ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Run and Gun : The Most Wanted

José Ignacio Mateo Cruzado

Junio de 2010

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

Run and Gun: The Most Wanted

Autor del proyecto: José Ignacio Mateo Cruzado

Departamento: Lenguajes y sistemas informáticos

Directores del proyecto:

Daniel Molina Cabrera

Manuel Palomo Duarte

Índice de contenido

Agradecimientos

Le agradezco este trabajo a mi familia por su apoyo recibido. Sin ellos, no hubiera llegado hasta aquí. También me gustaría dedicar en estos días difíciles, el esfuerzo y dedicación puestos en este proyecto a mi abuelo, por todo lo que representó en mi infancia y aportó para ser lo que soy hoy día. Muchas gracias y hasta siempre.

Capítulo 1

Introducción

1.1. Objetivos: Proyecto propuesto

El objetivo principal del proyecto es la creación de un videojuego arcade 2D de tipo "run and gun" (subtipo de beat´em up) con scroll vertical.
El videojuego está programado en lenguaje C++. La mayoría de los programas usados para la elaboración de este, son software libres y gratuitos, al igual que todos los elemento que lo componen. Podrá ejecutarse tanto en versiones GNU/Linux como en Windows para captar la atención de más usuarios.
"The Most Wanted", el más buscado, titulo del videojuego, sitúa al jugador en el salvaje oeste pudiendo manejar a dos protagonistas diferentes, Billy Banks, un famoso forajido que se dedica a dar caza a los más peligrosos delincuentes y ladrones buscados o Toro Salvaje, un guerrero indio Sioux que se tomará la justicia por su cuenta para vengarse de todos los delitos infligidos contra su tribu. Ambos infringirán la ley y así convertirse en los más buscados.

Este videojuego no pretende competir con otros juegos comerciales ya que el trabajo ha recaído sobre una única persona, por lo que no es tan ambicioso como otros proyectos del mercado. Por lo que su objetivo es aprender, en líneas generales, cada una de las tareas involucradas en la creación de un videojuego y de las decisiones tomadas en esta.
The Most Wanted se caracteriza por la creación personalizada del diseño gráfico, realizado cada uno de ellos a mano con paciencia y dedicación. Y aunque está muy limitado por lo comentado anteriormente, se ha buscado ante todo la adicción y diversión, pensado para jugadores casuales, que no exige ninguna destreza debido a su simplicidad y corta duración.
El videojuego ha sido liberado bajo licencia GPL v3 (GNU General Public License).
Se incluyen los términos de la licencia en inglés al final de la memoria. Para mas información visite: http://www.gnu.org/licenses

1.2. Descripción

The Most Wanted es un videojuego del tipo “run and gun” inspirado en juegos del mismo género creados para recreativas entre los años 80 y 90.
Nuestro videojuego está ambientado en el oeste americano de la segunda mitad del siglo XIX.

En “The Most Wanted”, el scroll va avanzando con ritmo impuesto y nuestro protagonista puede disparar en tres direcciones, hacia delante y diagonalmente a derecha e izquierda, usando para ello los tres botones de disparo.
El juego tendrá cinco niveles con un mismo objetivo, acabar con todos los forajidos que puedas y sobre todo que ellos no acaben contigo. Aunque con temática diferente ya que cada escenario es diferente y los enemigos encontrados también. Al final de cada nivel, el jugador debe enfrentarse a un "Enemigo final" considerablemente más fuerte que los enemigos comunes, con movimientos propios, por lo que nos llevará más tiempo y esfuerzo para derrotarlo.
El jugador muere cuando es alcanzado por una bala enemiga. Por eso, en el transcurso del nivel, el jugador podrá ir destruyendo barriles, los cuales contienen objetos ocultos que te ayudarán, obteniendo vidas extras, potenciando la velocidad o haciéndote inmune.
Otro aspecto importante del juego es la puntuación conseguida. Conforme más sacos de dinero cojamos y mayor número de enemigos derrotemos mayor será el botín, pudiendo conseguir el récord de puntuación.
El videojuego también tiene modo dos jugadores. Dos amigos podrán jugar una partida contra la máquina colaborando el uno con el otro. El objetivo es el mismo aunque no es posible conseguir ningún récord aun alcanzando la máxima puntuación. A la hora de coger un objeto deberán repartirlo como buenos amigos, al igual que las vidas. Aunque si son egoístas y competidores intentan coger el mayor número de objetos posibles sin que el otro tome ninguno.

1.3. Estructura del documento

En esta sección resumimos el contenido de cada uno de los capítulos y apéndices que forman la memoria.

Capítulo 1: Expone los objetivos principales del proyecto, así como una breve descripción de nuestro proyecto comentado anteriormente.

Capítulo 2: Se muestran los conceptos básicos necesarios y un breve repaso de la evolución de la historia del videojuego para una mejor comprensión de la memoria.

Capítulo 3: Se expone la planificación del proyecto.

Capítulo 4: Se muestra el diseño del juego; desde el aspecto artístico (gráficos y sonidos) hasta el diseño del software (casos de uso y documentación de las funciones).

Capítulo 5: Se explica las diferentes tareas necesarias para el desarrollo del proyecto. Incluido pruebas y herramientas utilizadas.

Capítulo 6: Expone las conclusiones que se han podido extraer del proyecto y de posibles mejoras o ampliaciones futuras.

Apéndice A: En este apéndice incluimos manual de instalación y el manual de usuario, en el que podemos encontrar explicaciones de como instalarlo y como empezar a jugar (controles del juego, elementos de la interfaz y características).

Capítulo 2

El Videojuego

En este capítulo trataremos de forma general el concepto del videojuego, su definición e historia, desde los inicios hasta la actualidad. Luego nos centraremos más en el género **beat´em up**, al cual pertenece nuestro videojuego. Profundizando en sus características, historia y principales títulos del género.

2.1. Definición del videojuego

Un videojuego (llamado también juego de vídeo) es un programa informático, creado expresamente para divertir, basado en la interacción entre una persona y un aparato electrónico donde se ejecuta el videojuego. Estos recrean entornos virtuales en los cuales el jugador puede controlar a un personaje o cualquier otro elemento de dicho entorno, para conseguir uno o varios objetivos por medio de unas reglas determinadas.

2.2. Historia del videojuego

La historia de los videojuegos data de 1948, cuando la idea de un videojuego fue concebida y patentada por Thomas T. Goldsmith Jr. y Estle Ray Mann. En 1958 el primer videojuego salio a la venta al publico llamado Tenis para dos. Después en 1972 el Magnavox Odyssey fue lanzada la primera consola de videojuegos disponible al publico.

Los inicios

Durante bastante tiempo ha sido complicado señalar cual fue el primer videojuego, principalmente debido a las múltiples definiciones de este que se han ido estableciendo, pero se puede considerar como primer videojuego el Nought and crosses, también llamado OXO, desarrollado por Alexander S. Douglas en 1952. El juego era una versión computarizada del tres en raya que se ejecutaba sobre la EDSAC y permitía enfrentar a un jugador humano contra la máquina.

En 1957 William Higginbotham creó, sirviéndose de un programa para el cálculo de trayectorias y un osciloscopio, Tennis for Two: un simulador de tenis para entretenimiento de los visitantes del Brookhaven National Laboratory. Este videojuego fue el primero en permitir el juego entre dos jugadores humanos. Cuatro años más tarde, en 1961, Steve Russell, un estudiante del Instituto de Tecnología de Massachussets, dedicó seis meses a crear un juego para computadora usando gráficos vectoriales: Spacewar!. En este juego, dos jugadores controlaban la dirección y la velocidad de dos naves espaciales que luchaban entre ellas. El videojuego funcionaba sobre un PDP-1 y fue el primero en tener un cierto éxito aunque apenas fue conocido fuera del ámbito universitario.

Imagen 2.1. William Higginbotham, físico estadounidense. Extraída de http://www.flickr.com/photos/brookhavenlab/3147769961/

Imagen 2.2. PDP1. Extraída de http://indicelatino.com/juegos/historia/origenes

En 1966 Ralph Baer empezó a desarrollar junto a Bob Tremblay un proyecto de videojuego llamado Fox and Hounds dando inicio al videojuego doméstico. Este proyecto evolucionaría hasta convertirse en la Magnavox Odyssey, la primera consola doméstica de videojuegos lanzada en 1972 que se conectaba a la televisión y que permitía jugar a varios juegos pregrabados. La Odyssey Home System Entertainment es considerada la primera generación de videoconsolas.

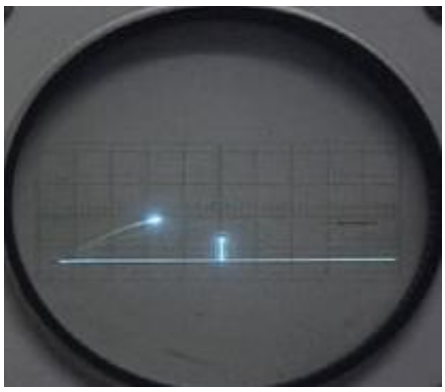


Imagen 2.3. Pong-tennis for two. Extraída de http://mikediluigi.com/wp-content/uploads/2008/12/tennis_for_two.jpg

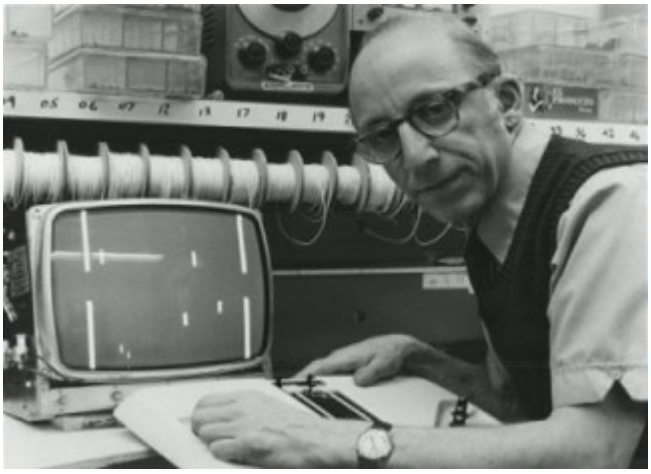


Imagen 2.4. Ralph Baer es considerado por muchos el inventor de los videojuegos. Extraída de http://indicelatino.com/juegos/historia/origenes/

Años 1970: La eclosión del videojuego

Un hito importante en el inicio de los videojuegos tuvo lugar en 1971 cuando Nolan Bushnell comenzó a comercializar Computer Space, una versión de Space War, en Estados Unidos, aunque es posible que se le adelantara Galaxy War otra versión recreativa de Space War aparecida a principios de los 70 en el campus de la universidad de Standford.

Imagen 2.5. Computer Space. Extraída de http://printliberation.com/blog/?p=2108

Imagen 2.6. Space War. Extraída de http://indicelatino.com/juegos/historia/origenes/

La ascensión de los videojuegos llegó con la máquina recreativa Pong, muy similar al Tennis for Two pero utilizada en lugares públicos: bares, salones, etc. El sistema fue diseñado por Harold Lee y Al Alcorn para Nolan Bushnell en la recién fundada Atari.

El juego se presentó en 1975 y fue la piedra angular del videojuego como industria. Durante los años siguientes se implantaron numerosos avances técnicos en los videojuegos (destacando los microprocesadores y los chips de memoria), aparecieron en los salones recreativos juegos como Space Invaders (Taito) o Asteroids (Atari) y sistemas domésticos como el Atari 2600.

Imagen 2.7. Asteroids. Extraída de http://www.gamealmighty.com/tag/hollywood

Años 1980: La década de los 8 bits

Los años 80 comenzaron con un fuerte crecimiento en el sector del videojuego alentado por la popularidad de los salones de máquinas recreativas y de las primeras videoconsolas aparecidas durante la década de los 70.

Durante los primeros años de la década llegaron al mercado doméstico sistemas como Oddyssey 2 (Phillips), Intellivision (Mattel), Colecovision (Coleco), Atari 5200, Commodore 64 (Commodore, Turbografx (NEC) mientras que en las máquinas recreativas triunfaron juegos como Pacman (Namco), Battle Zone (Atari), Pole Position (Namco), Tron (Midway) o Zaxxon (Sega). El negocio asociado a esta nueva industria alcanzó en poco tiempo grandes cosas. Sin embargo, en 1983 comenzó la que se ha dado por llamar crisis del videojuego de 1983, la cual afectó principalmente a Estados Unidos y Canadá, y que no llegaría a su fin hasta 1985.

En el resto del mundo se produjo una polarización dentro de los sistemas de videojuegos. Japón apostó por el mundo de las consolas con el éxito de la Famicom, consola lanzada por Nintendo en 1983 y conocida en occidente como NES (Nintendo Entertainment System), mientras que Europa se decantaba por los microordenadores como el Commodore 64 o el Spectrum.

Imagen 2.8. Famicom de Nintendo. Extraída de http://elblogdemanu.com/hiroshi-yamauchi/

A la salida de su particular crisis los norteamericanos continuaron la senda abierta por los japoneses y adoptaron la NES como principal sistema de videojuegos. A lo largo de la década fueron apareciendo nuevos sistemas domésticos como la Master System (Sega), el Amiga (Commodore) y el 7800 (Atari), que gozaron de diferentes niveles de popularidad según la región, y juegos hoy en día considerados clásicos como Tetris de Alexey Pajitnov. Ya hacia finales de los 80 comenzaron a aparecer las consolas de 16 bits como la Mega Drive (Genesis en Norteamérica) de Sega y los microordenadores fueron lentamente sustituidos por las computadoras personales basadas en la arquitectura de IBM.

Imagen 2.9. Master System de Sega. Extraída de http://www.turutupa.com/2008/04/01/nuestra-amada-master-system-ii/

En 1985 apareció Super Mario Bros. que supuso un punto de inflexión en el desarrollo de los juegos electrónicos. La mayoría de los juegos anteriores sólo contiene unas pocas pantallas que se repetían en un bucle y el objetivo simplemente era hacer una alta puntuación. El juego desarrollado por Nintendo supuso un estallido de creatividad.

Imagen 2.10. Super Mario Bros. Extraída de http://www.webdagoo.com/juega-todos-los-super-mario-bros-en-linea/

Por primera vez teníamos un objetivo y un final en un videojuego. En los años posteriores otras compañías emularon su estilo de juego.

En el campo de las recreativas, los 80 fueron una edad de oro con videojuegos como Defender, Rally - X, Dig Dug, Bubble Bobble, Gauntlet, Out Run o Shinobi, además de producirse un cambio en cuanto a la nacionalidad de los juegos, pasando a ser Japón la mayor productora de videojuegos para recreativas.

Otra rama de los videojuegos que creció con fuerza fue la de los videojuegos portátiles. Estos comenzaron a principios de los 70 con los primeros juegos completamente electrónicos lanzados por Mattel, los cuales difícilmente podían considerarse como videojuegos, y fueron creciendo en popularidad gracias a conversiones de recreativas como las realizadas por Coleco o adictivos microjuegos como las Game & Watch de Nintendo. La evolución definitiva de las portátiles como plataformas de videojuego llegó en 1989 con el lanzamiento de la Game Boy (Nintendo).

Años 1990: La revolución del 3D

A principios de los años 90 las videoconsolas dieron un importante salto técnico gracias a la competición de la llamada "generación de 16 bits" compuesta por la Mega Drive, la Super Famicom de Nintendo (cuyo nombre fue cambiado en occidente, pasando a ser Super Nintendo Entertainmet System "SNES"), la PC Engine de NEC, conocida como Turbografx en occidente y la CPS Changer (Capcom).

Junto a ellas también apareció la Neo Geo (SNK) una consola que igualaba las prestaciones técnicas de un arcade pero demasiado cara para llegar de forma masiva a los hogares.

Esta generación supuso un importante aumento en la cantidad de jugadores y la introducción de tecnologías como el CD-ROM, además de una importante evolución dentro de los diferentes géneros de videojuegos, principalmente gracias a las nuevas capacidades técnicas.

Mientras tanto diversas compañías habían comenzado a trabajar en videojuegos con entornos tridimensionales, principalmente en el campo de los PC, obteniendo diferentes resultados desde las "2D y media" de Doom, 3D completas de 4D Boxing a las 3D sobre entornos pre-renderizados de Alone in the Dark. Referente a las ya antiguas consolas de 16 bits, su mayor y último logro se produciría por el SNES mediante la tecnología 3-D de pre-renderizados de SGI, siendo su máxima expresión juegos como Donkey Kong Country y Killer Instinct. También surgió el primero juego poligonal en consola, la competencia de la SNES, Mega-Drive, lanzó el Virtual Racing, que tuvo un gran éxito ya que marcó un antes y un después en los juegos 3D en consola.

Imagen 2.11. Doom. Extraída de http://chancho-oscuro.com/?p=85

Rápidamente los videojuegos en 3D fueron ocupando un importante lugar en el mercado, principalmente gracias a la llamada "generación de 32 bits" en las videoconsolas: Sony PlayStation, Sega Saturn (que tuvo discretos resultados fuera de Japón); y la "generación de 64 bits" en las videoconsolas: Nintendo 64 y Atari jaguar. En cuanto a los PC, se crearon las aceleradoras 3D.

La consola de Sony apareció tras un proyecto iniciado con Nintendo (denominado SNES PlayStation), que consistía en un periférico para SNES con lector de CD. Al final Nintendo rechazó la propuesta de Sony, puesto que Sega había desarrollado algo parecido sin tener éxito, y Sony lanzó independientemente PlayStation.

Por su parte los arcades comenzaron un lento pero imparable declive según aumentaba el acceso a consolas y ordenadores más potentes. Para intentar compensar la huida de clientes, los fabricantes de máquinas arcade apostaron por potenciar hardwares específicos que difícilmente podían copiarse en un sistema doméstico como coches de tamaño real (Virtua Racing (Sega), Ridge Racer (Namco))

o pistas de baile (Dance Dance Revolution) entre otros. Desafortunadamente, la gran inversión que suponían estos aparatos sacó del mercado a muchos recreativos, con lo que el arcade pasó de ser un entretenimiento popular a estar recluido en unos pocos lugares muy específicos.

Por su parte los videojuegos portátiles, producto de las nuevas tecnologías más poderosas, comenzaron su verdadero auge, uniéndose a la Game Boy máquinas como la Game Gear (Sega), la Lynx (Atari) o la Neo Geo Pocket (SNK), aunque ninguna de ellas pudo hacerle frente a la popularidad de la Game Boy, siendo esta y sus descendientes (Game Boy Pocket, Game Boy Color, Game Boy Advance, Game Boy Advance SP, Game Boy Micro) las dominadoras del mercado.

Hacia finales de la de la década la consola más popular era la Playstation con títulos como Final Fantasy VII (Square), Resident Evil (Capcom), Winning Eleven 4 (Konami), Gran Turismo (Polyphony Digital) y Metal Gear Solid (Konami).

Imagen 2.12. Gran Turismo. Extraída de http://download.f60s.com/forums/t/76406.aspx

En PC eran muy populares los FPS como Quake (id Software), Unreal (Epic Megagames) o Half-Life (Valve) y los RTS como Command & Conquer (Westwood) o Starcraft (Blizzard Entertainment). Además las conexiones entre ordenadores mediante internet facilitaron el juego multijugador, convirtiéndolo en la opción predilecta de muchos jugadores, y fueron las responsables del nacimiento de los MMORPG como Ultima Online (Origin). Finalmente en 1998 apareció en Japón la Dreamcast (Sega), la cual llegaría a occidente en 1999 y daría comienzo a la "generación de los 128 bits".

Desde hay hasta nuestros días la industria del videojuego ha crecido a paso agigantado incorporando avances tecnológicos que parecen no tener límites.

2.3. Videojuegos Shoot'em up

2.3.1. Definición

Un «shoot 'em up», también escrito en inglés como «shmup», es un juego en el que el protagonista combate a un gran número de enemigos disparándoles mientras esquiva el fuego de estos. El jugador depende de su tiempo de reacción para tener éxito. Más allá de esto, los críticos difieren en qué elementos constituyen exactamente un shoot 'em up. Algunos restringen el género a juegos en los que aparece algún tipo de nave con movimiento sobre un fondo fijo o desplazante, es decir, lo que se vino conociendo en el mundo hispanohablante como «juegos matamarcianos» o «juegos de marcianitos». Otros incluyen además juegos en los que los protagonistas son robots o humanos a pie; juegos tipo «on-rails» («sobre raíles o rieles», o «hacia el interior de la pantalla») y juegos tipo «run and gun» («corre y dispara»). En un principio, los críticos describían cualquier juego donde el primer elemento de diseño era el disparo como «shoot 'em up», pero más tarde los shoot 'em up se convirtieron en un género específico y sectario basado en convenciones de diseño establecidas en los juegos de disparo de los años 80.

2.3.2. Diseño

Elementos comunes

Los shoot 'em up son un subgénero del género «shooter» («disparador»), clasificado a su vez como un tipo de videojuego de acción. Son juegos que se desarrollan normalmente vertical u horizontalmente en la pantalla, y los jugadores deben usar armas con determinado radio de acción. El avatar del jugador es típicamente un vehículo que se encuentra bajo constante ataque. Por lo tanto, el objetivo del jugador es disparar lo más rápidamente posible a todo lo que se mueva o lo amenace. En algunos juegos, el personaje del jugador puede soportar algún daño; en otros, un único blanco sobre este provoca su destrucción. Las principales habilidades requeridas en un shoot 'em up son las capacidades de reacción rápida y de memorizar los patrones de ataque enemigo. Algunos juegos presentan cantidades enormes de proyectiles enemigos, y el jugador debe memorizar sus patrones de una a otra partida si quiere sobrevivir. Los shoot 'em up son uno de los géneros de videojuegos en los que el ritmo de desarrollo de la acción es más acelerado.

