía Mitológica Colombiana



Fuente: Los Autores

En el gráfico se puede apreciar los pasos que el juego va ejecutando para poder llegar al juego que se desarrolla en el estado jugar.

23. IMPLEMENTACIÓN DE ACTORES PARA FMC

En capítulos anteriores ya se ha descrito como funcionan las diferentes características del motor y de la capa de definición de tipos (capas 1 y 2) en este capítulo se explicará la manera en que todos estos elementos se deben conjugar para hacer y/o representar los diferentes personajes de un videojuego, para lo cual se hará un análisis de las implementaciones más representativas de los actores que existen en FMC, a saber:

- CSZue: Implementación el personaje principal
- CSZancudo: Implementación de un enemigo simple
- CSActorParlante: Implementación de un elemento simple
- CSBufeo: Implementación de un enemigo principal

Es necesario tener en cuenta que estos no son todos los actores que se utilizaron para realizar el juego, sino simplemente son un muestreo representativo de la totalidad de actores, este muestreo se ha realizado puesto que exponer y analizar cada uno de los actores sería redundante pues en general utilizan las mismas características y solo cambia el desarrollo de su inteligencia artificial y su alternación entre estados, puntos que resultan imposibles analizar en detalles particulares ya que están fundamentados en gran parte del proceso de desarrollo de las caracterizaciones de cada personaje por lo cual es mucho más sencillo acudir a la descripción de la caracterización artística de cada uno que recurrir al análisis del desarrollo que permite al personaje adoptar una caracterización en particular.

23.1. CSZUE Y CSBUFEO

Esta es la implementación del actor principal de FMC, Zue es el protagonista del videojuego. En conjunto con CSBufeo (implementación de un enemigo principal) son las implementaciones de actor más complejas. Ambas implementaciones comparten su complejidad pero la diferencia es que una de responde a los estímulos directos del usuario a través de los dispositivos de entrada y la otra responde a los diferentes estados y situaciones del juego con respuestas programadas artificialmente.

A continuación se analizará el actor principal, CSZue por ser el que contiene los aspectos de CSBufeo.

Para programar a Zue se hizo uso de la clase CSActorCtrlInv[*] por lo cual usa toda las características que posee esta plantilla.

[*] Ver Cáp. 16.4. DEFINICIÓN DE CLASES BASE PARA ACTORES y 16.1.3. Tipos de Actor del presente documento

También existen estados desencadenantes, estos hacen un llamado a un actor secundario dentro de la clase CSZue y alternan su procesamiento con el de la propia clase, estos actores secundarios son utilizados para los ‘poderes especiales’ que posee el personaje.

23.1.1. Estados de Zue. Los estados del personaje principal definen sus comportamientos más característicos y son:

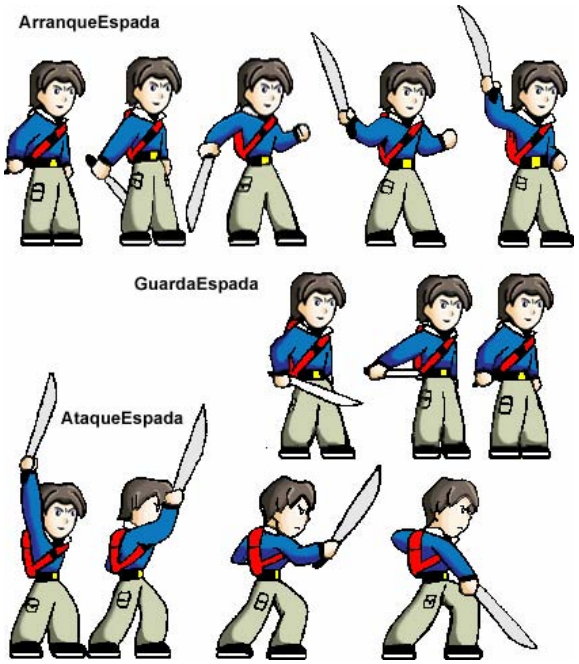
- Reposo : El actor esta quieto
- Caminar: El actor esta caminando
- Saltar: El actor esta saltando
- Golpea: El actor esta golpeando
- Magia: El actor esta produciendo magia
- Herido: El actor ha sido lastimado
- Muerto: El actor ha muerto
- Salto Camina : El actor ha saltado y se desplaza horizontalmente en el aire
- Salir: El actor envía la petición de salir de la escena.

Existen algunos subastados que fueron creados para controlar otras variantes del personaje cuando esta ejecutando algunas actividades particulares.

23.1.1.1. Estados de Espada. El manejo de la espada a nivel de programación es complejo, por lo cual es necesario partir toda la serie en subestados (Figura 150).

- Arranque Espada: Sucede al desenfundar la espada
- Guarda Espada: Sucede al guardar la espada
- Ataque Espada: Sucede cuando se esta atacando con la espada

Figura 150. Estados del lanzamiento de espada

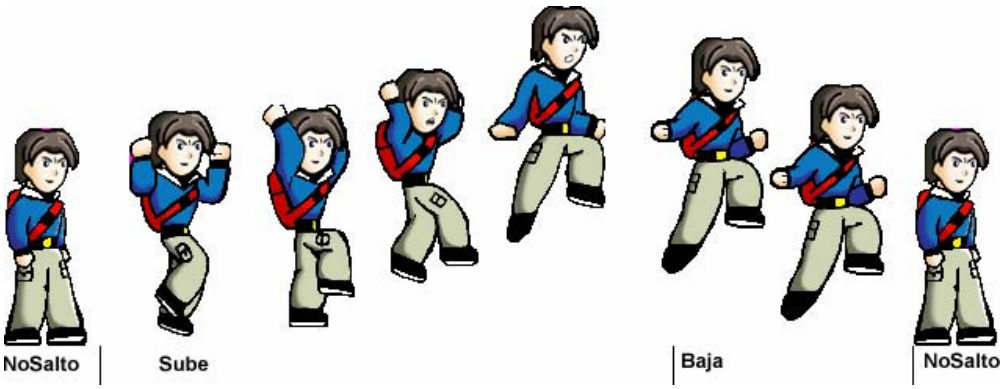


Fuente: Los Autores

23.1.1.2. Estados de salto. El salto comprende realmente tres comportamientos diferentes y para cada uno se ha definido un estado (Figura 151).

- Sube: Indica que el salto ha comenzado a elevarse
- Baja: Indica que el salto ha comenzado a descender
- No Salto: Indica que el salto no ha comenzado

Figura 151. Estados del salto



Fuente: Los Autores

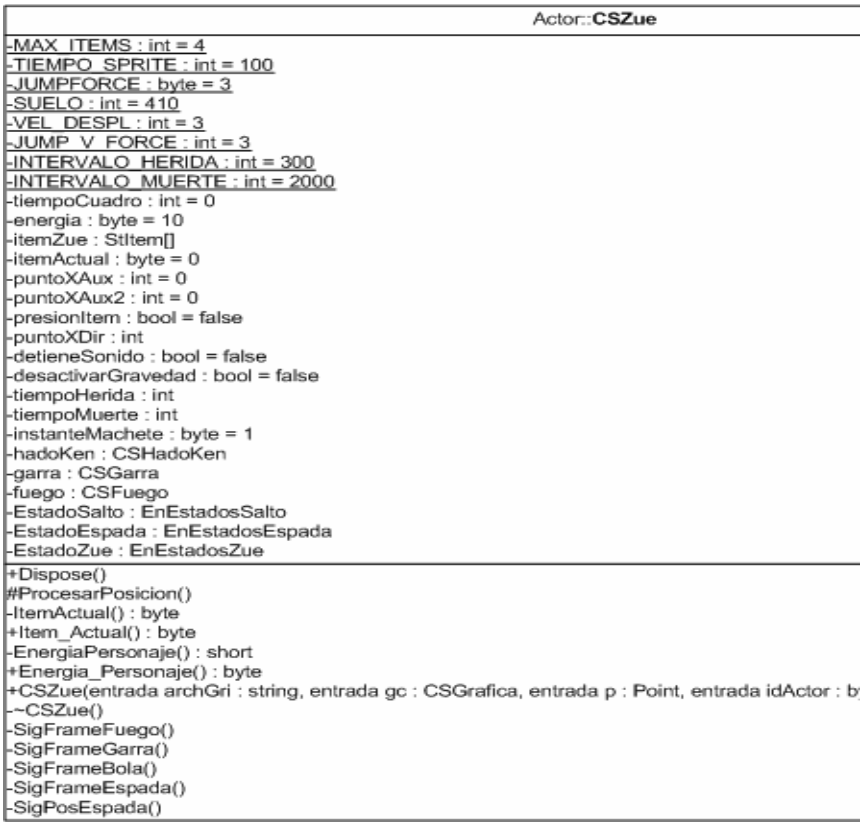
23.1.2. Estados desencadenantes de Zue. El estado de golpe en este actor desencadena la actividad de los actores secundarios involucrados con el personaje estos actores son:

- CSGarra : Libera un haz de poder a manera de garra delante del personaje
- CSFuego: Enciende una fogata alrededor del personaje
- CSHadoKen: Lanza una bola de poder en la dirección en a que esta mirando el personaje.

Se determina que 'poderes' (actores) se deben lanzar de acuerdo a un indicador del personaje que muestra cual es el poder que ha seleccionado actualmente.

23.1.3. Diagrama de clases. Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 152.Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) en los capítulos 5 y 6.

Figura 152. Diagrama de clases de CSZue



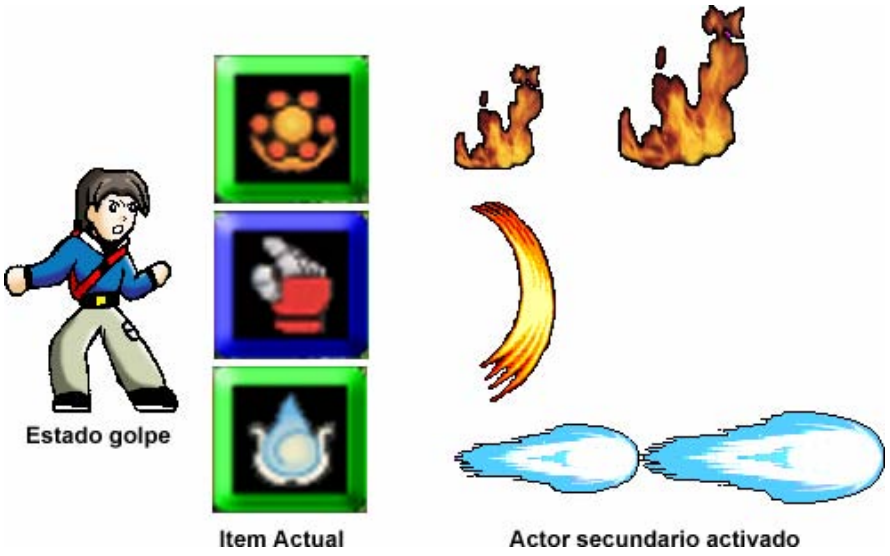
Fuente: Los Autores

23.2. ACTORES SECUNDARIOS DE CSZUE

Los actores secundarios de este personaje fueron diseñados para facilitar el desarrollo de los comportamientos del personaje cuando este envía uno de sus poderes especiales y así simplificar el ya de por si complejo intercambio de estados en este personaje.

En cada ciclo del bucle de juego del personaje principal se realiza llamado correspondiente a los métodos de los personajes secundarios del actor dependiendo del estado en que se encuentre y del ítem que tenga actualmente activo (Figura 153).

Figura 153. Llamado de actores secundarios de acuerdo al estado y al ítem actual del personaje



Fuente: Los Autores

23.2.1. CSHadoKen. Este posee dos únicos estados: activo e inactivo el primer estado provoca que el actor aparezca y se desplace horizontalmente en la dirección en que se encuentra el punto de observación del personaje principal y el segundo produce que se desaparezca de la pantalla (Figura 154).

Figura 154. Cuadros de animación del actor CSHadoKen

Fuente: Los Autores

23.2.2. CSFuego. Al igual que el anterior actor secundario solo posee dos únicos estados: activo e inactivo el primer estado provoca que el actor aparezca y permanezca animado durante un periodo de tiempo determinado para simular el efecto de fuego encendido (Figura 155).

Figura 155. Cuadros de animación del actor CSFuego

Fuente: Los Autores

23.2.3. CSGarra. Al igual que el anterior actor secundario solo posee dos únicos estados: activo e inactivo el primer estado provoca que el actor aparezca y permanezca animado durante un periodo de tiempo determinado mientras se desplaza suavemente en dirección al punto de observación del personaje principal (Figura 156).

Figura 156. Cuadros de animación del actor CSGarra

Fuente: Los Autores

23.2.4. Diagramas de clase de actores secundarios. Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de las Figura 157, 158, 159.

Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 157. Diagrama de Clases de CSGarra

Actor::**CSGarra**
-VELOCIDAD : int = 10 -activo : bool = false -puntoIni : Point
+Dispose() #ProcesarPosicion(entrada actuador : Point) +CSGarra(entrada archGri : string, entrada gc : CSGrafica, entrada p : Point, entrada idActor : byte, entrada volumenSonido : float) -~CSGarra() +ProcesarSigFrame() +ProcesarPosicion() +Lanzar(entrada d : EnDireccion, entrada x : int, entrada y : int)