Típicamente aparece un gran número de personajes enemigos. Estos enemigos pueden comportarse de un modo determinado en función de su tipo, o atacar en formaciones que el jugador debe aprender a predecir. El juego básico tiende a ser de avance continuo, y muchos compensan esto con batallas particulares contra «jefes» o «monstruos» y con diferentes armas. Los shoot 'em up raramente usan física real. Los personajes pueden cambiar de dirección instantáneamente, sin inercia, y los proyectiles se mueven en líneas rectas y a velocidades constantes. El personaje que controla el jugador puede recoger «power ups» (o «pows», «potenciadores») que dan a este mayor protección, una «vida» extra, o armas mejoradas. Las diferentes armas a menudo son adecuadas para diferentes enemigos, pero en este tipo de juegos raramente se sigue la pista a la munición disponible. En su lugar, los jugadores tienden a disparar indiscriminadamente, con la confianza de que sus armas sólo pueden dañar blancos legítimos.

Tipos

Los shoot 'em up se categorizan por sus elementos de diseño, especialmente el punto de vista del jugador y el movimiento: Los categorizados como "fixed shooter" (disparador fijo) constan de niveles cada uno de los cuales cabe en una sola pantalla. El movimiento del protagonista está fijado a un único eje, y los enemigos atacan en una única dirección (tal como la de descender desde la parte superior de la pantalla). Estos juegos se denominan a veces como "gallery shooters" (disparadores de galería). Los clasificados como "rail shooter" (disparador de raíl o riel) limitan al jugador a moverse por la pantalla mientras el juego sigue una ruta específica; estos juegos presentan un punto de vista hacia el interior de la pantalla, con el cual se ve la acción desde detrás del personaje. Los que reciben el nombre de "tube shooter" (disparador de tubo) presentan una nave que vuela por un tubo abstracto.

En los clasificados como "scrolling shooter" (disparador con desplazamiento de pantalla) se incluyen juegos con desplazamiento vertical u horizontal de la pantalla. En un shoot 'em up con desplazamiento vertical (o "vertical scroller"), la acción se ve desde arriba y la pantalla se desplaza de arriba a abajo (o de abajo a arriba, ocasionalmente). Ello tiene la ventaja de permitir complejos patrones de enemigos así como de hacer posible que gráficos simples, incluso, funcionen de forma convincente. Los vertical scrollers son los más adecuados para las máquinas arcade con pantallas altas; las pantallas usadas para computadoras personales o videoconsolas tienden a ser más anchas que altas y por lo tanto son menos adecuadas para juegos con desplazamiento vertical. El otro tipo de "scrolling shooter" es el horizontal shooter, también conocido en inglés como "side-scrolling shooter" (disparador de desplazamiento lateral), en el cual la acción se ve lateralmente y se desplaza horizontalmente. Un pequeño número de "scrolling shooters", tales como Zaxxon, de Sega, muestran una perspectiva isométrica. Otros presentan un desplazamiento de pantalla en pantalla mediante el uso de un dispositivo "flip-screen" (pantalla plegable): cuando el jugador alcanza el borde de la pantalla, todo un nuevo escenario aparece de una vez. Algunos shooters pueden presentar movimiento multidireccional ("multi-directional shooter"), generalmente con una pantalla estática.

"Bullet hell" (cortina de fuego o cortina de balas) es un shoot 'em up en el cual toda la pantalla está a menudo casi completamente llena de balas enemigas. También se conoce este tipo en inglés como "curtain fire" (fuego en cortina), "manic shooters" (disparadores frenéticos) o "maniac shooters"

(disparadores maníacos). Este estilo de juego se originó a mediados de la década de los 90, y es un descendiente de los "scrolling shooters". Los "cute 'em up" presentan gráficos de colores brillantes y representan escenarios y enemigos surrealistas. Los más modernos, particularmente los japoneses, emplean personajes e insinuaciones abiertamente sexuales.

"Run and gun" (o run & gun) describe un tipo de shoot 'em up en el que el protagonista se desplaza a pie, a veces con la posibilidad de saltar. Estos juegos usan perspectivas de desplazamiento lateral, vertical o isométrica, y pueden presentar además movimiento multidireccional. Estos tipos de juegos también pueden denominarse "scrolling shooters".

2.3.3. Historia

Orígenes y ascenso

Los orígenes exactos del género son un tema de cierta confusión. El periodista de videojuegos Brian Ashcraft señala a Spacewar! (uno de los primerísimos videojuegos) como el primer shoot 'em up, pero el posterior Space Invaders se cita más frecuentemente como el «primero» u «original» del género. Spacewar! fue desarrollado en el Instituto Tecnológico de Massachusetts en 1961, para diversión de los desarrolladores; no obstante se rehízo cuatro veces como juego arcade de principios a mediados de los 70. En el juego aparecía un combate entre dos naves espaciales, inspiradas por la contemporánea carrera espacial de la Guerra Fría. Sin embargo, no fue hasta la llegada en 1978 del seminal Space Invaders, creado por la empresa japonesa Taito Corporation, que el género se hiciese prolífico. Space Invaders enfrentaba al jugador contra múltiples enemigos que descendían desde la parte superior de la pantalla a una velocidad que aumentaba a ritmo constante. El juego usaba criaturas alienígenas inspiradas por La guerra de los mundos (de H. G. Wells) debido a que los desarrolladores eran incapaces de reproducir el movimiento de un avión; al mismo tiempo los alienígenas reemplazaban a los enemigos humanos por motivos morales (relativos a la representación de matanzas de seres humanos) en lo concerniente a Taito Corporation. Al igual que ocurriría con subsiguientes shoot 'em ups de la época, la acción se situaba en el espacio ya que la tecnología disponible sólo permitía un fondo negro. El juego introducía también la idea de dar al

jugador un número determinado de «vidas». Space Invaders fue un éxito comercial masivo, causando escasez de monedas en Japón. Al año siguiente, Galaxian, producido por Namco, llevó el género más lejos con patrones de enemigos más complejos y gráficos más elaborados.

Imagen 2.3.1. Space Invaders. Extraída de http://lauraberry.files.wordpress.com/2008/04/space-invaders.jpg

Edad de oro y refinamiento

En 1981 apareció el juego Defender, que introducía el desplazamiento de pantalla en los shoot 'em up, ofreciendo niveles que se extendían horizontalmente. A diferencia de la mayoría de los juegos posteriores del género, el jugador podía mover su personaje en cualquier dirección. Al año siguiente, Konami introdujo Scramble, un shooter de desplazamiento lateral con desplazamiento forzado. Fue el primer shooter de desplazamiento que ofrecía múltiples niveles diferentes. Tempest, de Atari, publicado en 1981, fue uno de los primeros "tube shooters" y un intento temprano de incorporar una perspectiva tridimensional a un juego de disparo. Tempest en última instancia

terminó influyendo en la mayoría de los "rail shooters". Al mismo tiempo aparecieron juegos de disparo de desplazamiento vertical. Xevious (publicado en 1982) es citado frecuentemente como el primer "vertical shooter", y, aunque estuvo de hecho precedido por varios otros juegos de desplazamiento vertical, fue el más influyente. Xevious fue también el primero en representar convincentemente paisajes en oposición a escenarios puramente de ciencia ficción. Mientras Asteroids (1979) permitía al jugador rotar la nave espacial, el grandemente aclamado Robotron 2084, de 1984, fue el más influyente sobre subsiguientes shooters multidireccionales. Space Harrier, de Sega, un rail shooter publicado en 1985, abrió un nuevo campo gráficamente, y su amplia variedad de escenarios a lo largo de múltiples niveles dio a los jugadores un aliciente más aparte de las puntuaciones.

1985 presenció también el lanzamiento de Gradius, de Konami. Gradius daba al jugador mayor control sobre las opciones de armamento, introduciendo de este modo otro elemento de estrategia. El juego también introducía la necesidad para el jugador de memorizar los niveles con el objeto de conseguir cualquier tipo de éxito. Gradius, con su icónico protagonista, definió el shoot 'em up de desplazamiento lateral y generó una serie que se prolongó durante varias secuelas. El siguiente año vio la aparición de una de las series de vanguardia de Sega con su juego Fantasy Zone. El juego fue aclamado por sus gráficos y su escenario surrealista y por el protagonista, Opa-Opa, que fue durante un tiempo considerado la mascota de Sega. El juego, que tomaba prestado el dispositivo de Defender que permitía al jugador controlar la dirección del vuelo, fue, junto con el anterior Twinbee (1985), un arquetipo del subgénero "cute 'em up". R-Type, otro aclamado shoot 'em up de desplazamiento horizontal, publicado en 1987 por Irem, empleaba un desplazamiento de pantalla más lento de lo habitual, con niveles de mayor dificultad que reclamaban estrategias metódicas. Raiden, de 1990, fue el comienzo de la aparición de otra largamente aclamada serie en este período.

Imagen 2.3.2. R-Type. Extraída de http://www.iphoneland.it/wp-content/uploads/r-type_arcade1.jjpg

Shoot 'em ups tales como Ikari Warriors, de SNK (1986), que presentaban a personajes a pie, en lugar de una nave espacial, se hicieron populares a mediados de los años 80 con el nacimiento de películas de acción como Rambo: First Blood Part II. El primer juego de este tipo se desconoce, pero el primer ejemplo influyente fue Commando, de Data East, publicado en 1985. Commando desencadenó comparaciones con Rambo, y en efecto la crítica contemporánea consideró los temas militares y los protagonistas similares a Rambo o a los personajes interpretados por Arnold Schwarzenegger como prerrequisitos para un shoot 'em up, como oposición a los videojuegos de acción-aventura. Contra, de 1989 (del cual fue publicada en Europa una versión modificada con el nombre de Probotector) fue particularmente aclamado por sus objetivos multidireccionales y la posibilidad de jugar dos personas a la vez de forma cooperativa. Sin embargo, a principios de los años 90 y con la popularidad de las videoconsolas de 16 bits, el género scrolling shooter se vio saturado, con desarrolladores luchando por conseguir que sus juegos fuesen destacados (una excepción fue el inventivo Gunstar Heroes, por Treasure).

Imagen 2.3.3. Commando. Extraída de http://images.11888freeonlinegames.com/download-games/commando/commando-game.jpg

Evolución hacia los “bullet hell” y popularización

A principios de los años 90 surgió un nuevo tipo de shoot 'em up, llamado “bullet hell”, “manic shooter” o “maniac shooter”. Este tipo de juego requería que el jugador esquivase abrumadoras cantidades de proyectiles enemigos y le reclamaba reacciones aún más rápidas. Los juegos bullet hell nacieron de la necesidad de los desarrolladores de shoot 'em up bidimensionales de competir con la emergente popularidad de los juegos tridimensionales: se pretendía impresionar a los jugadores con enormes cantidades de misiles sobre la pantalla. Batsugun (1993), de Toaplan, proporcionó la plantilla prototípica para esta nueva raza de videojuegos, y la compañía Cave (formada por antiguos empleados de Toaplan, incluyendo al creador principal de Batsugun Tsuneki Ikeda, después de que la segunda quebrase) creó el juego típico de este género, DonPachi, en 1995. Los “manic shooter” marcaron otro hito a partir del cual el género shoot 'em up empezó a dirigirse a jugadores más dedicados. Juegos tales como Gradius habían sido más difíciles que Space Invaders o Xevious,[38] pero los juegos “bullet hell” eran aún más introvertidos y dedicados a los fans del

género que buscaban mayores desafíos. Mientras que los juegos "shooter" que mostraban protagonistas a pie se trasladaron en su mayoría a los géneros basados en tres dimensiones, series populares y de largo recorrido como Contra y Metal Slug continuaron recibiendo nuevas secuelas. Los "rail shooters" raramente han sido publicados en el nuevo milenio, y sólo Rez y Panzer Dragoon Orta han conseguido reconocimiento de culto.

Imagen 2.3.4. Contra. Extraída de http://www.the-isb.com/images/Contra02.jpg

Radiant Silvergun (1998), de Treasure, introducía un elemento narrativo en el género. Obtuvo numerosas aclamaciones por parte de la crítica por su diseño refinado, aunque nunca se publicó fuera de Japón y continúa siendo un objeto muy buscado por los coleccionistas. Su sucesor, Ikaruga (2001), presentaba mejores gráficos y fue de nuevo aclamado como uno de los mejores juegos del género. A diferencia de Radiant Silvergun, se publicó también en la videoconsola Xbox Live Arcade. El género ha sufrido un poco con el resurgimiento de los servicios en línea de la Xbox 360 y la Wii, mientras que en Japón los shoot 'em up de arcades mantienen un nicho de popularidad profundamente arraigada. Geometry Wars: Retro Evolved se publicó en Xbox Live Arcade en 2005 y destacó especialmente sobre las varias reediciones y videojuegos casuales disponibles en el servicio. Sin embargo, y a pesar del continuado atractivo del género para un grupo entusiástico de jugadores, los desarrolladores de juegos shoot 'em up están cada vez más acuciados financieramente por el poder de las videoconsolas y sus géneros, reclamados por el público.

Capítulo 3

Planificación

3.1. Planificación temporal

La planificación del proyecto establecida originalmente ha coincidido en gran medida con los tiempos esperados. Aunque durante el desarrollo, algunas tareas se han visto retrasadas debido a la aparición de nuevas ideas y problemas surgidos.

3.1.1. Fases del desarrollo

En esta sección detallamos el desglose de las tareas.

Descripción del proyecto: 1 día.

Diseño: 37 días.

- **Gráficos:** 27 días.
  - **Búsqueda de ideas:** 20 días.
  - **Historia:** 1 día.
  - **Menús:** 1 día.
  - **Personajes principales:** 1 día.
  - **Fases:** 1 día.
  - **Interfaz:** 1 día.
  - **Objetos:** 1 día.
  - **Enemigos:** 1 día.
- **Sonido:** 2 días.
  - **Búsqueda de música:** 1 día.
  - **Búsqueda de efectos:** 1 día.
- **Software:** 8 días.
  - **Casos de uso:** 1 día.
  - **Clases conceptuales:** 2 días.
  - **Contratos de operaciones:** 5 días.

Desarrollo: 112 días.

- **Modelado 2D:** 28 días.
  - **Aprendizaje GIMP:** 1 día.
  - **Menús:** 4 días.
  - **Personajes principales:** 10 días.
  - **Fases: 4** días.
  - **Objetos: 2** días.
  - **Enemigos:** 6 día.
  - **Interfaz:** 1 día.
- **Sonido:** 2 días.
  - **Edición con Audacity:** 2 días.
- **Codificación:** 82 días.
  - **Aprendizaje SDL:** 1 día.
  - **Clase Teclado:** 1 día.
  - **Clase Imagen:** 1 día.
  - **Clase Menú:** 6 día.
  - **Clase Animación:** 1 día.
  - **Clase Juego:** 40 días.
  - **Clase Control Animación:** 1 día.
  - **Clase Personaje:** 5 días.
  - **Clase Autómata:** 5 días.
  - **Clase Fase:** 3 días.
  - **Clase Colisión:** 5 días.
  - **Clase Bala:** 3 días.
  - **Clase Objeto:** 1 días.
  - **Clase Enemigo:** 7 días.
  - **Clase Música:** 1 día.
  - **Clase Sonido:** 1 día.
- **Sonido:** 2 días.
  - **Edición con Audacity:** 2 días.

Pruebas y Mejoras: 5 días.

Unificación y depuración del código: 2 días.

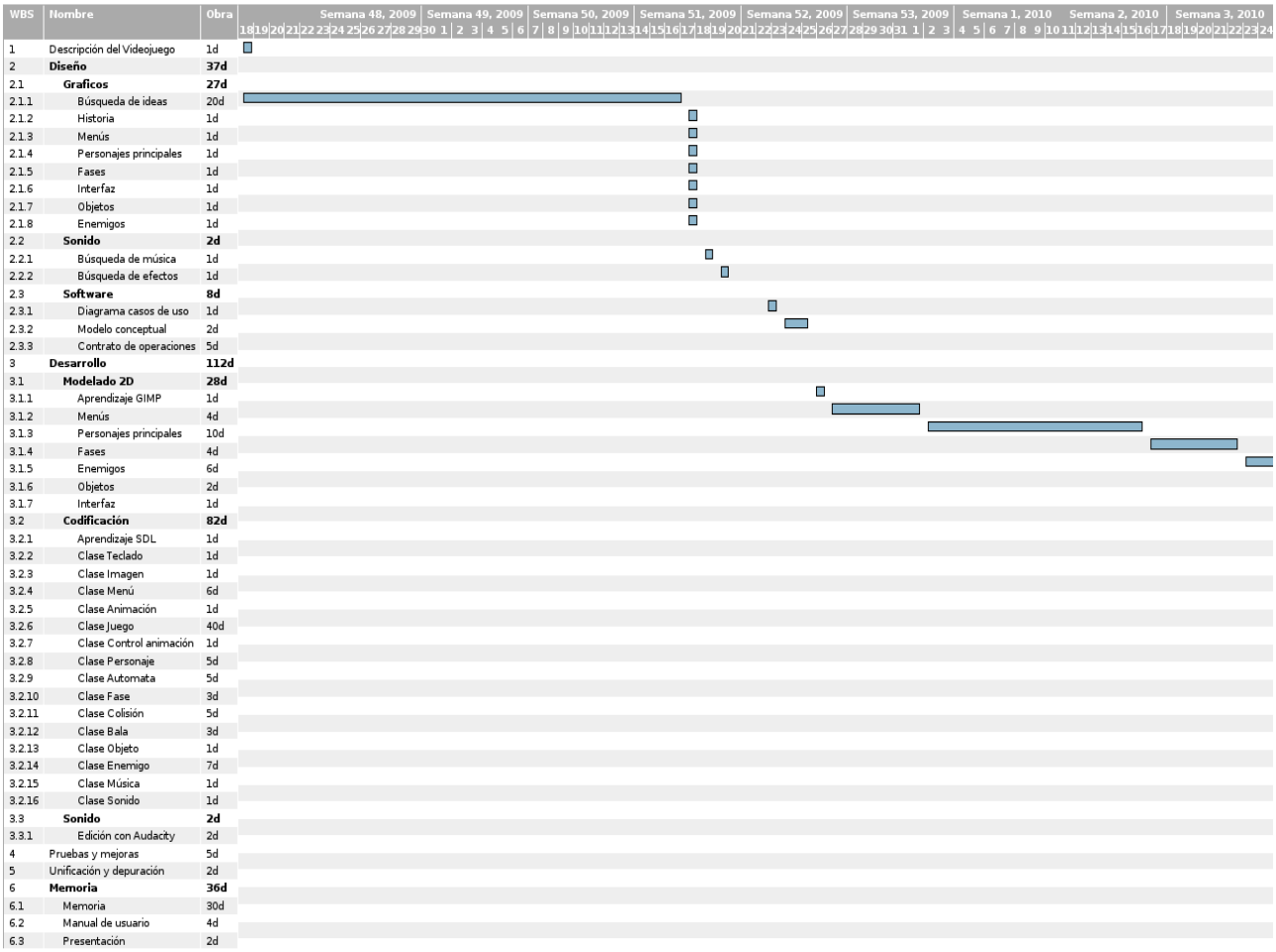
Memoria: 36 días.

- **Memoria:** 30 días.
- **Manual de usuario:** 2 días.
- **Presentación:** 4 días.

Tiempo total empleado en el desarrollo: 193 días.

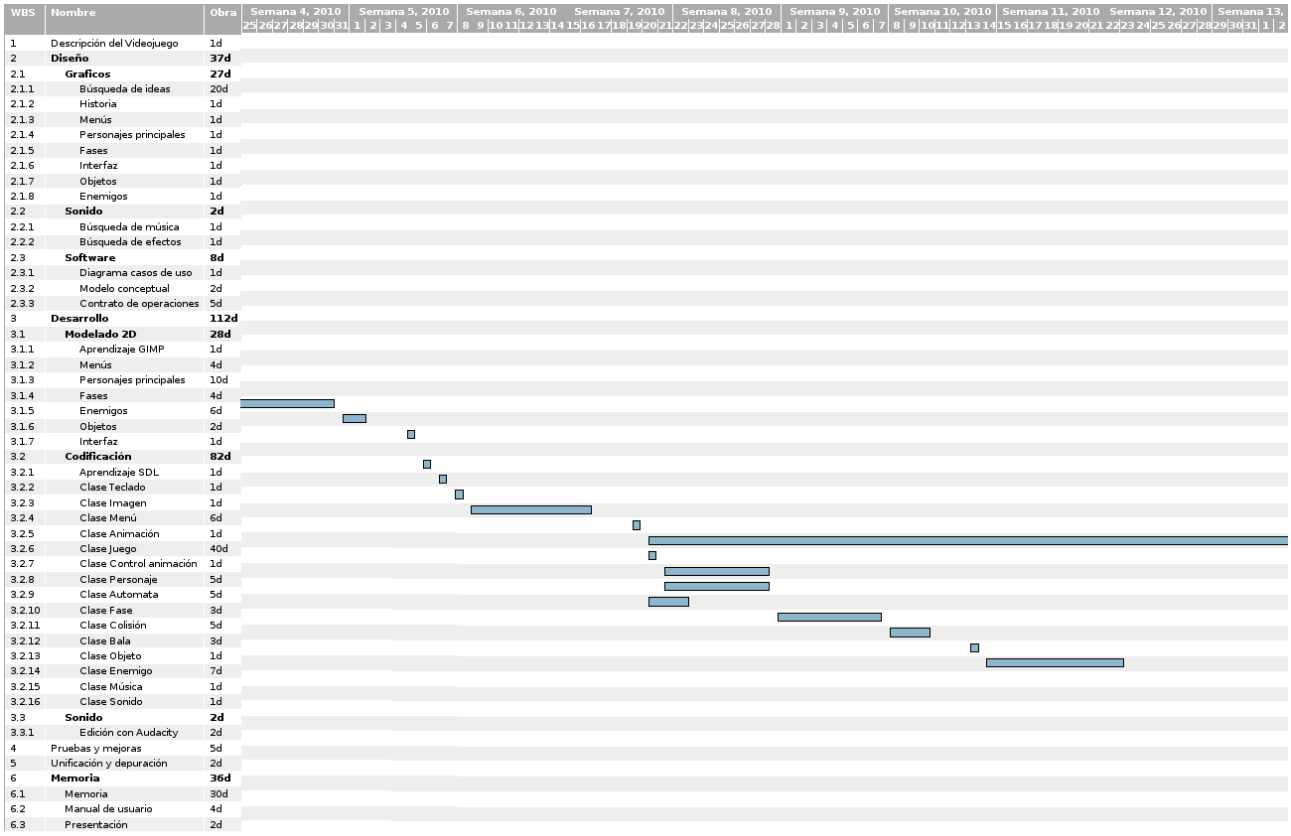
El tiempo planificado es de 6-7 meses aproximadamente que coincide con el trabajo realizado.