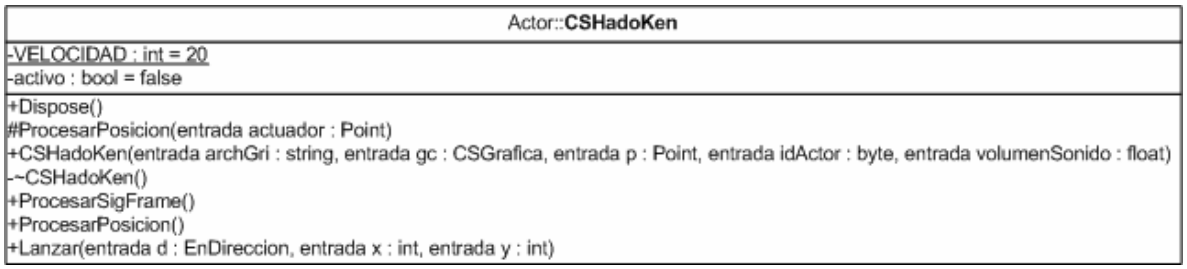
Fuente: Los Autores

Figura 158. Diagrama de Clases de CSFuego

Actor::**CSFuego**
-TIEMPO_CAMBIO_FRAME : int = 100 -TIEMPO_DE_VIDA : int = 2500 -tiempoIni : int = 0 -tiempo : int = 0 -activo : bool = false
+Dispose() #ProcesarPosicion(entrada actuador : Point) +CSFuego(entrada archGri : string, entrada gc : CSGrafica, entrada p : Point, entrada idActor : byte, entrada volumenSonido : float) -~CSFuego() +ProcesarSigFrame() +ProcesarPosicion() +Lanzar(entrada d : EnDireccion, entrada x : int, entrada y : int)

Fuente: Los Autores

Figura 159. Diagrama de Clases de CSHadoKen



Fuente: Los Autores

23.3. CSZANCUDO.

Esta es la implementación representativa de actores de rango intermedio entre los cuales también se pueden encontrar la candileja.

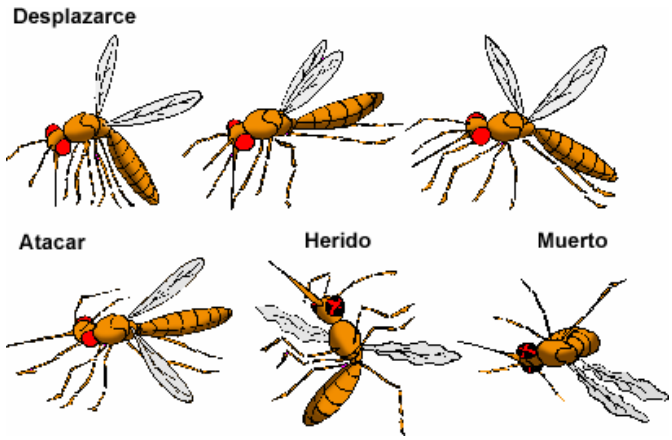
23.3.1. Estados de CSZancudo. Aunque la implementación define los estados para el zancudo, estos estados son igualmente validos para actores de gama media, los cuales presentan un rango similar de movimientos y estilo de comportamiento homogéneo.

Los estados son lo siguientes:

- Desplazarse: movimiento habitual del personaje
- Atacar: se produce al estar cerca del actor principal
- Herido: ocurre cuando el actor principal en estado de ataque entra en contacto con el actor.
- Muerto: después de múltiples estados de herido el personaje muere.

Se puede ver mejor el ejemplo en la Figura 160.

Figura 160. Estados de CSZancudo



Fuente: Los Autores

23.3.2. Diagramas de clase de actores secundarios. Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de las Figura 161.

Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 161. Diagrama de Clases de CSZancudo

Actor::**CSZancudo**
-tiempoSprite : int = 0 -tiempoVertical : int = 0 -TIEMPO_CAMBIO_FRAME : int = 50 -VELOCIDAD_ZANCUDO_H : int = 5 -ALTURA_VUELO_MAX : int = -10 -ALTURA_VUELO_MIN : int = 80 -TIEMPO_CAMBIO_VERTICAL : int = 100 -VELOCIDAD_ZANCUDO_V : int = 10 -SUELO : int = 370 -DISTANCIA_ATAQUE_X : int = 50 -DISTANCIA_ATAQUE_V : int = 80 -EstadoZancudo : EnEstadosZancudo -direccion : EnDireccion -potencia_rand : Random
+Dispose() #ProcesarPosicion(entrada actuador : Point) +CSZancudo(entrada archGri : string, entrada gc : CSGrafica, entrada p : Point, entrada idActor : byte, entrada volumenSonido : float) +CSZancudo(entrada archGri : string, entrada gc : CSGrafica, entrada cc : int, entrada p : Point, entrada idActor : byte, entrada volumenSonido : float) +CSZancudo(entrada archGri : string, entrada gc : CSGrafica, entrada ccb : int, entrada cca : int, entrada p : Point, entrada idActor : byte, entrada volumenSonido : float) -CSZancudoAd() -IA(entrada y salida actuador : Point)

Fuente: Los Autores

23.4. CSACTORPALANTE.

Esta es la implementación representativa de actores de rango bajo entre los cuales se encuentran los personajes de los pueblos que únicamente hablan.

23.4.1. Estados de CSActorParlante. Únicamente existen dos estados: callado, hablando, ver ejemplo en la Figura 162.

Figura 162. Estados de CSActorParlante

Fuente: Los Autores

23.4.2. Diagramas de clase. Teniendo en cuenta las necesidades, las diferentes situaciones que se han descrito en apartados anteriores y el diseño global del motor se ha diseñado el diagrama de clases de la Figura 163.

Mayores detalles de la implementación y funcionalidades se pueden apreciar en el Anexo A (documentación de desarrollo) y en los capítulos 5 y 6.

Figura 163. Diagrama de Clases de CSActorParlante

Actor::**CSActorParlante**
-tiempoCuadro : int = 0 -EstadoParlante : EnEstadosParlante -SUELO : int = 410
+Dispose() #ProcesarPosicion(entrada actuador : Point) +CSActorParlante(entrada archGri : string, entrada gc : CSGrafica, entrada p : Point, entrada idActor : byte, entrada volumenSonido : float) -~CSActorParlante()

Fuente: Los Autores

24. SITIO WEB DEL JUEGO

24.1. DISEÑO DE CONTENIDO

El sitio Web del juego, se creó con el fin de mostrar al público conceptos básicos de este proyecto de grado.

Este sitio se compone de 2 secciones, la primera es la parte promocional del videojuego que hace referencia a los siguientes aspectos:

- La historia que se desarrolla en el juego.
- La descripción básica de los personajes.

La segunda parte hace referencia a la información general del proyecto de grado extraída del documento descriptivo del juego, los aspectos incluídos en esta parte del sitio son:

- Filosofía
- Preguntas Frecuentes
- Características del Juego
- El Mundo del Juego
- Aspecto Físico
- Engine (Motor)
- Y la sección acerca de…

24.2. DISEÑO DE INTERFAZ

24.2.1. Sección Promocional. Para hacer más atractiva la interfaz gráfica del sitio, se opto por elaborar un escenario cuyo color es alusivo a cada personaje principal del juego, dejando una silueta con la forma de cada uno de los personajes para darle un toque de misterio con respecto a la forma física. Cuando el demo del juego este completo se reemplazarán dichas siluetas con los gráficos definitivos de los personajes.

Esta sección cuenta con el titulo del juego, una barra de navegación principal, 2 bloques de animación flash, el primero es un escenario animado, el segundo es una animación de texto dinámico y por último los créditos de elaboración del sitio.

La parte de los personajes cuenta con una barra de navegación vertical con los vínculos hacia cada uno de los escenarios.

24.2.2. Sección Información General. Su interfaz gráfica se diseño de tal forma que los usuarios pudiesen acceder de forma rápida a los contenidos listados anteriormente, utilizando el sistema de marcos para la estructura de esta parte del sitio, su aspecto visual es muy diferente al de la sección anterior pero su diseño es atractivo y ágil.

Cuenta también con el título del Proyecto, una barra de navegación general, un menú principal en la parte izquierda y un marco central para la información de los contenidos.

24.3. DESARROLLO FLASH Y HTML

24.3.1. Sección Promocional. Se escogió a Macromedia Flash MX 2004 en conjunto con Macromedia Fireworks Mx 2004 y Adobe Photoshop CS para la elaboración, modificación de los gráficos y animaciones utilizados en los escenarios.

Se implemento campos de texto dinámico con barra de desplazamiento dentro de las presentaciones flash para la incorporación de información.

Se desarrollo un micro aplicativo incorporado dentro de las aplicaciones flash, denominado precargador, cuya función es mostrar el progreso de la carga de la presentación flash en cada una de las secciones del sitio.

24.4. PUBLICACIÓN

Después de realizar la investigación del caso con respecto al alojamiento se encontró una opción de hosting de acuerdo a nuestras necesidades con la empresa Imagine S.A.
http://www.imagine.com.co/fmc/index.html

Figura 164 Sitio Web

Fuente: Los Autores

25. RESULTADOS

Los siguientes son los resultados más significativos del proyecto, la lista sería muy amplia, si describiera cada uno de los retos superados para llevar a cabo un proyecto de este tipo.

25.1. MOTOR DE VIDEOJUEGO

El motor de un videojuego es el corazón del mismo el que hace que todo el mudo mágico diseñado sea capaz de funcionar. Nuestro más sobresaliente resultado es haber creado un motor capaz de soportar videojuegos en 2 dimensiones.

En los capítulos anteriores se han podio apreciar las características del mismo, y su organización a mas bajo el nivel, el diseño del mismo tomo de varias reuniones de definición de objetivos y funcionalidades soportadas.

Algunas características se examinan a continuación.

25.1.1. Escenarios.

- Creación y manipulación de escenario.
- Manejo de múltiples capas(efectos de profundidad y movimiento)
- Implementación de celdas(técnica de tiles)
- Implementación de técnicas avanzadas de manipulación de marcos
- Métodos de administración de recursos en memoria para escenarios

25.1.2. Actores

- Creación de formatos gráficos para secuencias animadas
- Implementación y definición de interfaces para la creación de personajes animados
- técnicas de herencia para de recursos en memoria para personajes
- Diseño e implementación de administrador de personajes, para controlar los personajes en el juego.
- Administrador de personajes con interfaces que para la gestión de múltiples personajes en escena.

25.1.3. Graficas

- Manejador gráfico que abstrae la complejidad hardware permitiendo la utilización de los recursos relacionados con el video.
- Soporte para diferentes tipos blitting en superficies de memoria de 16, 24, 32 bits de profundidad de color.
- Soporte para tres tipos de resolución 640*480, 600*800, 1027*768.
- Compatibilidad para modo pantalla completa y modo ventada.
- Soporte de alternación entre modo de pantalla.
- Cambio Manual del tamaño de la ventana.
- Soporte para manejo de doble-buffer, controlando el barrido vertical de pantalla.
- Funcionalidades de texto en pantalla y dibujo de formas geométricos básicas

25.1.4. Videos

- Soporte para la reproducción de video en diferentes formatos (wmv, avi)
- Funcionalidades que permiten alternar entre modo de video y modo gráfico
- Reproducción asíncrona de video
- Soporte para lista de reproducción

25.1.5. Sonidos

- Soporte para formato wav
- Soporte para lista de reproducción
- Reproducción síncrona de sonidos

25.1.6. Música

- Soporte para formato wma, mp3.
- Soporte para lista de reproducción
- Reproducción sincronía de música
- Implementación de efectos de sonido(possitive crossfrading, negative crossfrading)

25.1.7. Entrada

- Ssoporte que abstrae la complejidad hardware permitiendo la utilización de diferentes dispositivos I/O.
- Soporte para Mouse
- Soporte para Teclado
- Soporte para joystick
- Interfaz para unificar el manejo de los dispositivos

25.1.8. Manejador de archivos

- Permite guardar y recuperar partidas en formato xml
- Soporte parametrito para determinar la cantidad de partidas

25.1.9. Definición de formato de archivo para escenas

- Herramienta para la construcción de archivos binarios para escenarios
- Cargador de archivos binarios de escenas interno.

25.1.10. Definición de formato de archivo para personajes

- Herramienta para la construcción de archivos binarios para personajes
- Cargador de archivos binarios de personajes interno.

25.1.11. Inteligencia artificial y personajes.

- Se desarrolló la inteligencia artificial para los personajes
- Los personajes responden de diferentes maneras a las situaciones que se les presentan. Con respecto al personaje principal.

25.2. DISEÑO DEL VIDEOJUEGO DE FANTASÍA MITOLÓGICA COLOMBIANA

La elaboración de un videojuego no solo tiene gran complejidad en el desarrollo del motor que lo soporte, la creación de juegos siempre a implicado un reto para nuestro intelecto y su caracterización mide cuanto de artista llevamos dentro, no es fácil construir una historia que cautive a los usuarios que esta ocasión son mas críticos que para una aplicación normal, la creación de videojuego debe cautivar por su creatividad, por la combinación de fantasía y realidad, por estar basados en un mundo mágico pero con reglas reales.

Fantasía mitológica colombiana se basó en la mitología colombiana, quién cuando niño no se cautivo y no pudo dormir después de una charla sobre la patasola, o sobre un delfín que toma forma humana en la selva amazónica, quién no escuchó del mohan, el cual no es solo uno tiene varias versiones en los diferentes departamentos que habita.

La historia es ficticia pero en ella se caracteriza la cultura de nuestros pueblos, el potencial que podrían tener los videojuegos para enseñar a los niños sobre gran variedad de temas es un recurso no explotado. Con fantasía mitológica se acerca al jugar a nuestros mitos y leyendas.

25.3. DISEÑO ARTÍSTICO

Aunque no es propio de nuestra profesión, el arte gráfico es en compañía con el motor y la historia en un video juego lo más importante para que al publico le guste. Un gran motor el cual soporte todas las funcionalidades necesarias no sería exitoso si no se explota con imágenes innovadoras y de gran calidad.

Para fantasía mitológica se desarrollo un diseño gráfico, con dibujantes colombianos y en colaboración con algunos diseñaros amigos del proyecto, aunque fue un cuello de botella para el proyecto ya que en nuestro medio no son muchos los diseñadores que tengan habilidades para la animación, y las empresas dedicadas a esta labor tienen costos muy altos.