A continuación el diagrama de Gantt.



WBS	Nombre	Obra
1	Descripción del Videojuego	1d
2	**Diseño**	**37d**
2.1	**Graficos**	**27d**
2.1.1	Búsqueda de ideas	20d
2.1.2	Historia	1d
2.1.3	Menús	1d
2.1.4	Personajes principales	1d
2.1.5	Fases	1d
2.1.6	Interfaz	1d
2.1.7	Objetos	1d
2.1.8	Enemigos	1d
2.2	**Sonido**	**2d**
2.2.1	Búsqueda de música	1d
2.2.2	Búsqueda de efectos	1d
2.3	**Software**	**8d**
2.3.1	Diagrama casos de uso	1d
2.3.2	Modelo conceptual	2d
2.3.3	Contrato de operaciones	5d
3	**Desarrollo**	**112d**
3.1	**Modelado 2D**	**28d**
3.1.1	Aprendizaje GIMP	1d
3.1.2	Menús	4d
3.1.3	Personajes principales	10d
3.1.4	Fases	4d
3.1.5	Enemigos	6d
3.1.6	Objetos	2d
3.1.7	Interfaz	1d
3.2	**Codificación**	**82d**
3.2.1	Aprendizaje SDL	1d
3.2.2	Clase Teclado	1d
3.2.3	Clase Imagen	1d
3.2.4	Clase Menú	6d
3.2.5	Clase Animación	1d
3.2.6	Clase Juego	40d
3.2.7	Clase Control animación	1d
3.2.8	Clase Personaje	5d
3.2.9	Clase Automata	5d
3.2.10	Clase Fase	3d
3.2.11	Clase Colisión	5d
3.2.12	Clase Bala	3d
3.2.13	Clase Objeto	1d
3.2.14	Clase Enemigo	7d
3.2.15	Clase Música	1d
3.2.16	Clase Sonido	1d
3.3	**Sonido**	**2d**
3.3.1	Edición con Audacity	2d
4	Pruebas y mejoras	5d
5	Unificación y depuración	2d
6	**Memoria**	**36d**
6.1	Memoria	30d
6.2	Manual de usuario	4d
6.3	Presentación	2d

Imagen 3.1. Diagrama de Gantt - Parte 1



WBS	Nombre	Obra
1	Descripción del Videojuego	1d
2	**Diseño**	**37d**
2.1	**Graficos**	**27d**
2.1.1	Búsqueda de ideas	20d
2.1.2	Historia	1d
2.1.3	Menús	1d
2.1.4	Personajes principales	1d
2.1.5	Fases	1d
2.1.6	Interfaz	1d
2.1.7	Objetos	1d
2.1.8	Enemigos	1d
2.2	**Sonido**	**2d**
2.2.1	Búsqueda de música	1d
2.2.2	Búsqueda de efectos	1d
2.3	**Software**	**8d**
2.3.1	Diagrama casos de uso	1d
2.3.2	Modelo conceptual	2d
2.3.3	Contrato de operaciones	5d
3	**Desarrollo**	**112d**
3.1	**Modelado 2D**	**28d**
3.1.1	Aprendizaje GIMP	1d
3.1.2	Menús	4d
3.1.3	Personajes principales	10d
3.1.4	Fases	4d
3.1.5	Enemigos	6d
3.1.6	Objetos	2d
3.1.7	Interfaz	1d
3.2	**Codificación**	**82d**
3.2.1	Aprendizaje SDL	1d
3.2.2	Clase Teclado	1d
3.2.3	Clase Imagen	1d
3.2.4	Clase Menú	6d
3.2.5	Clase Animación	1d
3.2.6	Clase Juego	40d
3.2.7	Clase Control animación	1d
3.2.8	Clase Personaje	5d
3.2.9	Clase Automata	5d
3.2.10	Clase Fase	3d
3.2.11	Clase Colisión	5d
3.2.12	Clase Bala	3d
3.2.13	Clase Objeto	1d
3.2.14	Clase Enemigo	7d
3.2.15	Clase Música	1d
3.2.16	Clase Sonido	1d
3.3	**Sonido**	**2d**
3.3.1	Edición con Audacity	2d
4	Pruebas y mejoras	5d
5	Unificación y depuración	2d
6	**Memoria**	**36d**
6.1	Memoria	30d
6.2	Manual de usuario	4d
6.3	Presentación	2d

Imagen 3.2 Diagrama de Gantt - Parte 2

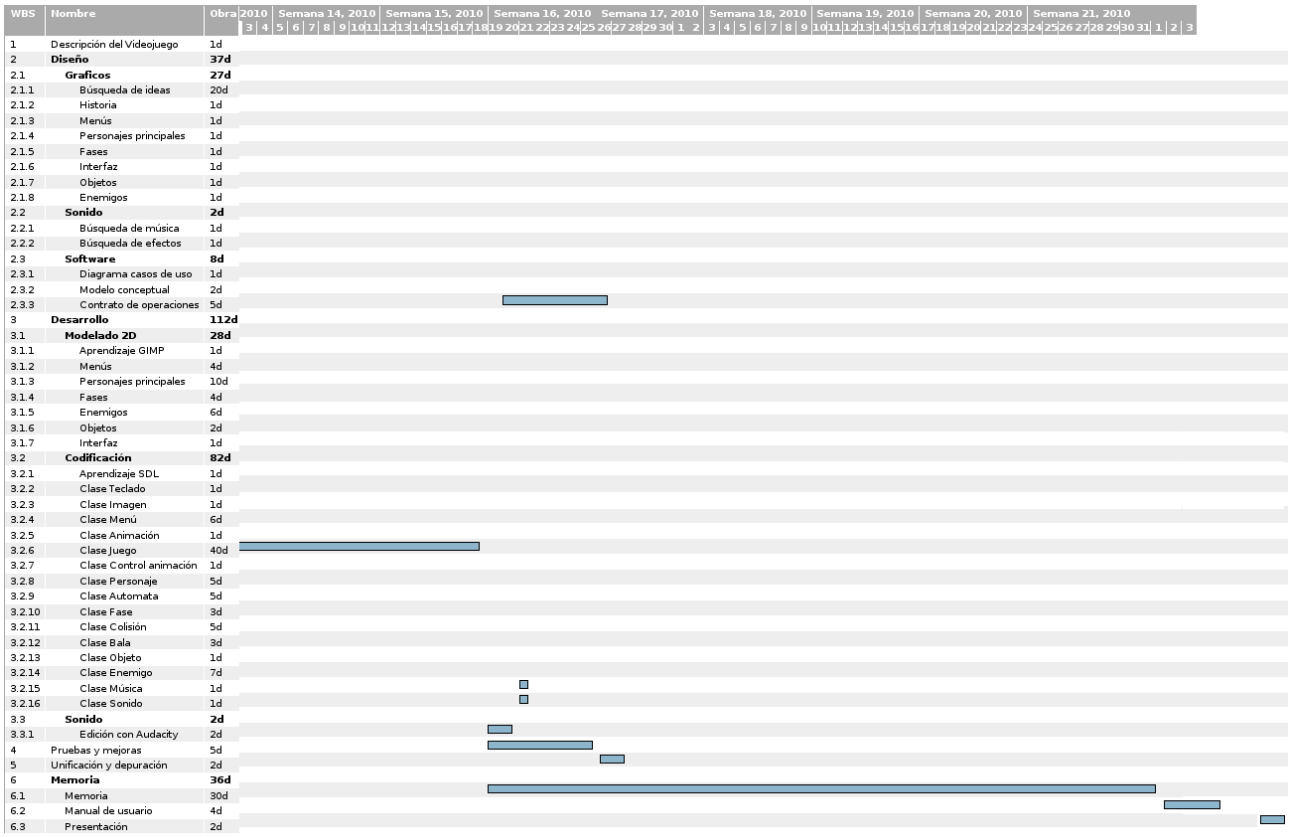


Imagen 3.3. Diagrama de Gantt - Parte 3

Capítulo 4

Diseño

En este capítulo tratamos todos los aspectos relacionado con el diseño y creatividad del videojuego (historia, gráficos, sonidos y software).

4.1. Historia

La historia de The Most Wanted está dividida en cinco capítulos. En cada una de ellas nuestro personaje debe enfrentarse a un enemigo diferente unido por un destino común, ser el más buscado. Dependiendo del personaje escogido, la historia avanza de forma diferente.

4.1.1. Historia de Billy Banks

Del pasado de Billy Banks se sabe más bien poco. Luchó como soldado del séptimo de caballería en las guerras indias, convirtiéndose en uno de los soldados más respetados del batallón. Cuentan los rumores que acabó con cientos de indios, cometiendo actos atroces contra ellos. Luego desapareció, ocultándose en la llanuras del norte, arrepentido por sus actos.
Durante más de diez años ha vivido entre indios, dándole por muerto. Considerado como un héroe, Billy ha vuelto con el objetivo de dar caza a los más buscados y reivindicar que el es el verdadero peligro del oeste.

Capítulo I

Billy Banks llega a la ciudad con el único objetivo de acabar con "Caracortada", un forajido que después de haber luchado en numerosas batallas indias, se dedica al vandalismo y al asesinato.
Su enemistad viene del pasado, cuando en otra ocasión ya se enfrentaron y aunque fue Billy quien le deformo la cara, aún no esta satisfecho. "Caracortada" quiere venganza pero él será el primero en la quema de Billy.
James "Caracortada", al mando de una de las mayores bandas de extorsión del oeste se ha hecho con la ciudad. No será tarea fácil.

Capítulo II

Billy deberá ir hacia las llanuras del norte donde se esconde un asesino al que hay que dar caza.
Butch es el segundo de la lista de Billy. Su motivo son los actos cometidos en contra de los indios. Y aunque Billy considera al forajido cobarde y sin escrúpulos por sus actos, su principal motivo es la recompensa. Hasta encontrarlo nos toparemos con multitud de forajidos que desean darnos caza.

Capítulo III

Billy decide dar caza a un bandido que se toma la justicia por su cuenta cuya recompensa es de 10000 dolares. Todo cambia cuando se entera que dicho vándalo es Roy Casidy. Un amigo del ejercito que luchó a su lado cometiendo los mismo abusos que él. Aunque la amistad se perdió cuando Billy se fue a las llanuras aún mantienen un gran respeto entre ellos. Es la oportunidad para zanjar discrepancias del pasado. Ninguno de los dos tienen miedo a morir. El oeste es demasiado pequeño para los dos.

Capítulo IV

La popularidad de Billy crece y sus ansias no están saciadas. En esta ocasión, Billy se dirige a

Wildcity. Una ciudad destrozada por la corrupción del sheriff Sulivan. Nada lo une con Billy. Los motivos de Billy van más allá de la simple coincidencia. Considera que la ley debe aplicarse a todos y sino es así, el tendrá que recordarse lo. No será bienvenido en *Wildcity*.

Capítulo V

Sin ningún motivo por el que vivir. Billy busca aferradamente la muerte dando caza a los mayores forajidos, mientras que la recompensa por su cabeza aumenta. Sólo hay un hombre cuya recompensa es mayor que la suya, Toro Salvaje. Prometió nunca jamás matar un indio pero esto es diferente. Billy luchará para proclamarse como el mayor forajido del salvaje oeste.

Fin

Después de esto, Billy desapareció y ningún caza recompensas dio con él.
La recompensa por su muerte ascendió a 50000 dólares. Fue la recompensa más alta del lejano oeste.

4.1.2. Historia de Toro Salvaje

Toro Salvaje fue un guerrero Sioux muy respetado de su tribu. Su vida cambio cuando el hombre blanco invadió sus tierras, arrasando sus cosechas y matando a su familia. Él fue uno de los únicos supervivientes de las guerras.
Desde entonces su único objetivo es vengar la muerte de su pueblo, acabar con todo aquel que participó en la matanza de sus seres queridos.

Capítulo I

Toro Salvaje va en busca de James, forajido que después de haber luchado en numerosas batallas indias, se dedica al vandalismo y al asesinato.
El deseo de Toro es saciar su venganza acabando con los soldados que participaron en la matanza. Toro deberá tener máximo cuidado con "Caracortada".

Capítulo II

Una mañana siendo joven mientras pescaba con su hermano mayor Hoja amarilla, apareció un grupo de bandidos que solía dar caza a indios indefensos.
Después de una larga persecución a caballo Toro logró escapar pero su hermano no. Hoja Amarilla murió a manos de Butch, un loco sanguinario sin escrúpulos.
Toro juró venganza y ahora va en su busca.

Capítulo III

Toro se dirige a la ciudad donde se oculta Roy Casidy, excombatiente de las guerras indias. Roy lideró la primera oleada contra su tribu. Ahora Toro quiere devolverle el golpe. No será fácil entrar en la ciudad.

Capítulo IV

La popularidad de Toro crece y sus ansias no están saciadas. En esta ocasión, Toro se dirige a *Wildcity*, una ciudad destrozada por la corrupción del sheriff Sulivan. Su motivo es acabar con el sheriff, el cual ha cometido actos atroces contra su tribu. Sulivan tiene como pasatiempos colgar indios rebeldes por toda su ciudad para infundir miedo a sus ciudadanos. Es algo que Toro no puede permitir.

Capítulo V

Sin ningún familiar con vida. Toro busca aferradamente la muerte de todos los implicados en su tragedia. Por suerte, un tal Billy Banks ex soldado asesino de cientos de indios Sioux han puesto precio a su cabeza. Toro desea su muerte, Billy aceptará el reto.

Fin

Después de esto, Toro se dirigió al norte alejándose del hombre blanco y de sus costumbres. La recompensa por su muerte ascendió a 50000 dólares. Fue la recompensa más alta que ofrecieron por un indio en el lejano oeste.

4.2. Gráficos

Los elementos gráficos que componen el juego están creados con la herramienta Gimp, dibujados con una tableta gráfica con un diseño muy sencillo.
The Most Wanted consta de los siguientes diseños en 2D:

4.2.1. Modelo del personaje principal

Billy Banks: Diseño de los diferentes movimientos del vaquero principal con sus respectivas armas de fuego. Formado por las siguientes animaciones:

- Animación andar recto.
- Animación andar hacia la derecha.
- Animación andar hacia la izquierda.
- Animación morir.
- Animación cuando se pasa con éxito un nivel.

Imagen 4.1. Sprite del personaje principal: Billy Banks

Toro Salvaje: Diseño de los diferentes movimientos del indio principal con sus respectivas armas de fuego. Posee las mismas animaciones que Billy Banks.

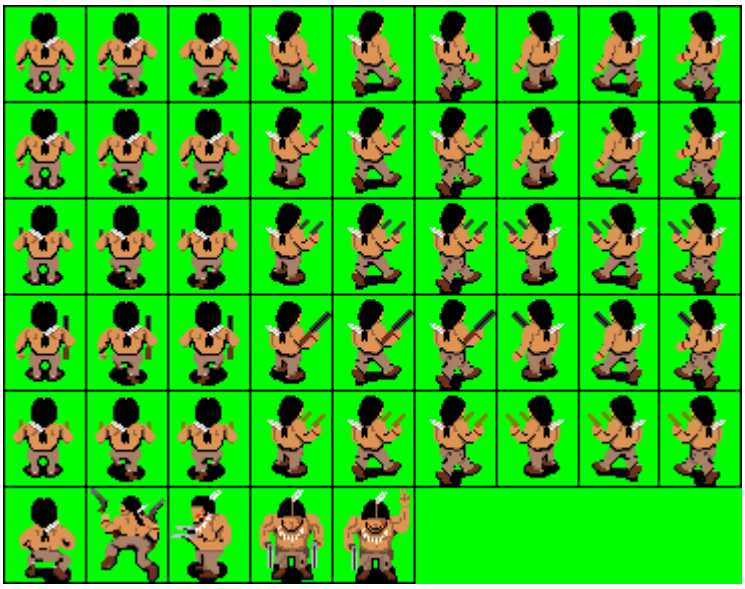


Imagen 4.2. Sprite del personaje principal: Toro Salvaje

Billy Banks a caballo: Diseño de los diferentes movimientos del vaquero principal montando a caballo y del caballo. Formado por las siguientes animaciones:

- Animación andar recto a caballo.
- Animación andar hacia la derecha a caballo.
- Animación andar hacia la izquierda a caballo.
- Animación el caballo muere.
- Animación el cabalga sólo.

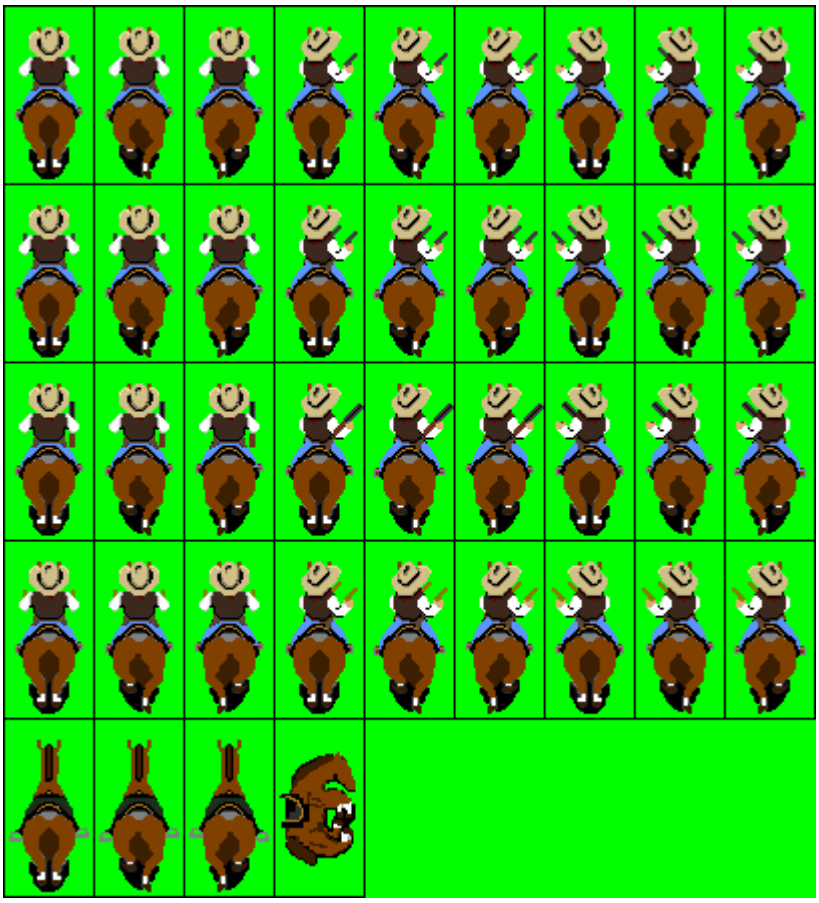


Imagen 4.3. Sprite de Billy Banks a caballo

Toro Salvaje a caballo: Diseño de los diferentes movimientos del indio principal montando a caballo y del caballo.