El haber construido las imágenes, sonidos, videos, animaciones para el juego es un logro muy importante, ya que no pertenecen a nuestra especialidad y de no haber logrado la calidad a la que se llegó, el impacto de videojuego sería menor. Aspecto que como producto podría ser muy perjudicial.

Los desarrollos artísticos más importantes fueron:

25.3.1. Animación

- Conceptualización de los personajes con base en el guión
- Diseño de Bocetos para personajes
- Evaluación y selección de bocetos
- Elaboración de secuencias de animación en medio físico (Papel y tinta)
- Digitalización de Imágenes
- Evaluación de herramientas para manipulación de gráficos
- Corrección de detalles posteriores a digitalización
- Creación de secuencias animadas
- Pruebas de animación en blanco y negro(Compilación de fotogramas)
- Correcciones a la animación
- Aplicación de color a las secuencias animadas
- Conversión de animaciones al formato GRI

25.3.2. Escenarios

- Elaboración del tablero de historia con base en el guión
- Creación y modelación de texturas para tercera capa del escenario (Relieve, Follaje)
- Adaptación de texturas al sistema de Tiles
- Creación y modelación de texturas para la elaboración de los elementos de las capas 2 y 4 (Árboles).
- Adecuación de las texturas de los elementos de capas 2 y 3 al sistema de tiles.
- Generación del archivo de escena (*.GRE) utilizando los gráficos en formato Tile.
- Menús (Entorno Gráfico)
- Elaboración del listado de opciones que estarán presentes en los menús.
- Generación de los diagramas de flujo para navegación entre menús.
- Creación e implementación de imágenes para el entorno gráfico del sistema de menús. (Cursor, Estados de Botón, Fondos).

25.3.3. Pantalla de Ítems y Barra de Energía

- Creación e implementación de imágenes para el entorno gráfico del sistema de menú de ítems con base en el guión.
- Creación de las imágenes que harán parte de barra de energía.

25.3.4. Banda Sonora y Sonidos

Recopilación de los sonidos y música en formato (WAV,WMA,MP3) que forman parte de:

- Eventos Botón
- Fondos Musicales para pantallas de menús
- Fondos Musicales para escenarios
- SFX Efectos de sonido para escenarios (Rayos, Truenos, Cascadas, Riachuelos, etc.)
- SFX Efectos de sonido para personaje principal
- SFX Efectos de sonido para personajes secundarios (aleteo y canto de aves, gruñidos de fieras, aleteo mosquitos, etc.)

25.3.5. Multimedia

- Video con animación del logo G.R. Media
- Video con animación del Titulo de Juego

25.4. PROMOCIÓN Y DIVULGACIÓN DEL DESARROLLO DE VIDEOJUEGOS EN LATINOAMÉRICA.

Los videojuegos siempre nos han llegado como un producto importado y como tal nos hemos vuelto consumidores, pero en Latinoamérica hay gran talento para producir nuestros propios productos de entretenimiento, y al igual que el cine; no será fácil, pero tenemos que comenzar.

El fomento del desarrollo de videojuegos se esta propagando a través de la red mundial de Internet, el encontrar recursos en nuestro idioma no es fácil, y lo mas grave no es de calidad. Es por ello que la idea siempre fue que nuestro proyecto sea de dominio público.

La generación de preguntas el intercambio de ideas se realizo a través de foros y la pagina del proyecto. Nuestro siguiente objetivo es liberar el código para que sea analizado por expertos y novatos en el desarrollo de videojuegos.

26. CONCLUSIONES

No solo se logró la completar en su totalidad el videojuego Fantasía Mitológica Colombiana sino que el diseño del videojuego condujo a la creación de un motor con funcionalidades básicas que permite la implementación de multiples videojuegos en 2 dimensiones que permite la reutilizacion completa de todos los componentes diseñados y desarrollados para el núcleo.

Se creó una historia y un guión completo para el videojuego tomando como referencia de algunas características míticas y geográficas de Colombia, generando asi una historia animada con los personajes y escenarios necesarios para representarla.

El entorno de desarrollo diseñado es abierto a la implementación de diversas historias y escenarios y personajes con múltiples comportamientos lo cual permite que el entorno sea aplicable a muchos otros videojuegos o aplicaciones diferentes del viodeojuego para el cual fue planeado originalmente.

La interfaz de comandos entre el juego y el jugador es personalizable para el desarrollador y reutilizable para cualquier videojuego 2D gracias a que este componente fue diseñado de manera similar a los demás modulos que integran el motor.

Se logró el desarrollo completo del software y los elementos artisticos involucrados en la creación del videojuego sobrepasando las expectativas iniciales del proyecto y así logrando crear un conjunto de componentes de aplicación genérica para múltiples tipos de juegos de video en 2 dimensiones.

BIBLIOGRAFÍA

ALLEGRO. Librería Gráfica. [Online], oct 2004 – abril 2005 [consultado por última vez 4 de febrero de 2005] www.allegro.com .

OCAMPO LÓPEZ, Javier. Leyendas Populares Colombianas. Bogotá: Plaza & Janes Editores. 1998. 235p

GRADY, Booch. El Lenguaje Unificado de Modelado (UML). New York: Ed. Addison Wesley. 2000. 432p.

HARRISON LYN, Thomas. Designing and implementing a 3D engine with C#.NET and managed DirectX 9. New York: A Press. 2004. 406p.

IVAR, Jacobson; GRADY Booch; JAMES Rumbaugh. El Proceso Unificado de Desarrollo de Software. New York: Ed. Addison Wesley. 2000. 436p.

MSDN Library. Sitio oficial de la ayuda en línea para DirectX y lenguajes .NET [Online]. New York: Periodo citado oct 2004 – enero 2005 [consultado por última vez 4 de enero de 2005]. Disponible en internet: http://msdn.microsoft.com/library/

NINTENDO - MÉXICO. Reportajes. [Online]. Mexico: Edicion mensual. Periodo citado oct 2004 – abril 2005 [consultado por última vez 4 de mayo de 2005]. Disponible en internet: http://www.nintendo.com.mx/Reportajes/dit/

REVISTA CLUB NINTENDO México DF.: editorial Televisa S.A., 1994. Ediciónes 5-09 páginas 30, 35. Autor AxY.

RUIZ, Antonio. Programación de Videojuegos. Página Personal [Online]. Mexico: Periodo citado oct 2004 – abril 2005 [consultado por última vez 4 de mayo de 2005]. Disponible en internet: www.balancegames.com/camaleon.