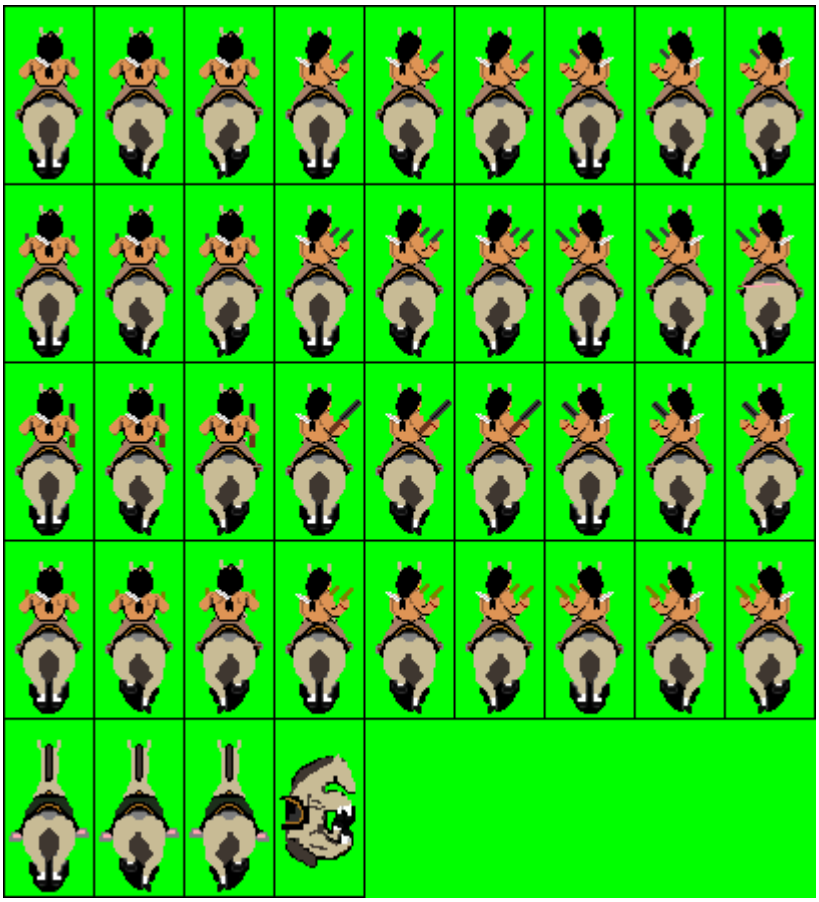


Imagen 4.4. Sprite de Toro Salvaje a caballo

4.2.2. Modelo de Enemigos

Diseño de los movimientos de los distintos enemigos que aparece en el videojuego.

Se han diseñado 9 modelos distintos de enemigos comunes y 5 enemigos finales.

Constan de las siguientes animaciones:

Enemigo 1:

- Animación andar.
- Animación morir.

Imagen 4.5. Sprite Enemigo 1

Enemigo 2:

- Animación andar recto.
- Animación andar hacia la derecha.
- Animación andar hacia la izquierda.
- Animación andar de espaldas.
- Animación morir.

Imagen 4.6. Sprite Enemigo 2

Enemigo 3:

Las mismas que el Enemigo 2.

Imagen 4.7. Sprite del Enemigo 3

Enemigo 4:

- Animación andar recto.
- Animación andar hacia la derecha.
- Animación andar hacia la izquierda.
- Animación saltar derecha.
- Animación saltar izquierda.
- Animación morir.

Imagen 4.8. Sprite Enemigo 4

Enemigo 5:

- Animación caer.
- Animación saltar derecha.
- Animación saltar izquierda.
- Animación morir.

Imagen 4.9. Sprite Enemigo 5

Enemigo 6:

- Animación andar.
- Animación disparar.
- Animación morir.

Imagen 4.10. Sprite Enemigo 6

Enemigo 7:

- Animación andar.
- Animación disparar.
- Animación morir.

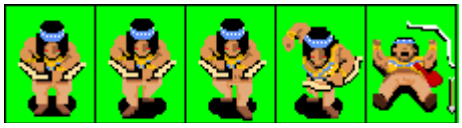


Imagen 4.11. Sprite Enemigo 7

Enemigo 8:

- Animación andar.
- Animación disparar.
- Animación morir.

Imagen 4.12. Sprite Enemigo 8

Enemigo ventana:

- Disparar hacia abajo.
- Disparar hacia arriba.

Imagen 4.13. Sprite Enemigo ventana

Enemigos finales:

- Animación andar.
- Animación morir.

Imagen 4.14. Sprite Enemigo final Fase I

Imagen 4.15. Sprite Enemigo final Fase II

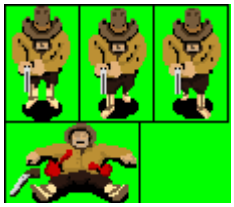


Imagen 4.16. Sprite Enemigo final Fase III

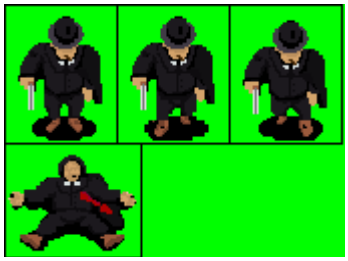


Imagen 4.17. Sprite Enemigo final Fase IV

Imagen 4.18. Sprite Enemigo fina Fase V

Imagen 4.19. Sprite Enemigo final Fase V

4.2.3. Modelo de Objetos

Barril:

Dentro de los barriles se ocultan distintos tipos de objetos y armas.

Dinero:

Al coger un saco de dinero nuestra puntuación incrementará 25 puntos.

Proyectiles:

- Bala

  Es el tipo de bala que utiliza el revólver, los dos revólveres y la escopeta.

  Cada bala quita una unidad de vida.

- Flecha

  Lanzada desde los arcos de los enemigos indios. Quita dos unidades de vida.

- Dinamita

  Explosivo utilizado por un tipo de enemigo. Quita dos unidades de vida.

- Cuchillos

  Arma utilizada por el enemigo final de la primera fase.

- Super bala.

  Es el tipo de bala que utiliza los dos revólveres especiales. Quita dos unidades de vida.

Armas:

- Revolver

  Arma predeterminado que posee el personaje principal.

- Dos revólveres

  Con este arma podremos disparar dos balas a la vez, destruyendo más rápido a los enemigos.

- Escopeta

  Este arma dispara cinco balas de golpe. Su destrucción es mucho mayor.

- Revólveres especiales.

Con este arma podremos disparar dos super balas a la vez, destruyendo más rápido a los enemigos.

Vida:

Se representa con un sombrero de cowboy. Se incrementará una vida cada vez que nos encontremos con una.

Montar a caballo:

Se representa con la silueta de un cowboy montado en un caballo. Al coger este objeto aparecerá un caballo en el cual, nuestro personaje se montará y estará protegido ante cualquier disparo.

Herradura:

Es una herradura de oro. Nuestro personaje gozará de inmunidad durante el tiempo que posea este objeto.

Botas:

Botas de cowboy. La velocidad del persona se incrementará durante el tiempo que poseamos este objeto.

Calavera:

Esqueleto de un búfalo. Al coger este objeto acabarás de golpe con todos los enemigos que estén en pantalla.

4.2.4. Interfaz

Menú Principal

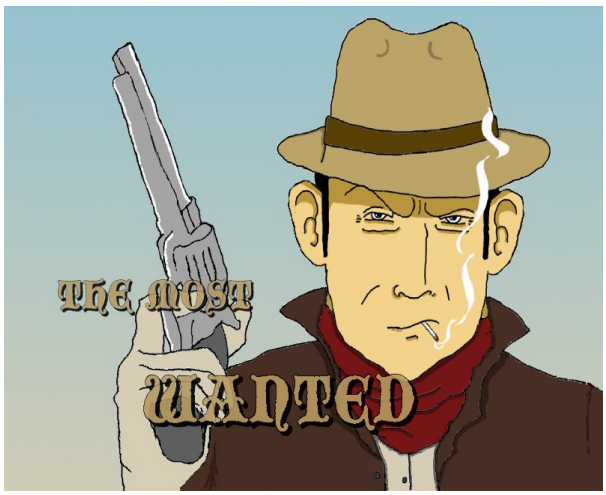


Imagen 4.20. Pantalla inicial

Para acceder al menú principal deberemos pulsar una tecla cualquiera estando en la pantalla inicial del juego.

Dentro del menú principal mediante los cursores de dirección podremos elegir una de las siguientes opciones:

Modo individual: Para comenzar la partida en modo un jugador. Si elegimos esta opción accederemos al menú de elección del personaje-1jugador.

Modo Dos jugadores: Para comenzar la partida en modo dos jugadores. Si elegimos esta opción accederemos al menú de elección del personaje-2jugadores.

Opciones: Para acceder al menú opciones.

Récords: Para acceder al registro de las máximas puntuaciones.

Créditos: Para acceder a los créditos del juego.

Salir: Para salir del juego.

Imagen 4.21. Menú principal

Elección del personaje-1player

En este menú se elegirá el personaje que más nos agrade para comenzar la partida. Si por el contrario deseamos regresar hacia el menú principal elegiremos la opción menú principal.

El personaje no se podrá cambiar una vez comenzada la partida.

Imagen 4.22. Menú elegir el personaje - 1 jugador

Elección del personaje-2player

En este menú se elegirá el personaje que controlará cada jugador. Si por el contrario deseamos regresar hacia el menú principal elegiremos la opción menú principal.

El personaje no se podrá cambiar una vez comenzada la partida.

Imagen 4.23. Menú elegir personaje - 2 jugadores

Opciones

En el menú de opciones podremos ver cuales son las teclas para poder jugar, activar o desactivar la música, activar o desactivar los efectos de sonido o por el contrario volver al menú principal.

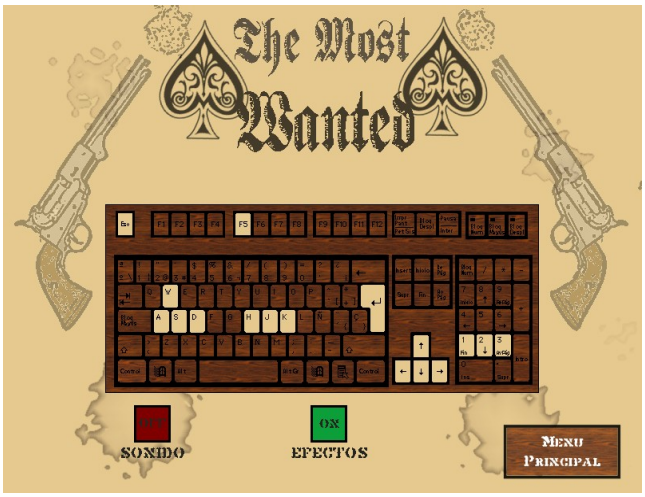


Imagen 4.24. Menú Opciones

Récords

Sección que nos informará de las mejores puntuaciones conseguidas. Para volver atrás simplemente elegir la única opción disponible, menú principal.

Imagen 4.25. Récords

Créditos

Esta sección únicamente nos informará del autor del juego. Para volver atrás simplemente elegir la única opción disponible; menú principal.

Imagen 4.26. Créditos

Salir

Al salir desde el menú principal, nos aparecerá la siguiente ventana para asegurarnos si deseamos salir o no.

Imagen 4.27. Salir

Diagrama de navegación

La siguiente imagen nos muestra un diagrama del funcionamiento de los menús:

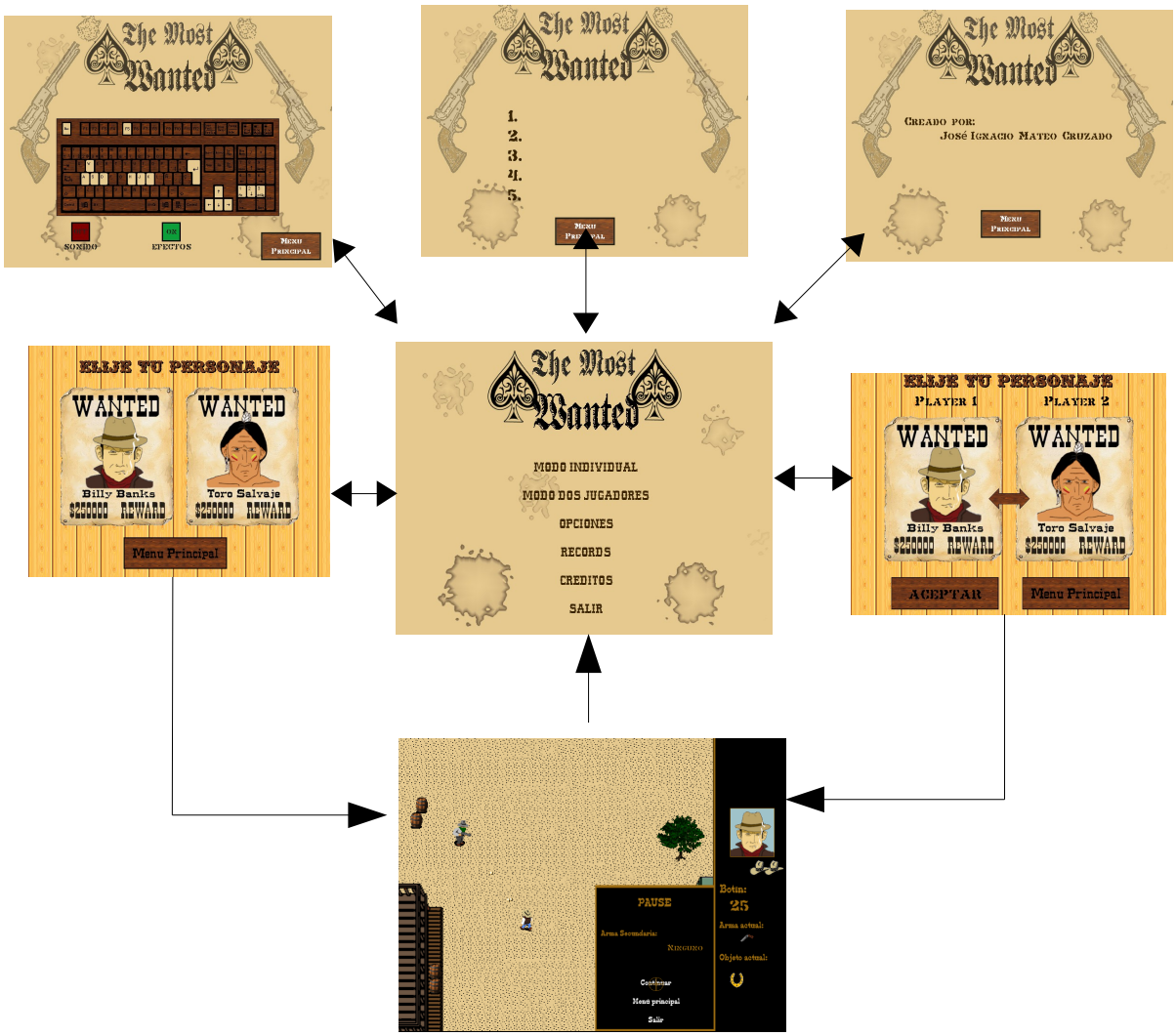


Imagen 4.28. Diagrama de navegación

Interfaz durante la partida

Esta situada en la parte derecha de la pantalla y está formada por los siguientes elementos:

Personaje: Muestra un pequeño dibujo con la cara del personaje elegido como si de una foto se tratase.

Puntuación: Muestra la puntuación acumulada a lo largo de la partida.

Vidas: Muestra el número de vidas del que disponemos. El jugador comienza la partida con dos vidas pudiendo acumular hasta un máximo de cinco.

Arma actual: Arma que está utilizando.

Objeto actual: Objeto que posee.

Arma secundaria: Arma secundaria que posees. Para cambiar el arma, pulsamos Esc y luego elegimos el arma secundaria.

Imagen 4.29. Interfaz durante la partida

4.2.5. Fases

The Most Wanted consta de cinco fases o niveles, cada una de ellas en un escenario diferente, donde encontraremos enemigos, objetos y dificultades diferentes.
Cada una de las fases comienza con una introducción, en la que nos mostrará cual es el enemigo final de la fase y la relación de este con nuestro personaje.

Imagen 4.30. Introducción de la primera fase

Tras ésta introducción comienza la partida. Al terminar la fase con éxito, aparecerá el enemigo final liquidado y la puntuación acumulada. Aceptaremos para continuar con la fase siguiente.
Si conseguimos superar con éxito la última fase, además de la introducción inicial, muestra el final de la historia. Si, por el contrario, perdemos todas nuestras vidas, el juego termina y la puntuación se registra si está entre las cinco más altas.

Fase I

La primera fase se desarrolla en una pequeña ciudad del oeste americano formada por salones, iglesias y casas. La dificultad es pequeña puesto que no aparece gran número de enemigos. El enemigo final es James "Caracortada", un temible enemigo experto en cuchillos.

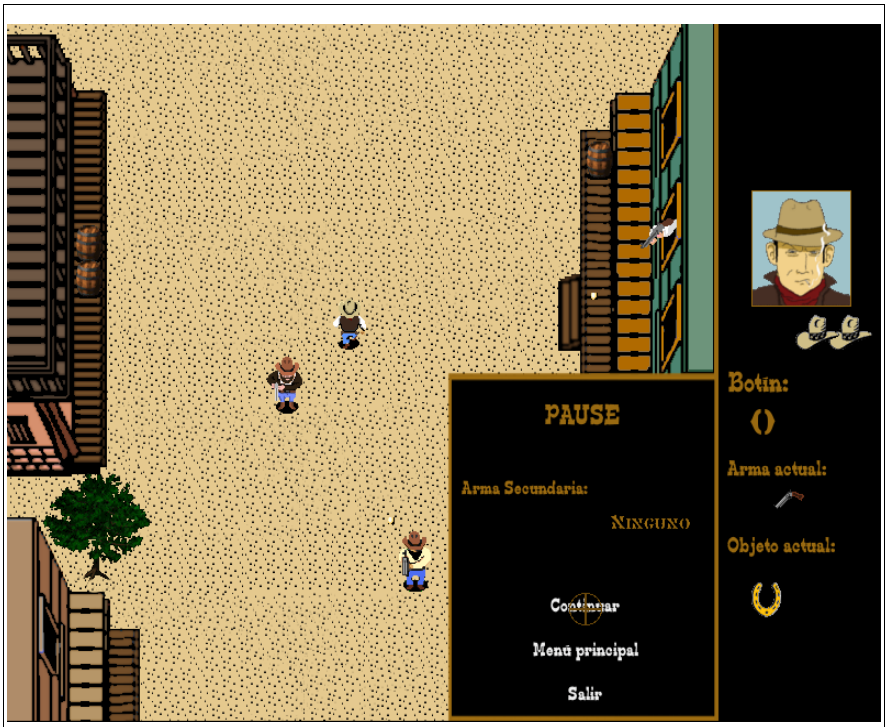


Imagen 4.31. Fase I

Fase II

La segunda fase se desarrolla en el desierto rocoso del suroeste americano donde encontraremos rocas y cactus. Aumenta el número de enemigos que aparecen y disminuye el número de objetos, por lo que la dificultad también es mayor. El enemigo final es Butch, pistolero de un sólo revólver.

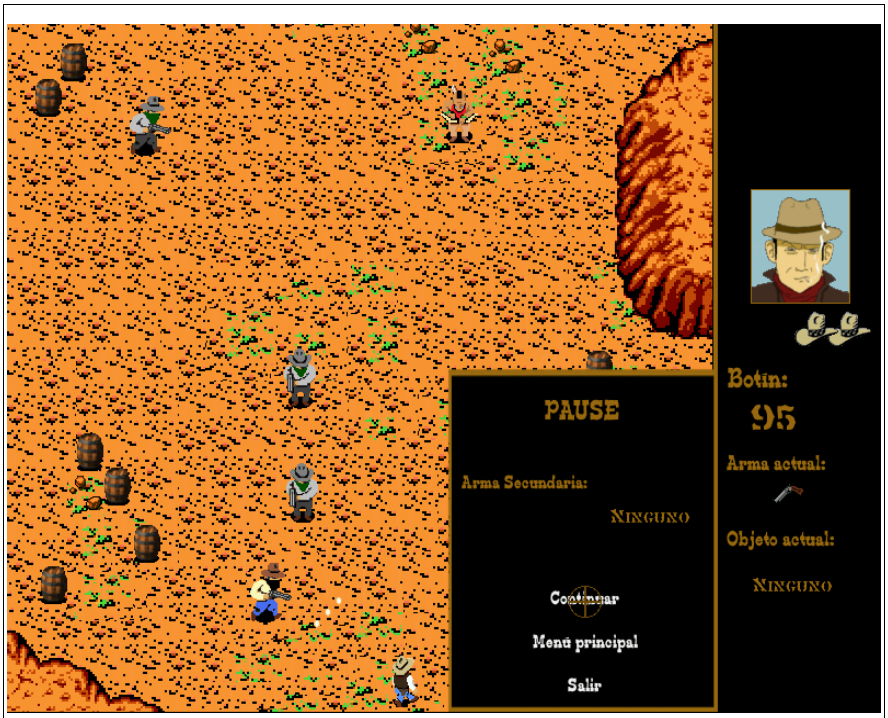


Imagen 4.32. Fase II

Fase III

La tercera fase transcurre en la misma ciudad de la primera siendo en esta ocasión de noche.
Para esta fase he tenido que volver a diseñar cada uno de los elemento que aparecen en ella, puesto que son más opacos que normalmente. Aunque sea la misma ciudad nos encontraremos con objetos y enemigos diferente a la primera. El enemigo final es Cassidy, experto en tiro con escopeta.

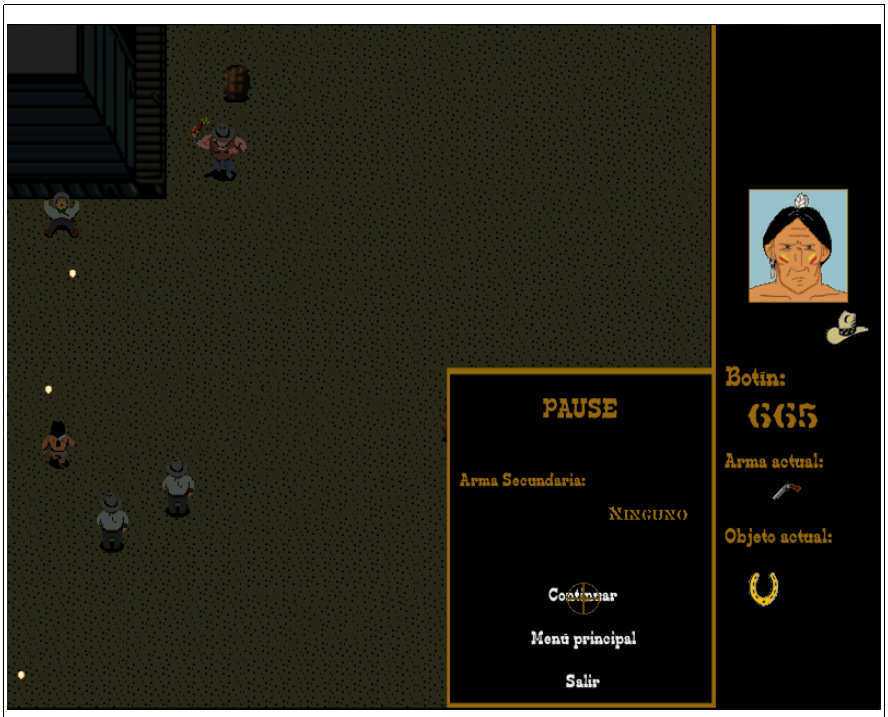


Imagen 4.33. Fase III

Fase IV

La cuarta fase transcurre la acción en *Wildcity* una ciudad gobernada por el sheriff Sulivan.

La ciudad está deteriorada. El número de enemigo es mucho mayor considerándose como la fase más difícil del juego. El enemigo final al que deberemos derrotar es Sulivan, un perfecto pistolero.

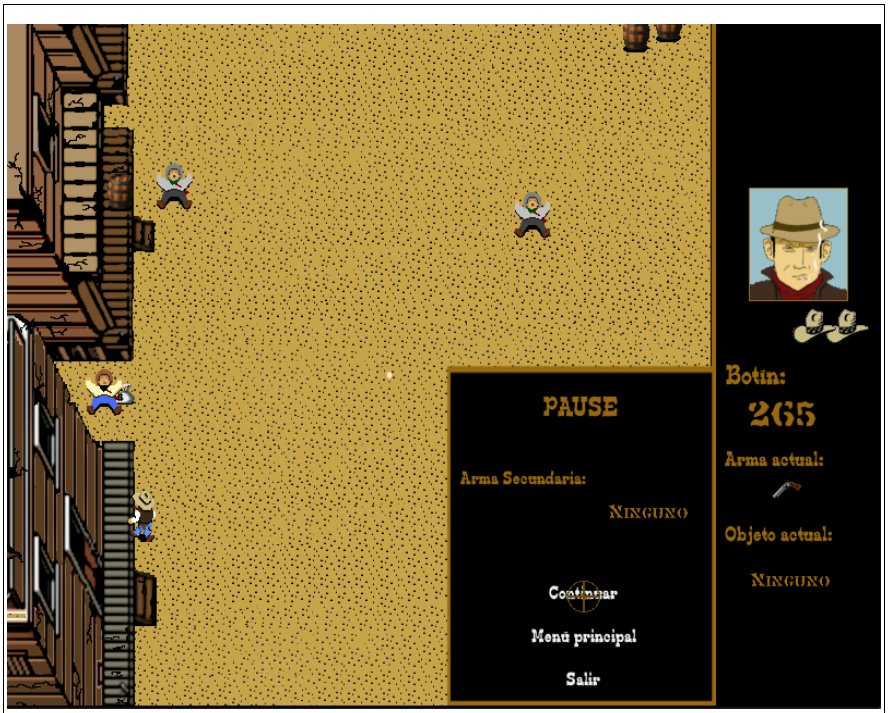


Imagen 4.34. Fase IV

Fase V y VI

La quinta y sexta fase transcurre en el mismo lugar, una aldea india. Sin embargo, según el personaje que elijamos la aldea tendrá diferencias. Los enemigos serán iguales del mismo modo que los objetos. El enemigo final también cambiará dependiendo del personaje.

Imagen 4.35. Fase V - Toro Salvaje

Imagen 4.36. Fase V - Billy

4.3. Sonido

4.3.1. Música

Música del menú

Para el menú, la música escogida es del género "spaguetti western", tan típica en las películas de este género.
La música del menú está compuesta por dos temas que dependiendo del azar sonará uno u otro, así habrá más variedad dando impresión de un videojuego más serio y menos monótono.

Música de las fases

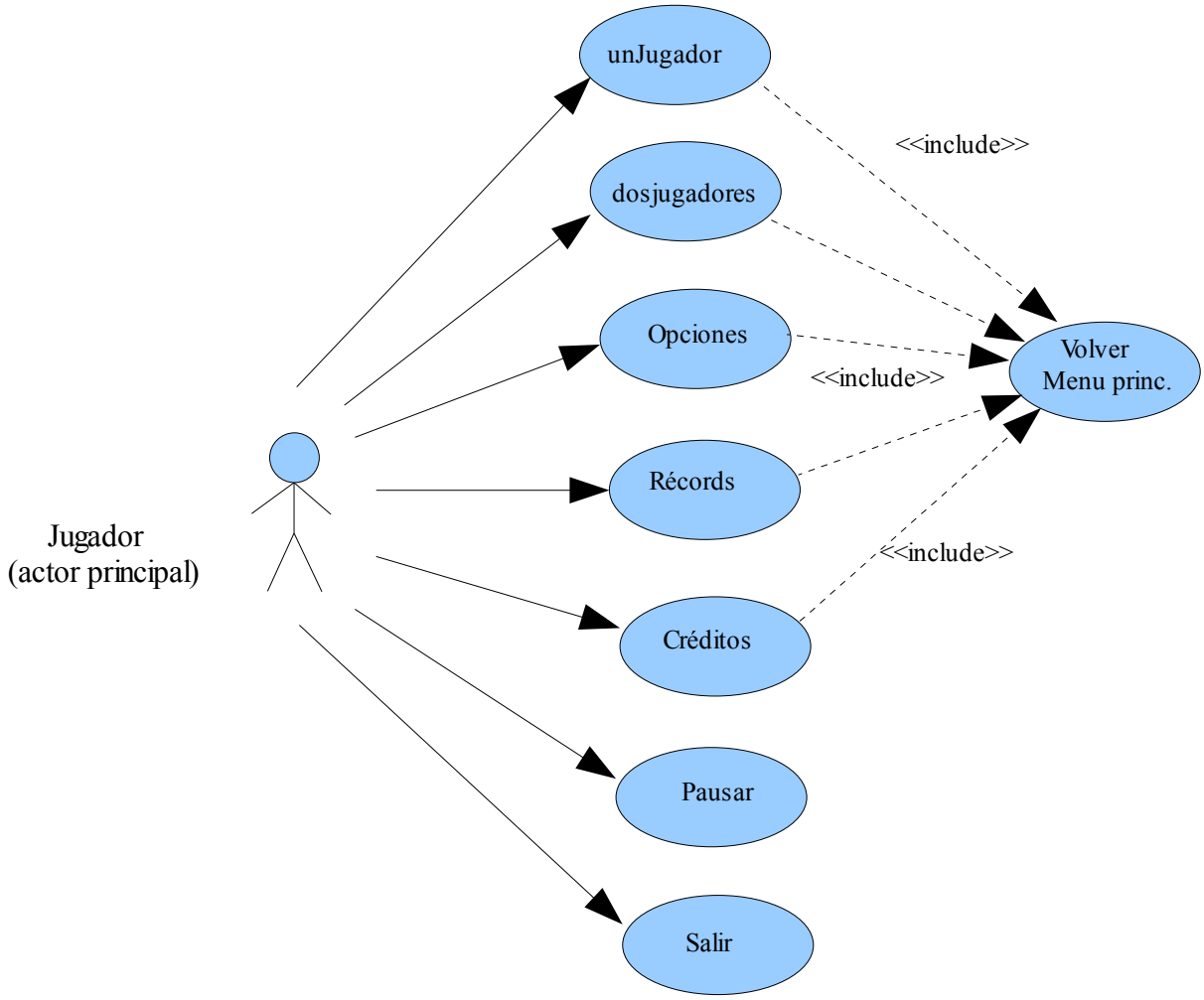
Para ambientar las fases, se ha elegido temas "spaguetti western" y country más animados.
Por cada fase existirá dos o tres temas diferentes para dar variedad a la hora de jugar.

4.3.2. Efectos

- Sonido de disparo.
- Sonido del caballo.
- Sonido al montarse en el caballo.
- Sonido al terminar una fase con éxito.
- Sonido de vida.
- Sonido al conseguir un objeto con tiempo.
- Sonido de explosión de dinamita.
- Sonido Game Over.

4.4. Diseño software

4.4.1. Diagrama de casos de uso



4.4.2. Caso de Uso: Volver Menú principal

Caso de Uso: Volver Menú principal
Nivel: Subfunción
Escenario Principal
1. El jugador elige la opción Menú principal.
2. El sistema devuelve la pantalla de menú principal.
Escenarios Alternativos
*a. El jugador cierra el videojuego.

4.4.3. Caso de Uso: unJugador

Caso de Uso: unJugador
Precondiciones: Ninguna
Postcondiciones: El jugador establece una partida en modo \Un Jugador"
contra la máquina.
Descripción: Partida de un solo jugador.
Resumen: El jugador quiere jugar una partida contra la maquina.
Actores: Jugador (Principal)
Escenario Principal
1. Jugador selecciona en el menú principal la opción Modo individual.
2. La maquina devuelve la pantalla de selección del personaje.
3. El jugador elige el personaje con el que desea jugar y le da a “Aceptar".
4. El sistema guarda el personaje seleccionado y comienza la partida.
Escenarios Alternativos
*a. El jugador cierra el videojuego.
3a. Menú principal: **Include Volver Menú principal**.

4.4.4. Caso de Uso: dosJugadores

Caso de Uso: dosJugadores
Precondiciones: Ninguna
Postcondiciones: El jugador establece una partida en modo Dos jugadores con otro jugador.
Descripción: Partida de dos jugadores.
Resumen: El jugador quiere jugar una partida colectiva con otro jugador.
Actores: Jugador 1(principal) y jugador 2(secundario)
Escenario Principal
1. Jugador selecciona en el menú principal la opción Modo Dos jugadores.
2. La maquina devuelve la pantalla de selección del personaje.
3. El jugador elige los personajes con el que desean jugar y le da a Aceptar.
4. El sistema guarda los personajes seleccionados y comienza la partida.

Escenarios Alternativos
*a. El jugador cierra el videojuego.
3a. Menú principal: **Include Volver Menú principal**.

4.4.5. Caso de Uso: Pausar el juego

Caso de Uso: Pausar el juego
Precondiciones: Existe una partida en proceso
Postcondiciones: La maquina pausa el juego.
Descripción: Partida de dos jugadores.
Resumen: El jugador quiere pausar la partida durante algún tiempo.
Actores: Jugador 1(principal) y jugador 2(secundario)
Escenario principal
1. La máquina muestra la ejecución de una partida.
2. El jugador pulsa el botón de pausa.
3. La maquina pausa la partida y muestra en pantalla el menú de pause.
4. El jugador, cuando esté listo, pulsa continuar.
5. La máquina reanuda la partida.
Escenarios alternativos
*a. El jugador cierra el videojuego.
2a. El jugador no pulsa el botón de pausa.
    2.1a. La maquina continúa con la partida.
4a. El jugador elige el arma secundaria.
    4.1a La maquina al reanudar convierte el arma secundaria en la principal.
4b. El jugador elige la opción de menú principal.
    4.1b. El sistema termina el partido o entrenamiento y muestra el menú.
4c. El jugador elige la opción salir
    4.1c. La máquina termina la partida y sale del juego.

4.4.6. Caso de uso: Créditos

Caso de uso: Créditos
Precondiciones: Ninguna
Postcondiciones: La maquina devuelve la pantalla de créditos.
Descripción: Persona que ha participado en el desarrollo del videojuego.
Resumen: El jugador quiere ver los créditos del videojuego.
Actores: Jugador (Principal)
Escenario principal
1. Jugador selecciona en el menú principal la opción Créditos.
3. El sistema muestra la pantalla de créditos.
4. Menú principal: **Include Volver Menú principal**.

Escenarios alternativos
*a. El jugador cierra el videojuego.

4.4.7. Caso de uso: Récords

Caso de uso: Récords
Precondiciones: Ninguna
Postcondiciones: La maquina devuelve la pantalla de los récords.
Descripción: Muestra las cinco puntuaciones más altas logradas
Resumen: El jugador quiere ver los récords registrados del videojuego.
Actores: Jugador (Principal)
Escenario principal
1. Jugador selecciona en el menú principal la opción Récords.
3. El sistema muestra la pantalla de récords.
4. Menú principal: **Include Volver Menú principal**.
Escenarios alternativos
*a. El jugador cierra el videojuego.

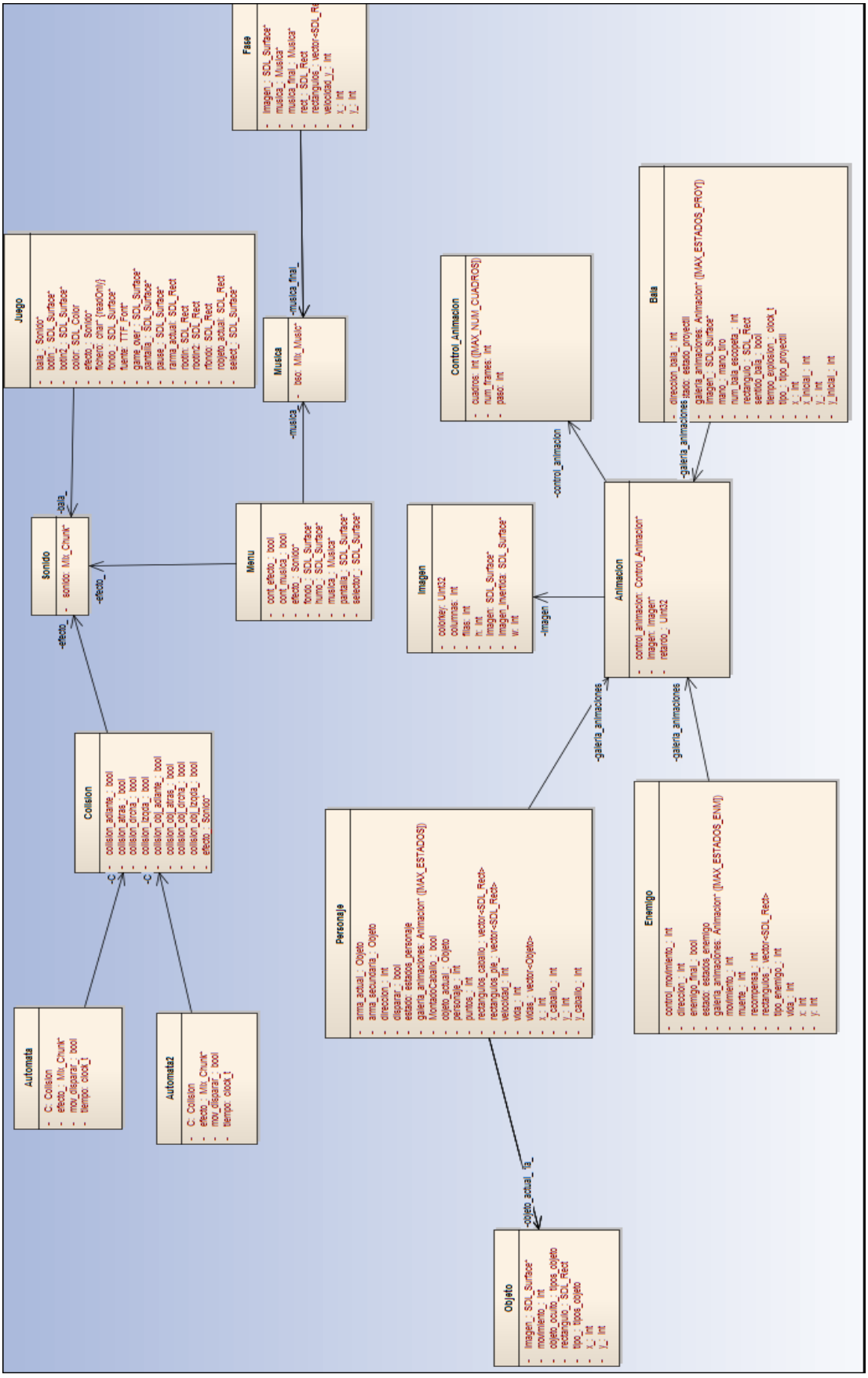
4.4.8. Caso de uso: Opciones

Caso de uso: Opciones
Precondiciones: Ninguna
Postcondiciones: La maquina devuelve la pantalla de opciones.
Descripción: El jugador des/activa la música y los efectos, además de conocer los controles.
Resumen: El jugador cambia las opciones del videojuego.
Actores: Jugador (Principal)
Escenario principal
1. El jugador selecciona en el menú principal la opción opciones.
3. El sistema muestra la pantalla de opciones.
4. El jugador selecciona una opción.
5. Menú principal: **Include Volver Menú principal**.
Escenarios alternativos
*a. El jugador cierra el videojuego.
4a. Opción teclado:
    4.1a. El sistema muestra los controles.
4b. Des/activar sonido.
    4.1b. El sistema des/activa el sonido.
4c. Des/activar efectos de sonido.
    4.1c. El sistema des/activa los efectos de sonido.

4.4.9. Caso de uso: Salir

Caso de uso: Salir
Precondiciones: Ninguna
Postcondiciones: El sistema cierra el juego.
Descripción: El jugador desea salir del juego.
Resumen: El jugador sale del videojuego.
Actores: Jugador (Principal)
Escenario principal
1. El sistema muestra el menú principal.
2. El jugador selecciona salir.
3. El sistema muestra la pantalla de confirmación de salida.
4. El jugador confirma que desea salir.
5. El sistema sale del juego
Escenarios alternativos
*a. El jugador cierra el videojuego.
4a. El jugador no desea salir.
    4.1a. El sistema vuele a mostrar la pantalla de menú principal.

4.4.10. Modelo Conceptual



Listado de las clases

Clase Juego: Controla el game loop del videojuego. Parte más importante.

Clase Menú: Control de Menús.

Clase Sonido: Creación de sonidos.

Clase Música: Creación de la banda sonora.

Clase Autómata. Control del movimiento e interacción con el personaje principal.

Clase Autómata2: Control del movimiento e interacción con el personaje secundario.

Clase Colisión: Control de colisiones.

Clase Personaje: Creación de los personajes.

Clase Enemigo: Creación de los enemigos.

Clase Objeto: Creación de los objetos.

Clase Bala: Creación de los tipos de balas.

Clase Animación: Creación de la animación del sprite.

Clase Imagen: Creación de una imagen.

Clase Control_Animación: Control de la animación.

Como podemos observar en el modelo anterior, el software está formado por un conjunto de clases relacionadas entre sí. Hay dos grupos claramente diferenciados.

La clase Sonido está relacionada con la clase Menú, Juego y Colisión.

La clase Música se relaciona con la clase Fase y la clase Menú.

La clase Colisión se relaciona con la clase Autómata y Automata2.

En el otro grupo se encuentran las clases que controlan el aspecto animado del juego. Personaje, Objeto, Bala y Enemigo, todas ellas se relacionan con la clase Animación y esta a su vez con Imagen y Control de Animación.

4.4.11. Contrato de las operaciones

Clase Menú

La clase Menú crea y controla los menús que aparecen en el videojuego.

- **Menu()**

  Inicia SDL y las librerías de imagen y sonido.
  Crea y configura la pantalla principal del juego, y reproduce la música del menú.

- **Menu(bool musica, bool efectos, int pant_completa)**

  Constructor alternativo creo el menú principal y reproduce la música si lo índica.

- **MenuPrincipal()**

  Construye el menú principal del juego para que el jugador pueda navegar por él.

- **void UnJugador()**

  Construye el menú Modo individual para que el jugador elija el personaje y así comenzar la partida.

- **void Menu::DosJugadores()**

  Construye el menú Modo dos jugadores para que los jugadores elijan sus personajes y así comenzar la partida.

- **void Opciones()**

  Construye el menú Opciones.

- **void Records()**

  Construye el menú de los Récords.

- **void Creditos()**

  Construye el menú de los Créditos.

- **~Menu()**

  Destruye el menú.

Clase Juego

La clase Juego se encarga de gestionar el estado del juego (pausar, reanudar, salir), crea la fase, los personajes, enemigos, objetos, etc. En definitiva, controla todos los elementos que intervienen en la partida.

- **Juego ()**

  Inicia SDL y las librerías de imagen y sonido.

- **void Jugar_1player (int personaje, bool Emusica, bool Eefecto, int Epantallacompleta)**

  Crea la historia inicial y al personaje principal.

- **void Jugar_2player (int personaje1, int personaje2, bool Emusica, bool Eefecto, int Epantallacompleta)**

  Crea la historia inicial y a los personajes principales.

- **bool Partida_1player(Personaje &principal, bool Emusica, bool Eefecto, int Epantallacompleta, const char* history, const char* history_fin, int num_fase)**

  Crea la fase, enemigos y objetos que aparecen en ésta. Contiene el game loop del juego.