SOLOCÓDIGO – Comunidad hispanoamericana de programadores. (Foro). [Online]. Madrid: oct 2003 – enero 2005 [consultado por última vez 4 de enero de 2005]. Disponible en internet: www.foros.solocodigo.com

TELEPORTMEDIA – Comunidad de desarrolladores de videojuegos. (Foro). s [Online] San Jose de Costa Rica: Periodo citado Oct 2003 – abril 2005 [consultado por última vez 10 de abril de 2005].
Disponible en internet: www.teleportmedia.com/foro , http://teleportmedia.com/viejoforo/

VJUEGOS Comunidad de desarrolladores de videojuegos (Foro)[Online]. Mexico: Periodo citado oct 2003 – enero 2005 [consultado por última vez 4 de enero de 2005]. Disponible en internet: http://www.vjuegos.org/modules.php?name=Forums

ANEXOS

Anexo A. Documentación del desarrollo
Dirigirse a los documentos del CDROM /doc/Documentación.chm , /VideoJuego/*.*

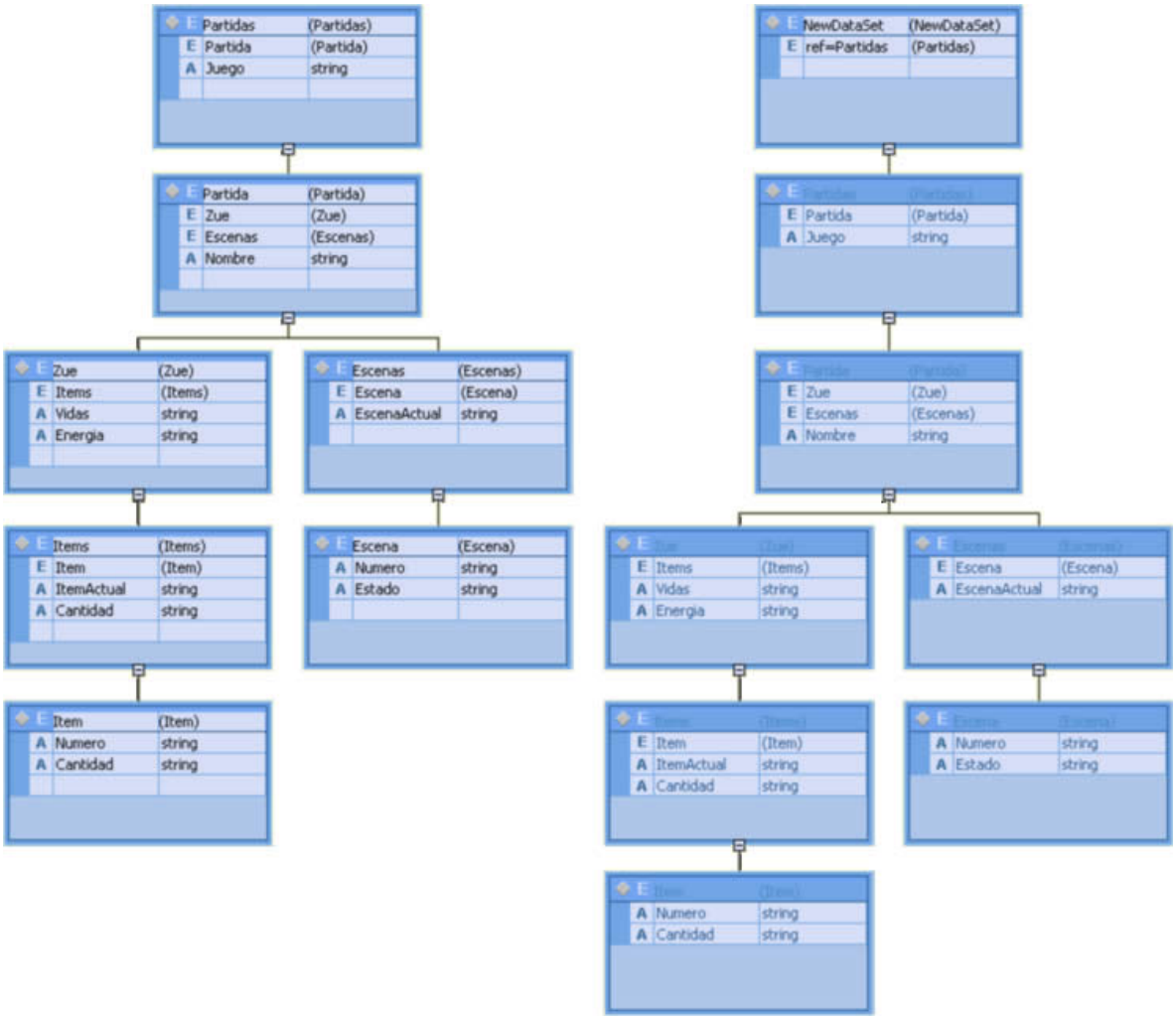
Anexo B. Descriptor XML para el administrador de partidas

```
<Partidas Juego="FMCZ">
  <Partida Nombre="claudio">
    <Zue Vidas="0" Energia="0">
      <Items ItemActual="0" Cantidad="0" />
    </Zue>
    <Escenas EscenaActual="1">
      <Escena Numero="1" Estado="NoIniciada" />
      <Escena Numero="2" Estado="NoIniciada" />
      <Escena Numero="3" Estado="NoIniciada" />
      <Escena Numero="4" Estado="NoIniciada" />
    </Escenas>
  </Partida>
  <Partida Nombre="Logan">
    <Zue Vidas="0" Energia="0">
      <Items ItemActual="0" Cantidad="0" />
    </Zue>
    <Escenas EscenaActual="1">
      <Escena Numero="1" Estado="NoIniciada" />
      <Escena Numero="2" Estado="NoIniciada" />
      <Escena Numero="3" Estado="NoIniciada" />
      <Escena Numero="4" Estado="NoIniciada" />
    </Escenas>
  </Partida>
  <Partida Nombre="NULL">
    <Zue Vidas="0" Energia="0">
      <Items ItemActual="0" Cantidad="0" />
    </Zue>
    <Escenas EscenaActual="1">
      <Escena Numero="1" Estado="NoIniciada" />
      <Escena Numero="2" Estado="NoIniciada" />
      <Escena Numero="3" Estado="NoIniciada" />
      <Escena Numero="4" Estado="NoIniciada" />
    </Escenas>
  </Partida>
  <Partida Nombre="gerna">
    <Zue Vidas="0" Energia="0">
      <Items ItemActual="0" Cantidad="0" />
    </Zue>
    <Escenas EscenaActual="1">
      <Escena Numero="1" Estado="NoIniciada" />
      <Escena Numero="2" Estado="NoIniciada" />
      <Escena Numero="3" Estado="NoIniciada" />
      <Escena Numero="4" Estado="NoIniciada" />
    </Escenas>
  </Partida>
</Partidas>
```