- **bool Partida_2player(Personaje &principal, Personaje &secundario, bool Emusica, bool Eefecto, int Epantallacompleta, const char* history, const char* history_fin, int num_fase)**

  Crea la fase, enemigos y objetos que aparecen en ésta. Contiene el game loop del juego.

- **void Objetos_Fase_dibujar(vector <Objeto> &obj)**

  Dibuja en pantalla todos los objetos.

- **void Enemigos_Fase_dibujar(vector <Enemigo> &e)**

  Dibuja en pantalla todos los enemigos.

- **bool Muerto(Personaje& principal, Fase& fase, bool Eefecto)**
- **bool Muerto_2player(Personaje& principal, Personaje& secundario, bool Eefecto)**

  Controla si debe reiniciarse la fase o por el contrario el juego ha terminado.

- **bool Pause(Personaje &principal, bool &terminar, int Epantallacompleta)**
- **bool Pause_2player(bool &terminar, int Epantallacompleta)**

  Crea el modo pause y pausa el juego.

- **~Juego()**

  Destruye todos los atributos de la clase juego.

Clase Fase

La clase Fase se encarga de crear y controlar los distintos niveles del juego.

- **Fase::Fase(int num_fase)**

  Crea la fase (con los rectángulos de colisión) y la música que sonará.

- **inline int Fase::pos_x() const**

  Devuelve la posición actual del eje x.

- **inline int Fase::pos_y() const**

  Devuelve la posición actual del eje y.

- **void Fase::dibujar(SDL_Surface *pantalla)**

  Dibuja la fase en pantalla.

- **inline void Fase::pos_x(int x)**

  Modifica la posición del eje x.

- **inline void Fase::pos_y(int y)**

  Modifica la posición del eje x.
- **inline void avanzar_y()**

  Avanza el scroll

- **void Fase::anadir_rectangulo(int x, int y, int ancho, int alto)**

  Añade rectángulo para detectar la colisión.

- **SDL_Rect rectangulos(int i) const**

  Devuelve el rectángulo almacenado en la posición i.

- **inline int Fase::num_rectangulos() const**

  Devuelve el número total de rectángulos

- **Musica* Fase::musica()**

  Devuelve la música de la fase creada.

- **Fase::~Fase()**

  Destruye la fase.

Clase Personaje

La clase Personaje es la encargada de controlar cada uno de los parámetros del personaje principal.

- **Personaje(int tipo_personaje)**

  Inicializa los parámetros del personaje.

- **void Comenzar_Fase(int num_fase)**

  Crea el personaje principal y los rectángulos de colisión.

- **void dibujar(SDL_Surface *pantalla)**

  Dibuja al personaje en pantalla.

- **int pos_x(void)**

  Devuelve la posición del eje x.

- **int pos_y(void)**

  Devuelve la posición del eje y.

- **int pos_x_caballo(void)**

  Devuelve la posición del eje x del caballo.

- **int pos_y_caballo(void)**

  Devuelve la posición del eje y del caballo.

- **int direccion(void)**

  Devuelve la dirección del personaje.

- **int vidas(void)**

  Devuelve el número de vidas del personaje.

- **bool disparar(void)**

  Devuelve la variable del control de disparo. Indica si el personaje dispara o no.

- **int velocidad(void)**

  Devuelve la velocidad con la que se mueve el personaje.

- **bool MontadoCaballo(void)**

  Indica si esta montado en caballo o por el contrario, se mueve a pie.

- **estados_personaje estado_actual(void)**

  Devuelve el estado del personaje.

- **Objeto ultima_vida(void)**

  Devuelve la última vida.

- **Objeto objeto_actual(void)**

  Devuelve el objeto actual que posee el personaje.

- **Objeto arma_actual(void)**

  Devuelve el arma actual que posee.

- **Objeto arma_secundaria(void)**

  Devuelve el arma secundaria del personaje.

- **void cambio_estado(estados_personaje status)**

  Modifica el estado del personaje.

- **void pos_x(int x)**

  Modifica la posición del eje x.

- **void pos_y(int y)**

  Modifica la posición del eje y.

- **void pos_x_caballo(int x)**

  Modifica la posición del eje x del caballo.

- **void pos_y_caballo(int y)**

  Modifica la posición del eje x.

- **void puntos(int puntos)**

  Incrementa la puntuación del personaje.

- **void direccion(int dir)**

  Modifica la dirección del personaje.

- **void vidas(int vida)**

  Incrementa las vidas

- **void disparar(bool dis)**

  Modifica la variable de control del disparo.

- **void ganar_vida(Objeto obj)**

  El personaje gana una vida.

- **void perder_vida()**

  El personaje pierde una vida.

- **void vidas_player1()**

  Modifica la posición de las vidas en pantalla del player 1.

- **void objeto_actual(Objeto act)**

  Modifica el objeto actual.

- **void arma_actual(Objeto arma_actual)**

  Modifica el arma actual.

- **void arma_secundaria(Objeto arma_secundaria)**

  Modifica el arma secundaria.

- **void velocidad(int velocidad)**

  Modifica la velocidad del personaje.

- **void MontadoCaballo(bool montarcaballo)**

  El personaje se monta a caballo o se cambia a pie.

- **void derecha_x(void)**

  El personaje mueve su posición hacia la derecha.

- **void izquierda_x(void)**

  El personaje mueve su posición hacia la izquierda.

- **void bajar_y(void)**

  El personaje mueve su posición hacia abajo.

- **void subir_y(void)**

  El personaje mueve su posición hacia arriba.

- **void vidas_dibujar(SDL_Surface *pantalla_)**

  Dibuja las vidas en pantalla.

- **void anadir_rectangulo_pie(int x, int y, int ancho, int alto)**

  Añade rectángulos de colisión del personaje cuando va a pie.

- **void anadir_rectangulo_caballo(int x, int y, int ancho, int alto);**

  Añade rectángulos de colisión del personaje cuando va a caballo.

- **int num_rectangulos_pie(void)**

  Devuelve el número total de rectángulos.

- **int num_rectangulos_caballo(void)**

  Devuelve el número total de rectángulos.

- **int puntos(void)**

  Devuelve la puntuación del jugador.

- **SDL_Rect rectangulos_pie(int i)**

  Devuelve el i-ésimo rectángulo de colisión: Personaje a pie.

- **SDL_Rect rectangulos_caballo(int i)**

  Devuelve el i-ésimo rectángulo de colisión: Personaje a caballo.

- **void disparar_arma (vector <Bala> &v, int &n, bool Eefecto)**

  El personaje dispara cualquier arma. Crea la bala.

- **void reiniciar()**

  Reinicializa parámetros.

- **void interfaz_player1()**

  Modifica la posición de los objetos que aparecen en la intervaz (arma actual,objeto actual,etc).

- **void interfaz_player2()**

  Modifica la posición de los objetos que aparecen en la intervaz (arma actual,objeto actual,etc).

- **~Personaje()**

  Destruye el personaje.

Clase Enemigos

La clase Enemigos se encarga de controlar los parámetros de los diferentes enemigos.

- **Enemigo()**

  Constructor vacío de la clase Enemigo.

- **Enemigo(enum tipo_enemigo, int x, int y, int direccion, int num_fase)**

  Crea el enemigo, añade los rectángulos de colisión e inicializa parámetros.

- **int pos_x(void)**

  Devuelve posición del eje x que ocupa el enemigo.

- **int pos_y(void)**

  Devuelve posición del eje y que ocupa el enemigo.

- **int tipo_enemigo(void)**

  Indica el tipo de enemigo que es.
- **int direccion(void)**

  Indica la dirección que tiene el enemigo.

- **int control_movimiento(void)**

  Devuelve variable que controla el movimiento del personaje.

- **int vida(void)**

  Devuelve el número de vidas que posee el enemigo.

- **int muerte(void)**

  Devuelve variable que controla el tiempo en estado muerte.

- **int recompensa(void)**

  Devuelve la recompensa por acabar por este enemigo.

- **bool enemigo_final(void)**

  Indica si es un enemigo final.

- **estados_enemigo estado_actual(void)**

  Devuelve el estado del enemigo.

- **void dibujar(SDL_Surface *pantalla)**

  Dibuja en pantalla al enemigo.

- **void cambio_estado(estados_enemigo status)**

  Cambia el estado del enemigo.

- **void pos_x(int x)**

  Modifica la posición del eje x.

- **void pos_y(int y)**

  Modifica la posición del eje y.

- **void direccion(int direccion)**

  Modifica la dirección del enemigo.

- **void control_movimiento(int mov)**

  Modifica la variable que controla el movimiento.

- **void vida(int vida)**

  Modifica el número de vidas del enemigo.

- **void muerte(int muerto)**

  Modifica la variable muerte.

- **void avanzar_y()**

  El enemigo avanza junto al scroll.

- **void no_avanzar_y()**

  El enemigo no avanza.

- **void derecha_x(void)**

  El enemigo se mueve hacia la derecha.

- **void izquierda_x(void)**

  El enemigo se mueve hacia la izquierda.

- **void bajar_y(void)**

  El enemigo se mueve hacia abajo.

- **void subir_y(void)**

  El enemigo se mueve hacia arriba.

- **void anadir_rectangulo(int x, int y, int ancho, int alto)**

  Añade rectángulo de colisión.

- **int num_rectangulos(void)**

  Devuelve el número total de rectángulos de colisión.

- **SDL_Rect rectangulos(int i)**

  Devuelve el i-ésimo rectángulo de colisión.

- **void disparar_arma_enemigo (vector <Bala> &v, int &n)**

  El enemigo dispara el arma . Crea la bala, el cuchillo, o el proyectil que sea.

- **void movimiento(Personaje principal, vector <Bala> &e, int &m)**

  Controla el movimiento de los personajes. Inteligencia Artificial del juego.

- **~Enemigo()**

  Destruye al enemigo.

Clase Objeto

La clase Objeto es la encargada de controlar los parámetros de los objetos que aparecen el videojuego.

- **Objeto()**

  Constructor vacío del objeto.

- **Objeto(enum tipos_objeto tipo, int pos_x, int pos_y, enum tipos_objeto objeto_oculto, int num_fase)**

  Crea el objeto, añade rectangulos de colisión e inicializa atributos.

- **void objeto_oculto(enum tipos_objeto obj)**

  Modifica el objeto oculto por un barril.

- **void dibujar(SDL_Surface *pantalla)**

  Dibuja en pantalla el objeto.

- **void pos_x(int x)**

  Modifica la posición del eje x del objeto.

- **void pos_y(int y)**

  Modifica la posición del eje y del objeto.

- **void movimiento(int mov)**

  Velocidad con la que avanza los objetos junto al scroll.

- **void avanzar_y()**

  El objeto avanza junto al scroll.

- **int pos_x(void)**

  Devuelve la posición del eje x del objeto.

- **int pos_y(void)**

  Devuelve la posición del eje y del objeto.

- **enum tipos_objeto tipo_objeto(void)**

  Devuelve el tipo de objeto que es.

- **enum tipos_objeto objeto_oculto(void)**

  Devuelve el tipo de objeto que es el objeto oculto.

- **void anadir_rectangulo(int x, int y, int ancho, int alto)**

  Añade rectángulo de colisión.

- **SDL_Rect rectangulo() const**

  Devuelve el rectángulo de colisión.

- **~Objeto()**

  Destruye el objeto.

Clase Bala

La clase Bala controla los parámetros de la bala.

- **Bala(enum tipo_proyectil tipo)**

  Crea la bala e inicializa sus atributos.

- **int pos_x(void)**
- **int pos_y(void)**

  Devuelve la posición del eje x e y respectivamente de la bala.

- **int pos_x_inicial(void)**
- **int pos_y_inicial(void)**

  Devuelve la posición del eje x e y inicial de la bala.

- **int direccion_bala(void)**

  Devuelve la dirección que tiene la bala.

- **bool sentido_bala(void)**

  Devuelve el sentido de la bala.

- **int num_bala_escopeta(void)**

  Devuelve el número de bala que es.

- **enum tipo_proyectil tipo()**

  Devuelve el tipo de proyectil: bala, super bala, flecha, cuchillo o dinamita.

- **clock_t tiempo_explosion()**

  Devuelve el tiempo de explosión de la dinamita.

- **estado_proyectil estado_actual(void)**

  Devuelve el estado de la bala.

- **bool limite(void)**

  Comprueba que la bala esta fuera de un rango.

- **void dibujar(SDL_Surface *pantalla)**

  Dibuja el proyectil en pantalla

- **void pos_x(int x)**
- **void pos_y(int y)**

  Modifica la posición del eje x e y respectivamente del proyectil.

- **void direccion_bala(int bala)**

  Modifica la dirección del proyectil.

- **void sentido_bala(bool bala)**

  Modifica el sentido del proyectil.

- **void num_bala_escopeta(int bala)**

  Modifica el número de la bala.

- **void cambio_estado(estado_proyectil status)**

  Cambia el estado del proyectil.

- **void mano_tiro(enum mano_tiro mano)**

  Cambia la mano del disparo.

- **void tiempo_explosion(clock_t tiempo)**

  Modifica el tiempo de explosión de la dinamita.

- **void posicion_inicial(Personaje principal, int direccion)**

  Establece la posición inicial de la bala.

- **void posicion_inicial_enemigo(Enemigo principal, int direccion)**

  Establece la posición inicial de la bala.
- **void diag_derecha()**

  Mueve la bala disparada por el personaje principal diagonalmente hacia la derecha.

- **void diag_izquierda()**

  Mueve la bala disparada por el personaje principal diagonalmente hacia la izquierda.

- **void recto()**

  Mueve la bala disparada por el personaje principal recto.

- **void enemigo_diag_derecha()**

  Mueve la bala disparada por el enemigo diagonalmente hacia la derecha.

- **void enemigo_diag_izquierda()**

  Mueve la bala disparada por el enemigo diagonalmente hacia la izquierda.

- **void enemigo_recto()**

  Mueve la bala disparada por el enemigo recto.

- **void anadir_rectangulo(int x, int y, int ancho, int alto)**

  Añade un rectángulo de colisión .

- **SDL_Rect rectangulo() const**

  Devuelve el rectángulo de colisión.

- **bool colision_objetos(vector <Objeto> &objetos, int num_fase)**

  Indica si existe colisión con algún objeto.

- **bool colision_personaje(Personaje &principal)**

  Indica si existe colisión con el personaje principal.

- **bool colision_enemigos(vector <Enemigo> &enemigos, Personaje &principal)**

  Indica si existe colisión con algún enemigo.

- **~Bala()**

  Destruye la bala.

Clase Colisión

La clase Colisión detecta las colisiones de los elementos con el personaje principal.

- **Colision()**

  Constructor vacío de colisión.

- **void colision_fase(Personaje &principal, Fase &fase)**

  Detecta si existe colisión con los componentes de la fase.

- **bool colision_objetos(Personaje &principal, vector <Objeto> &objetos, bool Eefecto)**

  Detecta si existe colisión con los objetos de la fase.

- **void colision_enemigos(Personaje &principal, vector <Enemigo> &enemigos)**

  Comprueba si existe colisión con los enemigos de la fase.

- **inline bool colision_izqda()**
- **inline bool colision_drcha()**
- **inline bool colision_adlante()**
- **inline bool colision_atras()**
- **inline bool colision_obj_izqda()**
- **inline bool colision_obj_drcha()**
- **inline bool colision_obj_adlante()**
- **inline bool colision_obj_atras()**

  Consulta los distintos atributos de la colisión.

- **inline void colision_izqda(bool izqda)**

- **inline void colision_drcha(bool drcha)**
- **inline void colision_adlante(bool adlante)**
- **inline void colision_atras(bool atrás)inline void colision_obj_izqda(bool izqda)**
- **inline void colision_obj_drcha(bool drcha)**
- **inline void colision_obj_adlante(bool adlante)**
- **inline void colision_obj_atras(bool atras)**

  Modifica los distintos atributos de la colisión.

Clase Autómata

La clase Autómata controla el movimiento del personaje principal.

- **Automata ()**

  Crea los atributos de la clase autómata.

- **void movimiento_apie (Personaje& principal,Teclado& teclado, Fase fase, vector <Objeto> &obj, vector <Enemigo> &enem, bool Eefecto)**

  Controla los movimientos y estados del personaje cuando va a pie.

- **void movimiento_acaballo (Personaje& principal, Teclado& teclado, Fase fase, vector <Objeto> &obj, vector <Enemigo> &enem, bool Eefecto)**

  Controla los movimientos y estados del personaje cuando monta a caballo.

- **bool mov_disparar(void)**

  Devuelve la variable que indica si el último movimiento realizado ha sido un disparo.

- **void mov_disparar(bool mov)**

  Modifica la variable.

- **~Automata()**

  Destruye la clase.

Clase Autómata2

La clase Autómata2 controla el movimiento y lo estados del personaje del jugador número dos.

Clase Música

La clase Música se utiliza para reproducir la música.

- **Musica(const char *ruta)**

  Inicializa SDL, SDL_mixer y carga el fichero de música.

- **void reproducir()**

  Reproduce la música.

- **void pausar()**

  Pausa la música.

- **~Musica()**

  Destruye la música. Libera memoria.

Clase Sonido

La clase Sonido se utiliza para reproducir los efectos de sonido .

- **Sonido::Sonido(const char *ruta)**

  Inicializa SDL, SDL_mixer, selecciona el número de canales y carga el ficheros del efecto de sonido.

- **void reproducir(int canal)**

  Reproduce el sonido.

- **~Sonido()**

  Destruye el sonido. Libera memoria.

Clase Control de Animación

Esta clase controla la secuencia de animación de los personajes del videojuego.

- **Control_Animacion(char *frames)**

  Constructor.

- **int cuadro(void)**

  Devuelve el paso actual.

- **bool es_primer_cuadro(void)**

  Indica si es el primer paso de la animación.

- **int numero_cuadros(void)**

  Indica el número de cuadros que compone la animación.

- **int avanzar(void)**

  Avanza al paso siguiente.

- **void reiniciar(void)**

  Vuelve al inicio de la animación.

- **~Control_Animacion()**

  Destructor.

Clase Imagen

La clase encargada de cargar y dibujar las distintas imágenes.

- **Imagen(const char *ruta, int filas = 1, int columnas = 1,Uint32 r = 0, Uint32 g = 255, Uint32 b = 0)**

  Constructor.

- **void dibujar(SDL_Surface *superficie, int i, int x, int y, int flip = 1)**

  Dibuja la imagen deseada.

- **int anchura(void)**

  Devuelve la anchura de la imagen.

- **int altura(void)**

  Devuelve la altura de la imagen.

- **int cuadros(void)**

  Cuadro determinado por la imagen.

- **~Imagen()**

  Destructor.

Clase Animación

La clase que controla las animaciones del juego.

- **Animacion(const char *ruta_imagen, int filas, int columnas, const char *frames, int retardo)**

  Constructor de la animación.

- **void animar(SDL_Surface *pantalla, int x, int y, int flip = 1)**

  Animación fija e infinita en pantalla.

- **void paso_a_paso(SDL_Surface *pantalla, int x, int y, int flip = 1)**

  Dibuja la animación paso a paso .

- **Uint32 retardo()**
  Devuelve el retardo.

Clase Teclado

La clase en donde se configuran las teclas que utilizaremos en el videojuego.

- **Teclado()**

  Constructor: Crea y configura el teclado del juego .

- **void actualizar(void)**

  Actualiza el teclado.

- **bool pulso(enum teclas_configuradas tecla)**

  Indica si la tecla pulsada está configurada.

- **~Teclado()**

  Destructor.

Capítulo 5

Desarrollo

En este apartado explicaremos el desarrollo del videojuego, describiendo las herramientas utilizadas y necesarias para la producción, así como, los problemas encontrados y sus respectivas soluciones.

5.1. Modelado 2D

Para el diseño de los gráficos desde un principio tuve claro que seria en 2D y que yo mismo lo diseñaría. Debido a la temática del juego comencé diseñando bocetos en papel hasta encontrar lo que iba buscando; dibujos sencillo sin muchos detalles, como si de dibujos animados se tratase.
Con algunos bocetos ya terminados, los primeros fueron los dos personajes principales, pensé una forma de pasar los dibujos del papel al pc que fuera lo más eficaz posible. Primero, lo intente escaneando los bocetos y luego, mediante el ratón modificar la imagen, sin embargo, era un proceso muy lento y poco exacto, no me gustaba nada. Así que, como ultima decisión decidí mediante una tableta gráfica pintar directamente sobre esta los bocetos diseñados.
Para ello, utilice dos programas que ya estaba familiarizado; el kolourPaint y el Gimp.

Imagen 5.1. KolourPaint. Extraída de http://es.wikipedia.org/wiki/KolourPaint

Imagen 5.2. Gimp. Extraída de http://www.cesarius.net/wp-content/uploads/2008/07/gimp24.jpg

El primero lo use para el diseño de los personajes y objetos mientras que el segundo lo utilice para el desarrollo de los menús, interfaz, fases e historia.

Imagen 5.3. Enemigo final dibujado con GIMP

Imagen 5.4. Enemigo final dibujado con GIMP

Para algunos casos he utilizado pequeñas técnicas del propio GIMP. Efectos como el humo del revolver, el fondo del menú o el propio revolver obtenidos aplicando diferentes filtros.

A continuación muestro algunas imágenes del proceso:

Imagen 5.5. Revolver original

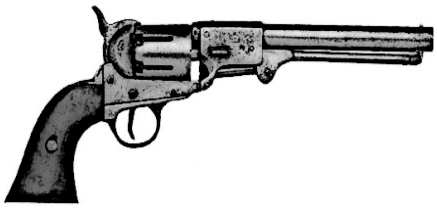


Imagen 5.6. Revolver; Marcador del juego

Imagen 5.7. Menú Principal, original

Imagen 5.8. Efecto de las balas al chocar

Imagen 5.9. Cartel de recompensa

Textura

La mayoría de las texturas han sido generadas con el GIMP. Otras en cambio, como los árboles o los ahorcados han sido obtenidas de componentes de videojuegos libres.

Algunas texturas presentes en el juego:

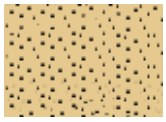


Imagen 5.10. Textura Fase 1

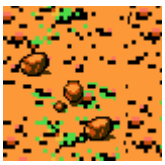


Imagen 5.11. Textura Fase 2

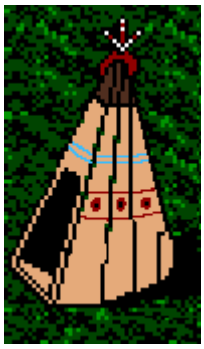


Imagen 5.12. Tienda india

Imagen 5.13. Persona ahorcada de la Fase 3

Imagen 5.14. Árbol

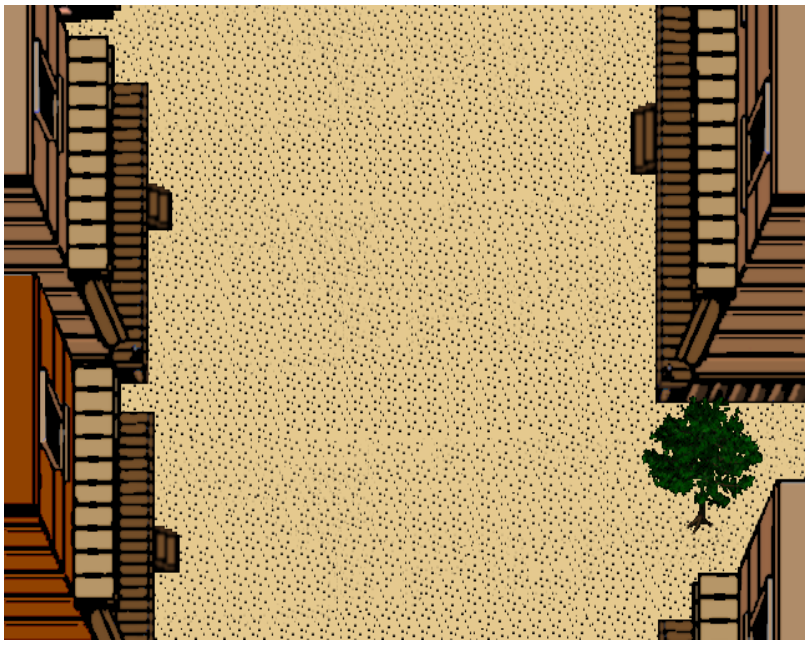


Imagen 5.15. Apariencia de la primera fase

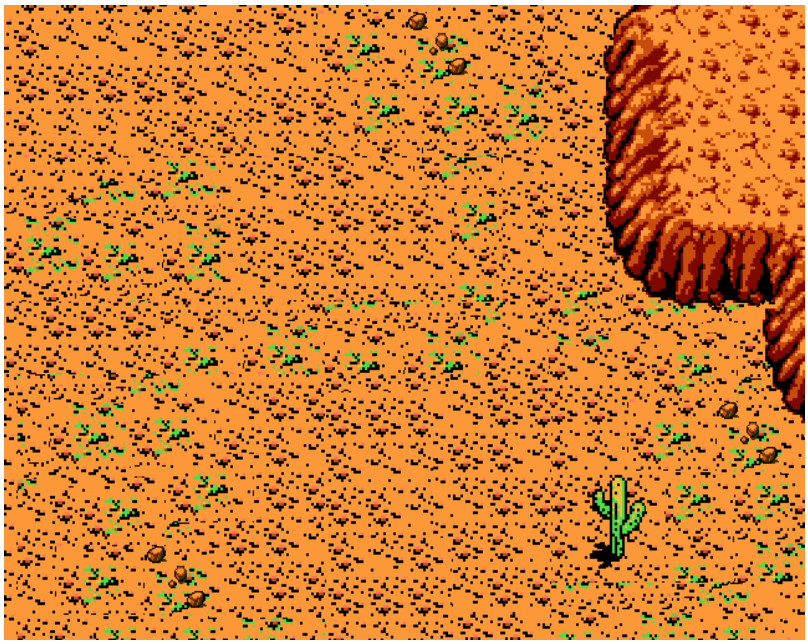


Imagen 5.16. Apariencia de la segunda fase

Animación

Este proceso junto a la implementación es uno de los más extensos por su tiempo. Requiere de paciencia y mucha dedicación. Puesto que se basa en 2D, deberemos crear al personaje en diferentes posturas para dotarlo de movimiento. Todos los personajes gozan de animaciones tanto para moverse como disparar y morir, por lo que he tenido que echar muchas horas diseñando.

A continuación muestro algunas de las animaciones de los personaje:

Imagen 5.17. Animación enemigo

Imagen 5.18. Animación andar enemigo final

Imagen 5.19 Animación cowboy a pie

Imagen 5.20. Animación indio a caballo.

5.2. Sonido

Tanto la música como los efectos han sido obtenidos de páginas web dedicadas a efectos de sonidos de libre distribución y música de autores libres.

A la hora de editar los diferentes sonidos del videojuego (cortar, amplificar, convertir a otros formatos, etc) hemos utilizado el programa Audacity.

Imagen 5.21. Audacity. Extraída de http://richardtesaluda.blogspot.com/2009/06/audacity.html

Los diferentes sonidos que escucharemos al interactuar con el videojuego son los siguientes:

Disparo: Cuando el jugador pulsa alguna tecla de disparo. Dependiendo del arma usada el efecto de sonido es diferente. Al navegar por el menú también lo escucharemos.

Contrarreloj: Cuando el personaje coge algún objeto que dura un tiempo fijo como la herradura o las botas.

Risa: Cuando coge un arma nueva, una vida o finaliza con éxito la fase.

Risa mala: Cuando el personaje muere y no tiene más vidas.

Campana: Cuando el personaje muere alcanzado por una bala o al chocar con un enemigo.

Grito: Cuando monta a caballo.

5.3. Codificación

Este apartado esta destinado a describir los aspectos técnicos relacionados con la codificación del videojuego.

5.3.1. Lenguaje utilizado

Para la implementación de The Most Wanted hemos utilizado el lenguaje C++, ya que estaba familiarizado con él, además de poder abarcar sin ninguna limitación nuestro videojuego.

5.3.2. Bibliotecas Gráficas utilizadas

Me he decantado por la biblioteca libSDL. Al igual que C++ es una biblioteca muy completa que me permitió abarcar el videojuego sin ningún problema, además de poseer conocimientos previos adquiridos en la asignatura de diseño de videojuego durante la carrera.
Otro aspecto importante de esta librería era la portabilidad a otro sistema operativo como Windows. Aunque desde el principio, el objetivo no era poder ejecutarlo en Windows sino en versiones GNU/

Linux solamente, al final decidí que también fuera posible poder disfrutar del videojuego en Windows.
En concreto se utilizaron:

SDL (Simple DirectMedia Layer): Conjunto de bibliotecas que proporcionan funciones básicas para realizar operaciones de dibujado 2D, gestión de efectos de sonido y música, y carga y gestión de imágenes.

SDL Image: Extiende notablemente las capacidades para trabajar con diferentes formatos de imagen.

SDL Mixer: Extiende las capacidades de SDL para la gestión y uso de sonido y música en aplicaciones y juegos.

SDL TTF: Permite usar fuentes TrueType en aplicaciones SDL.

5.3.3. Funcionalidades implementadas

Movimiento del personaje principal

La implementación del autómata es algo sencilla. Dependiendo del cursor que pulsemos el personaje se desplazará por el eje x e y,, además de cambiar de estado. Si pulsamos las teclas de disparo, el estado del personaje cambiará al que corresponda (izquierda,derecha o recto) y se creará un objeto de la clase bala controlada por funciones de posición. Las balas disparada por los personajes tendrán una velocidad y describirán un movimiento rectilíneo.

Colisión

Implementar la colisión de la forma más exacta y realista fue más compleja de lo que creía.
Para detectar la colisión tuve que crear rectángulos para cada uno de los elementos que pueden colisionar.
Existe colisiones entre el personaje principal y los elementos de las fases, los barriles, las balas, los enemigos y los distintos objetos que nos encontramos.
Para detectar la colisión con los elementos de la fase, diferencio cuatro tipos de colisiones: colisión por la izquierda, colisión por la derecha, colisión de frente y colisión atrás. Dependiendo del rectángulo del personaje principal que choque primero con los rectángulo del elemento en cuestión pues así será la colisión. Con esto conseguimos que si la colisión se produce por lado izquierdo, el personaje no podrá avanzar hacia esa dirección.
La colisión con las balas, enemigos y objetos es bastante más simple. Cuando coincide el eje x e y de alguno de los rectángulos se produce la colisión.

Movimiento de los enemigos

Probablemente la tarea más compleja sea la inteligencia artificial, dotar a los enemigos de cierta inteligencia. Decir de antemano que no tienen mucha, pues se basan en movimientos rectilíneos uniformes. Para controlar el movimiento de cada personaje tendrán una variable de control de movimiento que irá variando conforme se vayan moviendo. Cada enemigo tendrá un movimiento especifico por lo que existe doce o trece tipos de movimientos. Algunos tan sencillos como que vayan andando y que cada cierto tiempo disparen al frente, otros disparan con una dirección según donde nos encontremos y otros llegarán a saltar.
Los enemigos finales gozan de una inteligencia superior, asi como de movimientos más complejos y menos intuitivos.

Objetos

Al producirse una colisión entre el personaje y cualquier objeto, algunos de los atributos del personaje se verán afectados. Por ejemplo, si colisionamos con unas botas vaqueras, la velocidad del personaje aumentará desplazándose más rápido durante un periodo de tiempo. Si por el contrario

colisionamos con un saco de dinero, la puntuación aumentará. Y así, con todos los objetos que colisionamos.

DosJugadores

En el caso del modo dos jugadores, hemos utilizado el mismo motor que utilizamos para un jugador, con la salvedad que el jugador 2 utiliza otra clase autómata puesto que los controles son diferentes.
Además, he implementado la colisión entre los dos personajes para que no se superpongan a lo largo de la partida.

Archivos de configuración

Mi primera intención era que lo enemigos y objetos apareciesen en posiciones aleatorias cada vez que comenzaremos una fase y que su número total no estuviese predeterminado en el caso de los enemigos, sino conforme se destruyeran los enemigos se generasen unos nuevos. Al final no he conseguido hacerlo, y me decante por poner un numero predeterminado de enemigos y que apareciesen desde una posición concreta.
Para ello, he creado archivos de texto donde se escriben el tipo de enemigo, coordenadas y dirección inicial que posee, los cuales el sistema lee al comienzo de cada fase. Lo mismo con los objetos. Así podríamos cambiar la dificultad del juego sin tener que modificar el código, añadiendo enemigos, eliminando objetos. En definitiva, modificando la fase a nuestro gusto.

5.4. Pruebas

5.4.1. Introducción

El desarrollo de un plan de pruebas es fundamental para completar la finalización de cualquier aplicación. El plan de prueba consiste en realizar un gran numero de pruebas llevando al programa a los puntos límites donde podrá fallar.

Las pruebas que realizaremos sobre la aplicación son:

Pruebas de contenido: Estas pruebas nos permiten detectar errores en el contenido de la aplicación.

Pruebas de interfaz: Estas pruebas validan los aspectos relativos al funcionamiento de la interfaz y como responde ante los jugadores, así como la forma de mostrarles la información de sus unidades y las acciones que han realizado.

Pruebas de navegación: Estas pruebas nos sirven para comprobar, que podemos acceder a todos los menús y podemos realizar todos los cambios de configuración que queramos sin ningún tipo de error.

Pruebas de componentes: Estas pruebas comprueban de una forma algo mas general que tanto en modo un jugador como en modo multijugador, cumplen todas sus especificaciones dentro de los caso de usos descritos con anterioridad.

5.4.2. Plan de pruebas

Las pruebas a las que he sometido el videojuego son las siguientes:

Pruebas de Configuración:

- Prueba de cambio de Resolución.

  Probar que un cambio de resolución en cualquier momento tiene efectos durante todo el videojuego.
- Prueba de activación/desactivación de música y efectos de sonido.

  Probar si al des/activar la música o los efectos en el menú opciones tiene efectos a lo largo de la partida.
- Prueba de récords.

  Para comprobar los récords se ha probado las diferentes posibilidades que tenemos a

la hora de leer el fichero record.txt. Se ha leído con el fichero vacío, medio vacío y lleno. En todos los casos el resultado ha sido satisfactorio.

Pruebas de navegación:

Para comprobar que la navegación por los menús funcionan correctamente se ha navegado por cada uno de ellos de todas las formas posibles obteniendo un resultado satisfactorio.

Prueba de partida Un Jugador:

- Mover Personaje.

  Probar que el personaje se mueve de forma correcta y fluida por toda la pantalla sin superar los límites impuestos.
- Probar colisiones.

  Para probar las colisiones, he tenido que comprobar si las colisiones con los distintos objetos, enemigos , balas y elementos de las fase son correctas. Ha sido un proceso lento ya que las posibilidades de colisión eran muchas.
- Probar enemigos.

  Probar si cada uno de los movimientos de los enemigos responden con éxito.

Prueba de partida Dos Jugadores:

- Mover Personaje.

  Probar que los personajes se muevan de forma correcta y fluida por toda la pantalla sin superar los límites impuestos y si los controles responden bien para cada jugador.
- Probar funciones.

  Probar las diferentes funciones creadas para el control del jugador número dos.
- Probar enemigos.

  Probar si cada uno de los movimientos de los enemigos responden con éxito.

5.5. Herramientas utilizadas

KolourPaint

KolourPaint es un editor de imágenes para KDE libre y fácil de usar.

Su objetivo es ser fácil de entender y de usar, además, proporcionar un nivel de funcionalidad orientada hacia el usuario medio.

Es un excelente reemplazo o sustituto para MSPaint, sin embargo KolourPaint posee características más avanzadas.

Asimismo, KolourPaint puede ejecutarse en otros entornos aparte de KDE, tales como Xfce y GNOME,además también funciona sobre algunos manejadores de ventanas como Fluxbox y Blackbox.

GIMP

GIMP (GNU Image Manipulation Program) es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un programa libre y gratuito. Está englobado en el proyecto GNU y disponible bajo la Licencia pública general de GNU.
La primera versión de GIMP se desarrolló inicialmente en sistemas Unix y fue pensada especialmente para GNU/Linux. Existen versiones totalmente funcionales para Windows, para Mac OS X, y se incluye en muchas distribuciones GNU/Linux. También se ha portado a otros sistemas operativos, haciéndolo el programa de manipulación de gráficos disponible en más sistemas operativos. Se le puede considerar como una alternativa firme, potente y rápida a Photoshop para muchos usos, aunque no se ha desarrollado como un clon de él y posee una interfaz bastante diferente.

Audacity

Audacity es una aplicación informática multiplataforma libre, que se puede usar para grabación y edición de audio, fácil de usar, distribuido bajo la licencia GPL.

Es el editor de audio más difundido en los sistemas GNU/Linux.

Características:

- Grabación de audio en tiempo real;
- Edición archivos de audio tipo Ogg Vorbis, MP3, WAV, AIFF, AU y LOF;
- Conversión entre formatos de audio tipo;
- Importación de archivos de formato MIDI y RAW;
- Edición de pistas múltiples.
- Inversión de audio
- Agregar ecos al sonido

C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.

El nombre C++ fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre "C con clases". En C++, la expresión "C++" significa "incremento de C" y se refiere a que C++ es una extensión de C.

OpenOffice

OpenOffice es una suite ofimática libre (código abierto y distribución gratuita) que incluye herramientas como procesador de textos, hoja de cálculo, presentaciones, herramientas para el dibujo vectorial y base de datos.[4] Está disponible para varias plataformas, tales como Microsoft Windows, GNU/Linux, BSD, Solaris y Mac OS X. Soporta numerosos formatos de archivo, incluyendo como predeterminado el formato estándar ISO/IEC OpenDocument (ODF), entre otros formatos comunes. A febrero de 2010, OpenOffice soporta más de 110 idiomas.
OpenOffice.org tiene como base inicial a StarOffice, una suite ofimática desarrollada por StarDivision y adquirida por Sun Microsystems en agosto de 1999. El desarrollo de la suite está liderado por Oracle Corporation (inicialmente por Sun Microsystems), en conjunto con otras compañías como Novell, RedHat, RedFlag CH2000, IBM, Google, entre otras. El código fuente de la aplicación está disponible bajo la Licencia pública general limitada de GNU (LGPL) versión 3.
El proyecto y el programa son denominados «OpenOffice» de forma informal, aunque «OpenOffice.org» es el nombre oficial completo, ya que la denominación openoffice es una marca registrada en posesión de otra empresa.

Doxygen

Doxygen es un generador de documentación para C++, C, Java, Objective-C, Python, IDL(versiones Corba y Microsoft) y en cierta medida para PHP, C# y D. Dado que es fácilmente adaptable, funciona en la mayoría de sistemas Unix así como en Windows y Mac OS X. La mayor parte del código de Doxygen está escrita por Dimitri van Heesch.
Doxygen es un acrónimo de dox(document) gen(generator), generador de documentación para código fuente.

LibSDL es conjunto de librerías desarrolladas con el lenguaje C que proporcionan funciones básicas para realizar operaciones de dibujado 2D, gestión de efectos de sonido y música, y carga y gestión de imágenes. SDL es una abreviatura en inglés de Simple DirectMedia Layer.

WxDev-C++

WxDev-C++ es un entorno de desarrollo integrado libre basado en el popular Dev-C++.

Hay varias características nuevas no encontradas en el Dev-C++ original. Uno de ellas es un diseñador RAD visual que trabaja como el C++Builder para crear aplicaciones wxWidgets. La última versión, la 7.0, agrega soporte para compiladores de Microsoft. Se está trabajando para el soporte para otros compiladores.

Capítulo 6

Conclusiones y mejoras

6.1. Conclusión

Quizás mi expectativas eran la de desarrollar un videojuego mucho más completo, jugable y adictivo, sin embargo conforme íbamos avanzando en la producción, los problemas y limitaciones que encontrábamos eran mayores. Aún así, en términos globales creo que el videojuego ha cumplido con todos los objetivos propuestos desde el principio en mayor o menor medida.

El estilo del videojuego, típico de las maquinas recreativas, fue uno de las principales objetivos pudiendo afirmar que que he logrado alcanzar lo que buscaba.

El diseño gráfico posiblemente el trabajo más laborioso y largo del proyecto, y sin poder ser comparable a los gráficos de otros videojuegos, considero que gozan de un nivel aceptable.

Con respecto a la jugabilidad, es buena aunque mejorable. Probablemente si hubiese contado con la colaboración de alguien más o hubiera utilizado diseños de otros videojuegos libres, la jugabilidad y el motor gráfico hubiese sido mucho más potente.

También he tenido problemas con el tema de sonido. No sólo por encontrar el tipo de música acorde al estilo del videojuego sino a la hora de implementarlos pues iban con un pequeño retardo respecto al teclado.

Por último, destacar que todo el proyecto ha sido desarrollado usando software libre,
tanto las librerías (SDL) como los programas para editar imágenes(Gimp), sonido(Audacity), etc.

Como era de esperar, todo el contenido del proyecto será liberado bajo una licencia libre, la GPL v3.

6.2. Mejoras

El juego está pensado para que puedan incluirse nuevas mejoras.
Gracias a la licencia, se puede incluir nuevas fases, utilizando los enemigos disponibles, modificando parte mínima del código debido a su correcta modulación. También será posible añadir nuevos enemigos, objetos o personajes principales, eso si, primero es necesario diseñarlos, modificar el código y sin olvidarnos en ningún momento del autor del videojuego.
Además, si comprendemos bien el código y nos atrevemos a manipularlo podremos incluso cambiar la velocidad de los personajes, balas disparadas o el avance del scroll, haciéndolo más rápido o más lento.

Otro recurso muy útil son los ficheros de donde leemos los enemigos y objetos de cada fases. Con este recurso podremos personalizar la dificultad del videojuego. Según queramos podremos cambiar el número de enemigos y la posición de estos en la que aparecen. Lo mismo ocurre con los objetos, pudiendo incluir más o menos vidas, más sacos de dinero o más armas. Para este caso no tendremos que modificar nada del código simplemente los ficheros objetos y enemigos.

Bibliografía

Referencias de C++

[FundaC] F. Palomo Lozano, I. Medina Bulo, G. Aburruzaga García. Fundamentos de C++. ISBN-13: 978-8498280074

Referencias de SDL

[SDLTut] Alberto García Serrano. Programación de videojuegos con SDL.

http://www.agserrano.com ISBN-13: 978-8495836083

[SDLWik] The SDL community. SDL documentation Wiki.

http://www.libsdl.org/cgi/docwiki.cgi.

[AGAlba] Antonio G. Alba. Tutorial de libSDL para la programación de videojuegos.

http://softwarelibre.uca.es/wikijuegos.