Anexo C. Descriptor XML para el administrador de partidas

```
<?xml version="1.0" ?>
<xs:schema id="NewDataSet"
targetNamespace="http://tempuri.org/savefile.xsd"
    xmlns:mstns="http://tempuri.org/savefile.xsd"
 xmlns="http://tempuri.org/savefile.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
 attributeFormDefault="qualified" elementFormDefault="qualified">
 <xs:element name="Partidas">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Partida" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Zue" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="Items" minOccurs="0"
maxOccurs="unbounded">
           <xs:complexType>
            <xs:sequence>
             <xs:element name="Item" minOccurs="0"
maxOccurs="unbounded">
              <xs:complexType>
               <xs:attribute name="Numero" form="unqualified"
                type="xs:string" />
               <xs:attribute name="Cantidad" form="unqualified"
                type="xs:string" />
              </xs:complexType>
             </xs:element>
            </xs:sequence>
            <xs:attribute name="ItemActual" form="unqualified"
                type="xs:string" />
            <xs:attribute name="Cantidad" form="unqualified"
                type="xs:string" />
           </xs:complexType>
          </xs:element>
         </xs:sequence>
         <xs:attribute name="Vidas" form="unqualified"
type="xs:string"
         />
         <xs:attribute name="Energia" form="unqualified"
type="xs:string"
         />
        </xs:complexType>
       </xs:element>
       <xs:element name="Escenas" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="Escena" minOccurs="0"
maxOccurs="unbounded">
           <xs:complexType>
            <xs:attribute name="Numero" form="unqualified"
            type="xs:string" />
            <xs:attribute name="Estado" form="unqualified"
            type="xs:string" />
           </xs:complexType>
          </xs:element>
         </xs:sequence>
         <xs:attribute name="EscenaActual" form="unqualified"
         type="xs:string" />
```

Anexo D. Diagrama descriptor XSD.