[Pazera] Ernest Pazera. Focus On SDL. ISBN-13: 978-1592000302

[SDLWik] The SDL community. SDL documentation Wiki.

http://www.libsdl.org/download-1.2.php

Referencias de definiciones

[WikiEs] Wikipedia Community. Wikipedia, la enciclopedia libre.

http://es.wikipedia.org

[WikiEn] Wikipedia Community. Wikipedia, the free encyclopedia.

http://en.wikipedia.org

GPL

[GNU] http://www.gnu.org/licenses/gpl-3.0.html

Otras webs de interés

http://www.matton.es

http://www.jamendo.com

Apéndice A

Manual de usuario

A.1. Instalación

The Most Wanted está disponible para sistemas GNU/linux y Windows.
A continuación se detallan los pasos necesarios para instalar y poder disfrutar del juego en ambos sistemas.

La web donde encontraremos todo el materia necesario para su correcta instalación se encuentra en http://www.libsdl.org/.

A.1.1. Instalación en GNU/Linux

Copia de archivos
Descomprime el videojuego en la carpeta que desees.

Instalación de librerías
Para poder compilar y ejecutar el juego con éxito, es necesario instalar varias librerías y utilidades que el juego utiliza.

Imagen A.1. SDL: Conjunto de librerías. Extraída de http://www.tr0ll.net/libsdl/logo/sdl_logo_1600x1200.png

Tendrás que ejecutar el gestor de paquetes de tu distribución e instalar los siguientes paquetes (y sus propias dependencias).

- libsdl1.2debian: Paquete de librerías SDL
- libsdl1.2-dev: Paquete de librerías SDL para desarrollo. Importante para poder compilar nuestros programas.
- libsdl-image1.2: Paquete para poder trabajar con diferentes tipos de imágenes.
- libsdl-image1.2-dev: Paquetes para desarrollo.
- libsdl-mixer1.2: Paquete para trabajar con diferentes formatos de sonido. SDL trae sus propias funciones para trabajar con sonidos, pero con libsdl-mixer podremos trabajar mejor.
- libsdl-mixer1.2-dev: Paquetes para desarrollo.
- libsdl-ttf1.2: Paquete para trabajar con fuentes ttf.
- libsdl-ttf1.2-dev: Paquetes para desarrollo.
- gcc

Compilación

Para compilar nos situaremos desde la consola en la carpeta TheMostWanted y ejecutaremos make. Con esta orden el juego se compila y se enlaza, creando el ejecutable del juego. A priori el juego está compilado.

Ejecución

Después de la compilación, comprobamos que se ha creado un nuevo archivo llamado TheMostWanted en la carpeta TheMostWanted. Éste es el ejecutable del juego.
Ejecútalo desde la consola mediante la orden ./TheMostWanted o con doble clic de ratón sobre el archivo. Disfruta del juego.

Imagen A.2. Captura en Linux

A.1.2. Instalación en Windows

Para poder ejecutar el juego en Windows con éxito, es necesario instalar varias librerías y programas.

Instalación WxDev-C++

Este entorno de desarrollo es para editar el código del nuestro videojuego. No es necesario para la ejecución del juego.

Podremos descargarlo de la siguiente página web:

http://wxdsgn.sourceforge.net

Instalación de bibliotecas

Para poder hacer uso del videojuego tenemos que instalar unas serie de librerías necesarias.

Instalaremos estas cuatro bibliotecas: SDL(base), SDL_image, SDL_mixer y SDL_ttf.

El procedimiento de instalación es similar para todas ellas.

Instalar SDL (básico)

Debemos descomprimir SDL-1.2.8-win32.zip y SDL-devel-1.2.8-mingw32.tar.gz en diferentes carpetas, por ejemplo SDL_run y SDL_dev respectivamente.

El archivo contenido en SDL-1.2.8-win32.zip, SDL.dll, se debe copiar a la carpeta System dentro de la carpeta principal de Windows, generalmente la ruta completa es "c:\windows\system\".

El archivo SDL-devel-1.2.8-mingw32 incluye varios archivos y carpetas:

- Copie el contenido de la carpeta "lib" (4 archivos) en "c:\dev-cpp\lib\"
- Dentro de la carpeta include, copie la carpeta SDL completa en "c:\dev-cpp\include\".

Este mismo proceso lo repetiremos con el resto de las bibliotecas.

Ejecución

Después de la esto, simplemente bastará instalando el ejecutable TheMostWanted.exe dentro de la carpeta TheMostWanted. El proceso de instalación es sencillo, sólo indicar el destino de la carpeta a guardar. Por último, dentro de la carpeta ejecutamos el juego; TheMostWanted.exe.

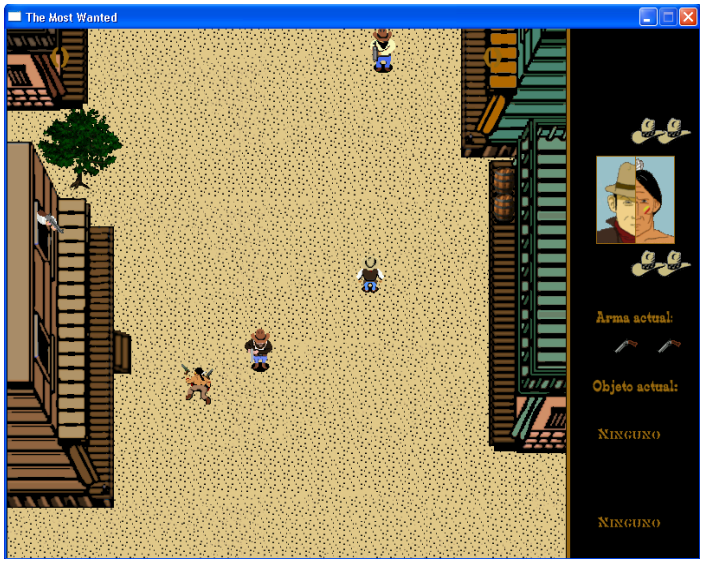


Imagen A.3. Captura en Windows xp.

A.2. Comienza el juego

A.2.1. Menú principal

Cuando inicies The Most Wanted, el sistema te redirigirá al menú principal donde tendrás cinco opciones posibles que elegir. Mediante los cursores de dirección selecciona tu elección deseada con el selector.

Para ejecutar una acción utilizamos la tecla "Aceptar" y para salir de un submenú y volver al menú principal la misma tecla sobre la opción Menú Principal.

El menú principal tendrá las siguientes opciones:

Modo individual - Juega una partida en modo individual contra la máquina.
Modo Dos jugadores - Juega una partida con otro jugador contra la máquina.
Opciones - Gestiona la música y los efectos del juego.
Récords - Conoce las máximas puntuaciones del juego.
Créditos - Conoce acerca de los creadores del juego.
Salir - Sale del juego.

A.2.2. Modo individual

Comienza una nueva partida. Selecciona un personaje e intenta superar los cinco niveles para completar el juego. Consigue la mayor puntuación retando a los mayores forajidos del salvaje oeste.

Durante la partida nos pondremos en la piel de Billy Banks, un famoso cowboy o Toro Salvaje, un vengativo indio de la tribu Sioux, que se dedican a ir capturando a los más peligrosos forajidos.
El scroll va avanzando automáticamente y solo podremos disparar en tres direcciones, hacia delante y diagonalmente a derecha e izquierda, usando para ello los botones de dirección y las teclas de

"Disparo". Haciendo combinaciones de esos tres botones podremos cambiar la dirección de nuestros disparos. Dependiendo del arma que poseamos el tipo de disparo será diferente.
Antes de enfrentarnos a los delincuentes más buscados de cada nivel, nos cruzaremos con gran parte de sus secuaces. Si uno de su disparos nos alcanza perderemos una vida, volviendo a comenzar la fase en curso.
A lo largo de la partida podremos conseguir distintos elementos que nos darán una mayor protección o facilidades, los cuales los podremos encontrar al disparar a los barriles.
Al final de cada uno de los 5 niveles que componen The Most Wanted nos encontraremos con un jefe final. Evidentemente estos personajes son más difíciles de eliminar ya que requieren una multitud de impactos para acabar con ellos. En buena parte, la gran dificultad del juego depende de estos enemigos.

A.2.3. Modo dos jugadores

Selecciona los personajes y comienza una nueva partida junto a un amigo.
Juntos intentareis derrocar al mayor numero de enemigos para conseguir superar las cinco fases que componen el juego.
Los elementos deberéis repartiros los o competir por ellos.
¿Quien de los dos conseguirá el mayor botín?

A.2.4. Elementos

Armas
Existen tres tipos de armas que iremos obteniendo a lo largo de la partida. El arma obtenida se asignará al arma secundaria del personaje.

- Revolver: Dispara una única bala.

- Dos revólveres: Dispara dos balas, una por cada revolver.

- Escopeta: Dispara 3 balas a la vez.

- Dos revólveres especiales: Dispara dos super balas, balas que nos dará una cadencia de disparo mayor, una por cada revolver.

Objetos

Dependiendo del objeto conseguido nuestro personaje obtendrá mayor protección, velocidad o facilidades.

- Caballo

  Nuestro personaje montará a lomos de un caballo, aumentando nuestra resistencia hasta dos disparos.

- Botas

  Podremos desplazarnos con una mayor velocidad en el caso de recogerlas durante un pequeño periodo de tiempo.

- Calavera

  En el caso de coger este objeto, acabaremos con todos los enemigos en pantalla de golpe.

- Herradura

  Podremos desplazarnos con total inmunidad en el caso de recogerla durante un pequeño periodo de tiempo.

- Sombrero

  Incrementará en una el numero de nuestras vidas.

- Dinero

  Incrementa la puntuación 25 puntos.

A.2.5. Interfaz

La interfaz durante la partida nos indica:

- El personaje que manejamos.

Imagen A.4. Foto de Billy Banks

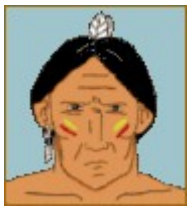


Imagen A.5. Foto de Toro Salvaje

- El número de vidas que poseemos en cada momento.

Imagen A.6. Vidas durante la partida

- El botín acumulado durante la partida.

Imagen A.7. Botín acumulado

- Arma actual que manejamos.

Imagen A.8. Arma actual: Revolver

- Objeto actual que poseemos.

Imagen A.9. Objeto actual: Ninguno

Para cambiar de arma pulsaremos "Salir" y accederemos al menú de pause, donde podremos cambiar de arma.

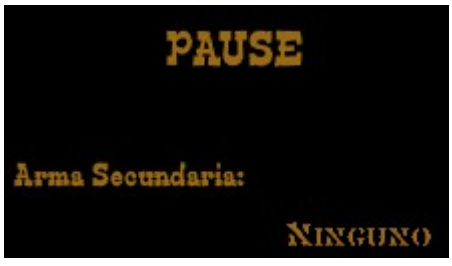


Imagen A.10. Arma secundaria: Ninguno

A.2.6. Menú opciones

El menú de opciones te permite conocer los controles para jugar al juego, así como des/activar la música y los efectos de sonido de este.

Opciones de pausa
Al pausar "Salir" la partida se pausa y muestra las opciones en una pantalla auxiliar. Selecciona esas opciones con el selector y la tecla "Aceptar".

Arma secundaria
Elige este arma

Continuar
Reanuda el juego.

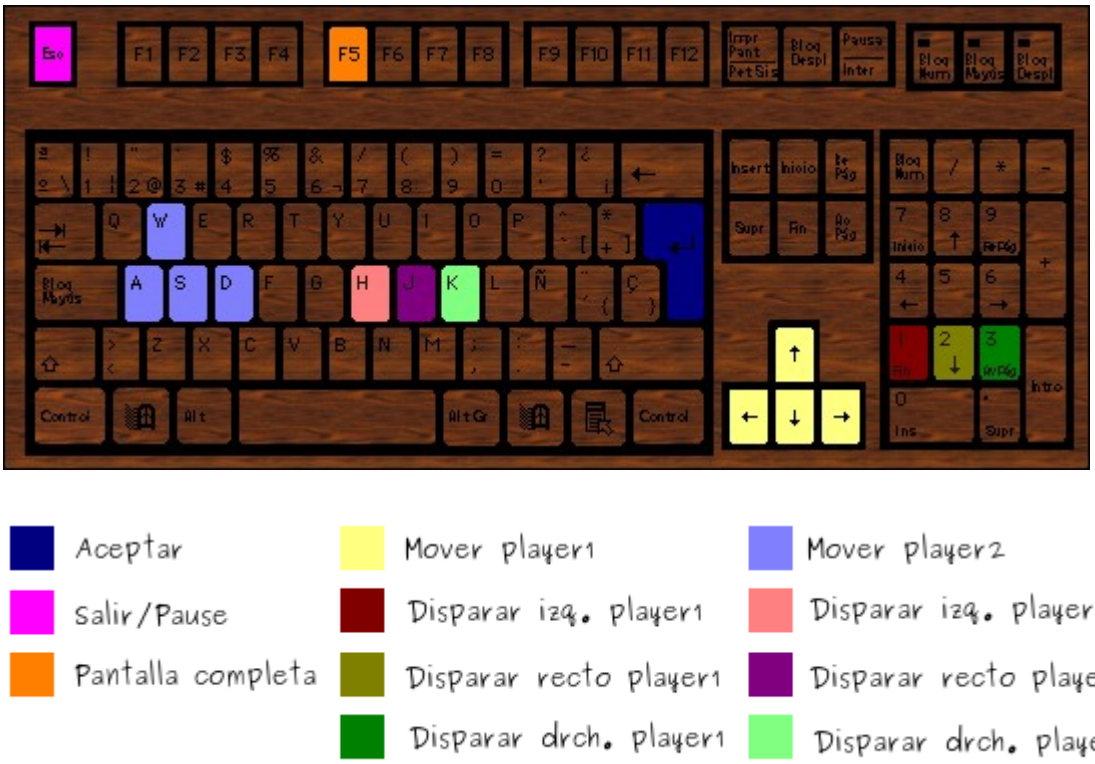
Menú principal
Abandona la partida en curso y regresa al menú principal.

Salir
Finaliza la partida actual y abandona el juego.

A.2.7. Récords

Muestra las cinco mejores puntuaciones registradas en el juego.

A.2.8. Controles y movimientos



A.2.9. Créditos

Programación y Diseño

José Ignacio Mateo Cruzado

Audio

Música de libre distribución.

Autor: Prolific Arts.

A.2.10. Diagrama de Menús

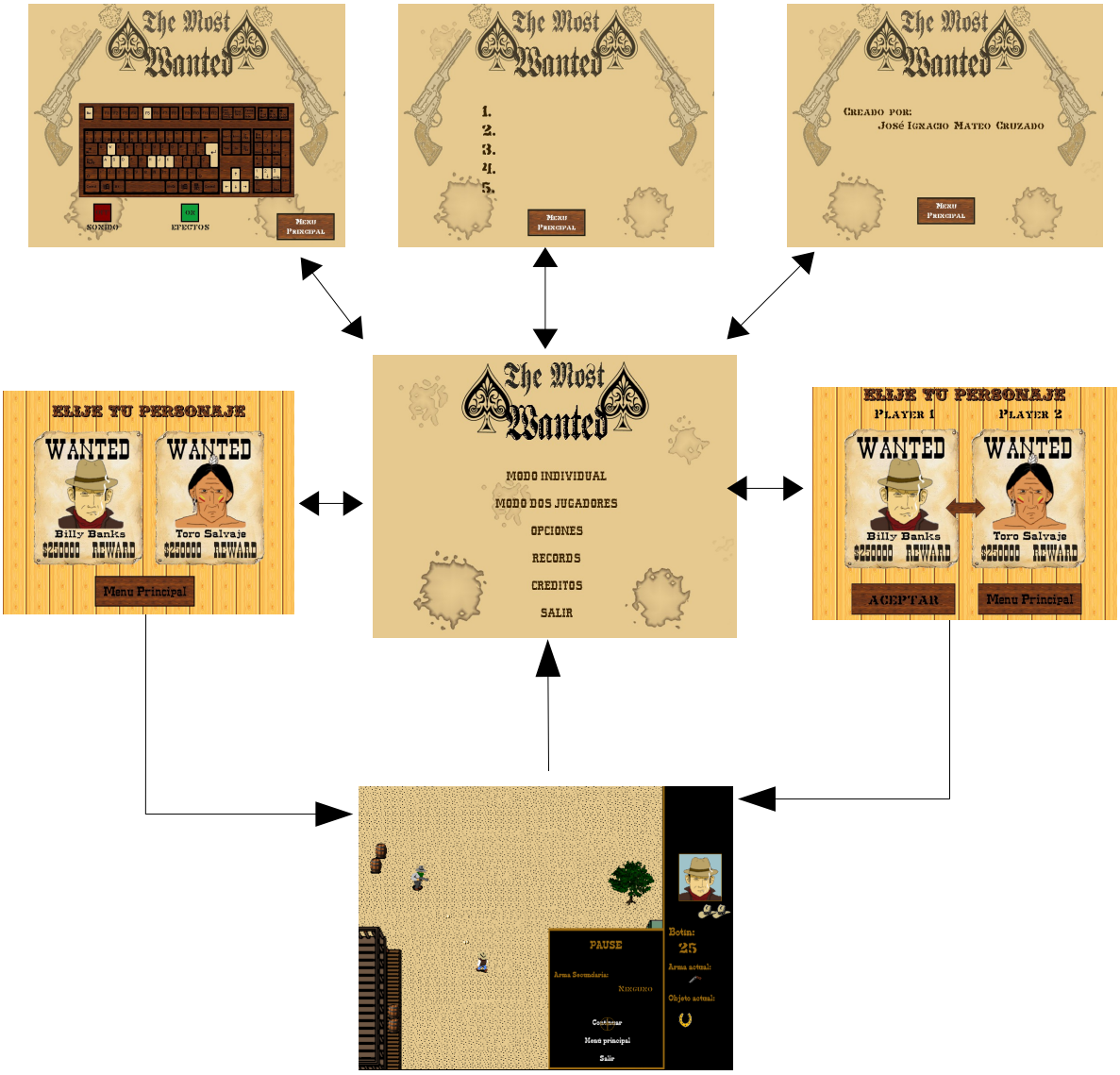


Imagen A.11. Diagrama de navegación

Licencia GPL v3

GNU GPL

La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente sus siglas del inglés GNU GPL, es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Existen varias licencias "hermanas" de la GPL, como la licencia de documentación libre de GNU (GFDL), la Open Audio License, para trabajos musicales, etcétera, y otras menos restrictivas, como la MGPL, o la LGPL (Lesser General Publical License, antes Library General Publical License), que permiten el enlace dinámico de aplicaciones libres a aplicaciones no libres.

Validez Legal

La licencia GPL, al ser un documento que cede ciertos derechos al usuario, asume la forma de un contrato, por lo que usualmente se la denomina contrato de licencia o acuerdo de licencia. En los pases de tradición anglosajona existe una distinción doctrinal entre licencias y contratos, pero esto no ocurre en los pases de tradición civil o continental. Como contrato, la GPL debe cumplir los requisitos legales de formación contractual en cada jurisdicción. La licencia ha sido reconocida por juzgados en Alemania, particularmente en el caso de una sentencia en un tribunal de Munich, lo que indica positivamente su validez en jurisdicciones de derecho civil.

GPL versión 3

A finales de 2005, la Free Software Foundation anunció que trabajaba en la versión 3 de la licencia GPL, cuyo primer borrador fue presentado para su discusión pública el 16 de enero de 2006.

La discusión se alargó 18 meses, habiendo sido publicados cuatro borradores. Finalmente, la versión oficial fue publicada el día 29 de junio de 2007 y es accesible a través del Portal de GNU.[4] La nueva versión contempla los siguientes aspectos:

- Las diversas formas en que alguna persona podría quitar libertades a los usuarios.
- Prohibir el uso de software libre en sistemas que utilizan la llamada Gestión de derechos digitales o DRM, sistema criticado por la comunidad del software libre.
- Resolver ambigüedades y aumentar la compatibilidad de GPLv3 con otras licencias.
- Facilitar su adaptación a otros países.
- Incluir cláusulas que defiendan a la comunidad de software libre del uso indebido de las patentes de software.
- Mostrar usuarios registrados.

Compatibilidad

Muchas licencias libres como MIT License, y GPL, son compatibles con la GPL (ver lista completa en los enlaces externos). Esto significa que se puede combinar código licenciado bajo GPL con código que se encuentre bajo una licencia compatible sin ningún tipo de problema, ya que el código resultante debe satisfacer las condiciones de ambas licencias. Sin embargo, otras licencias calificadas como libres no son compatibles con la GPL, lo que dificulta la reutilización de código; por ello se incita a los desarrolladores de software libre a licenciar su código bajo GPL o licencias compatibles con la GPL, pudiendo aprovecharse de las ventajas que ello conlleva.

Existe una proliferación de licencias libres que añaden algún tipo de condición a otra licencia compatible con la GPL (en la que se basan), haciendo difícil determinar si la nueva licencia es compatible o no con la GPL. Esto obliga a recurrir a expertos en la materia, que era lo que en un principio se pretendía evitar, por lo que no se recomienda esta práctica.

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the

software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a

"modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the

executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below.

Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes

into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially,

and only if you received the object code with such an offer, in accord with subsection 6b.

- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User

Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License

with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your

rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the

work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the

benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently

authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a

warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify
> it under the terms of the GNU General Public License as published by
> the Free Software Foundation, either version 3 of the License, or
> (at your option) any later version.
>
> This program is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
> GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License
> along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.