Anexo E. Evolución ilustraciones de Zue

Borrador

Concepto 1

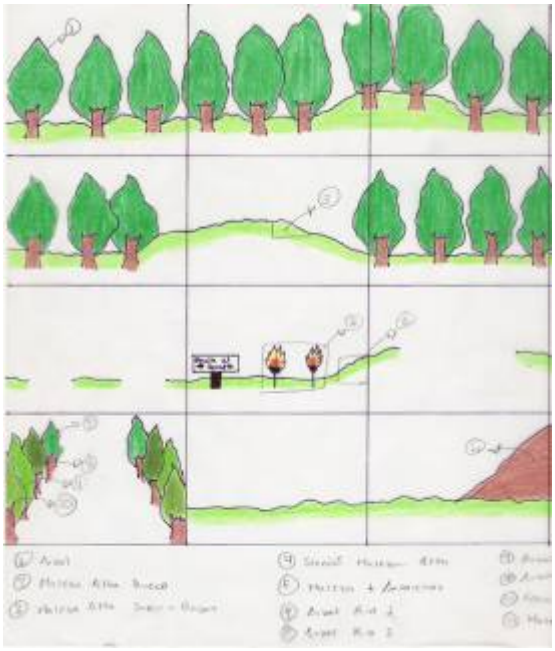
Concepto 2

Beta 1

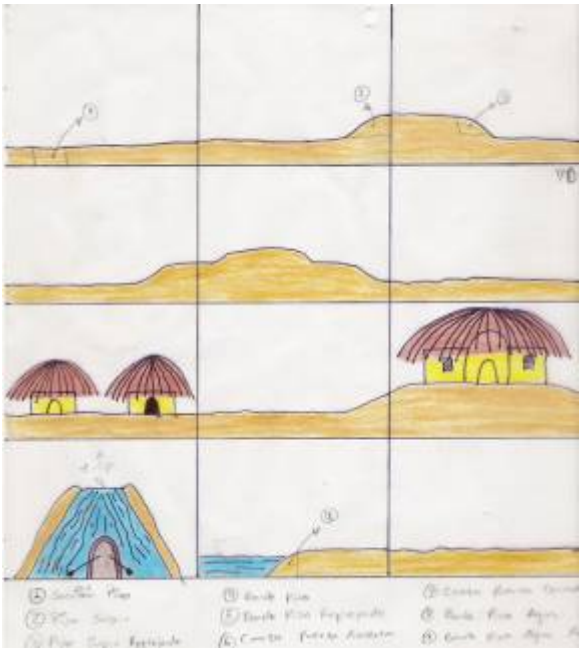
Beta 2

Final

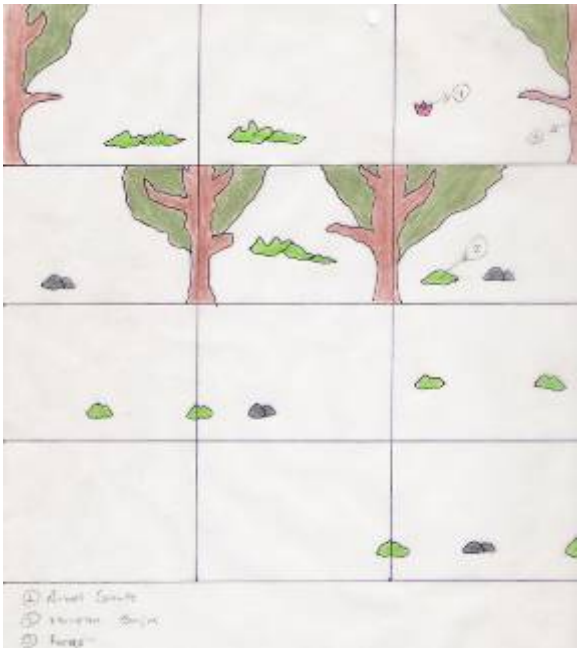
Anexo F. Tablero de historia



Tablero de historia Capa Escenografía



Tablero de historia Capa Relieve

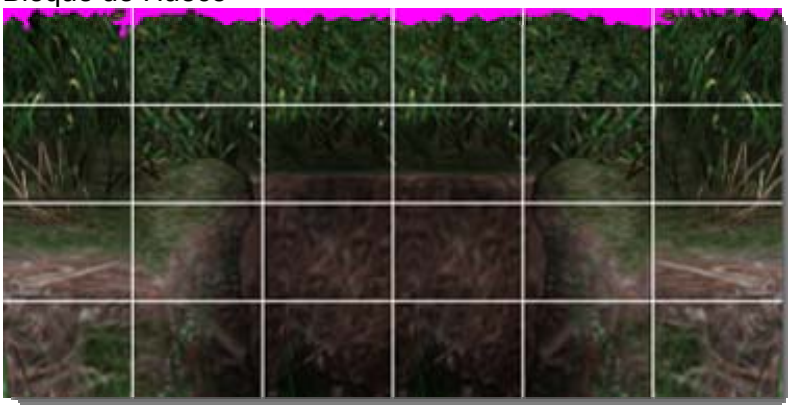


Tablero de historia Capa Frontal

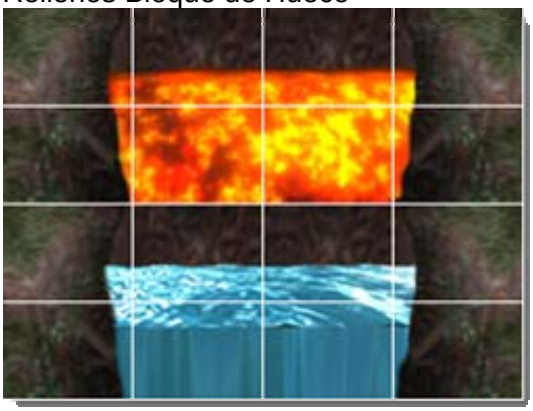
Anexo G. Tiles Escenas

Bloque de Piso

Bloque de Hueco

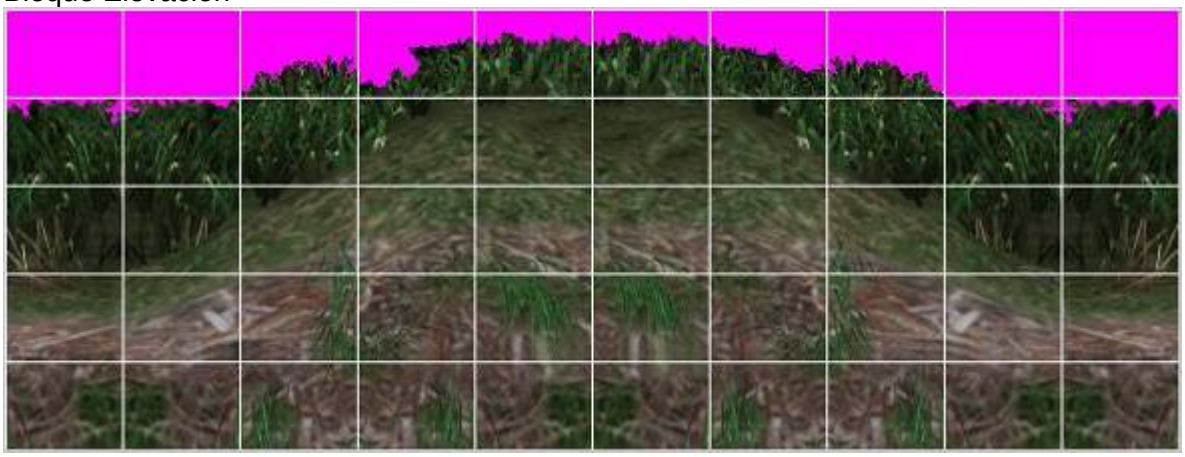


Rellenos Bloque de Hueco



Bloque de Púas

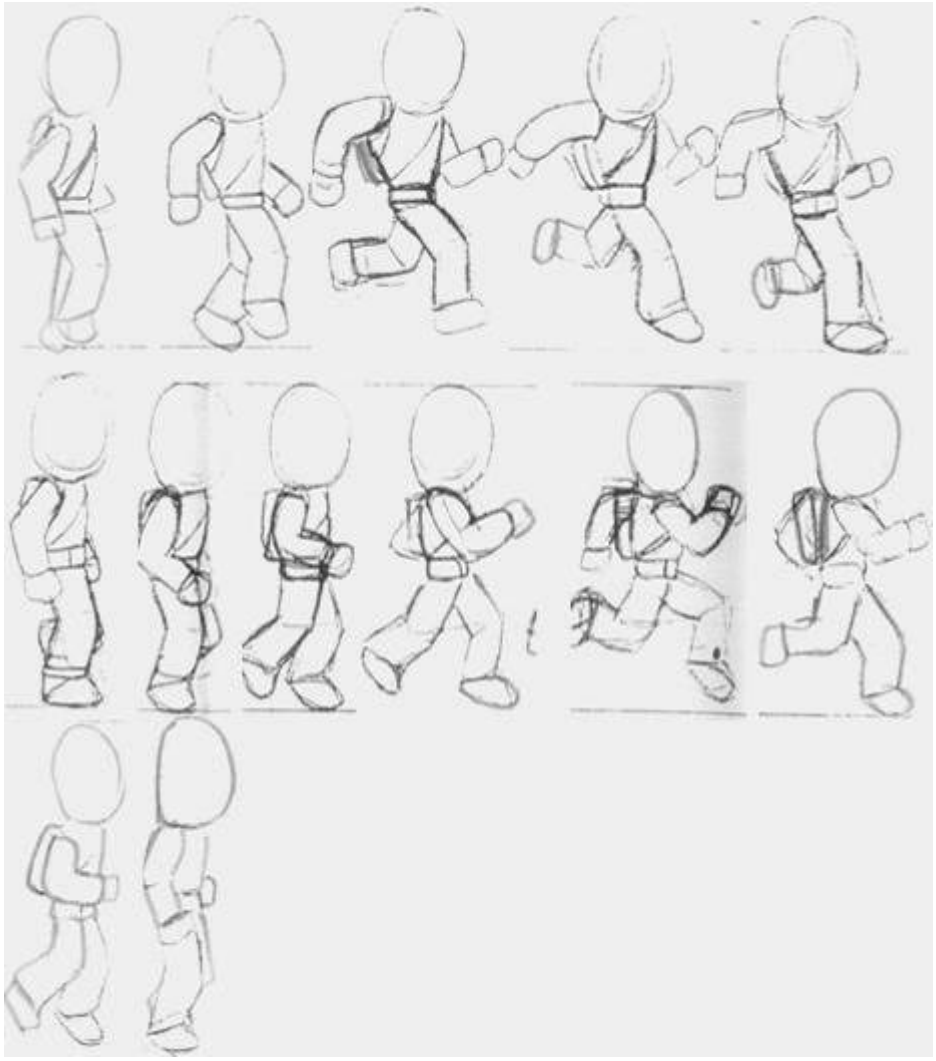
Bloque Elevación



Bloque Cascada

Choza

Anexo H. Primeros bocetos



Anexo I. Secuencia de animación de Zue

Correr

Saltar

Daño y Muerte

Magia Azul

Magia Fuego

Ataque con Garra

Ataque con Arma

Anexo J. Personajes secundarios

Familia Indígena

Hombre

Mujer

Niño

Otros

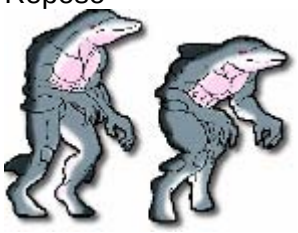
Chaman

Anciano

Anexo K. Animaciones enemigos principales.

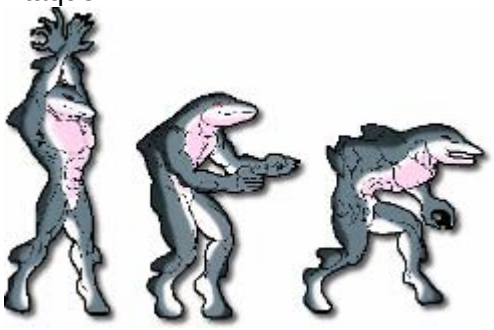
Bufeo

Reposo



Ataque 1

Ataque 2



Daño y Muerte

Anexo L. Animaciones enemigos secundarios.

Tigre

Reposo

Caminar

Ataque

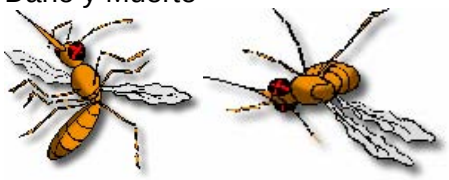
Daño y Muerte

Zancudo

Volar

Ataque

Daño y Muerte



Muerte

Candileja

Levitar

Ataque

Daño y Muerte