UNIVERSIDAD AUTONOMA METROPOLITANA

INGENIERIA EN
COMPUTACION

DIVICION DE CIENCIAS BASICAS E INGENIERIA

PROGRAMACION DE UN WAR-GAME 2D PARA PLATAFORMAS MÓVILES Y TOUCH SCREEN CON J2ME

PROYECTO TERMINAL

P R E S E N T A:

LUIS MIGUEL CABRAL GUZMAN*
Matricula 205300732

TUTOR:
M. EN C. PABLO LEYVA HUGO
No. Económico 17060

Unidad Azcapotzalco Enero 2012

A mis padres:
Luis Cabral Paredes y
Patricia Guzmán Ledesma

A:
Azalea Karina Nájera Hidalgo
Ximena Amelie Cabral Nájera;
mi familia

♥

La gloria es efímera, pero la obscuridad dura eternamente.

Napoleón Bonaparte I

Si buscas resultados distintos, no hagas siempre lo mismo.

Albert Einstein.

Tu tiempo es limitado, de modo que no lo malgastes viviendo la vida de alguien distinto. No quedes atrapado en el dogma, que es vivir como otros piensan que deberías vivir. No dejes que los ruidos de las opiniones de los demás acallen tu propia voz interior. Y, lo que es más importante, ten el coraje para hacer lo que te dicen tu corazón y tu intuición.

Steve Jobs

Agradecimientos

Primeramente agradezco a Dios por este gran destino que sé me ha guardado y que ahora me ofrece, por las oportunidades que he tenido y aprovechado, por la salud y las pruebas que no olvidare y que me han formado, sinceramente te doy gracias.
En cada una de las etapas de este trabajo hubo personas a las cuales deseo agradecer especialmente.
Durante la realización de este trabajo: el apoyo fundamental de mi novia Karina Nájera Hidalgo "Kary", sin el cual este trabajo no hubiese sido tan agradable; a lo largo de todo el tiempo ¡Gracias! Agradezco a mi hija Ximena Amelie Cabral Nájera, por todos los momentos de alegría y satisfacción que me ha brindado.
No quiero dejar de agradecer a mis padres Luis Cabral Paredes y Patricia Guzmán Ledesma su apoyo y sacrificio que luego de lustros de cuidados y educación han rendido frutos en mi formación profesional y en mi formación como ser humano, mismos que me han permitido sin lugar a dudas realizar mis estudios de ingeniería. Así mismo, agradezco a mi hermano Luis Angel Cabral Guzmán por todo el apoyo que me ha brindado durante mis estudios, desde lo personal, hasta los apoyos académicos y puntos de vista que sin lugar a dudas han hecho de este trabajo algo mejor, al igual que agradezco a mi hermana Isabel Cabral Guzmán que con su propia vida me obliga a seguir adelante.
Agradezco también a toda mi familia así como a mis amigos que conservo desde la secundaria hasta la universidad por su cariño, aprecio y amistad.

A todas las personas y entidades mencionadas ¡Muchas gracias! ¡Gracias a todos ustedes he logrado alcanzar esta importante meta!

Índice general

1. Introducción **1**

1.1. Objetivos. 1
1.1.2. Objetivos general. 1
1.1.3. Objetivos particulares 1
1.2. Marco teórico. 2
1.2.1 Justificación. 2
1.2.2 Metodólogia. 3

2. Antecedentes **4**

2.1. Proyectos relacionados. 4
2.2. Sector comercial. 5
2.2.1. Plataforma futura de los videojuegos 6

3. Descripción técnica **7**

3.1. Arte. 7
3.2. Lenguaje de desarrollo. 8
3.2.1. MIDP y CLCD 9
3.2.2. Ciclo de vida de un MIDlet. 9
3.3. Game loop 11
3.3.1. Estados del videojuegoGame loop 12
3.3.1.1. Estado: Presentacion 13
3.3.1.2. Estado: Menú Principal 15
3.3.1.3. Estado: MENU START GAME. 15
3.3.1.4. Estado: SHOW_CREDITS. 16
3.3.1.5. Estado: TUTORIAL. 16
3.4. Algoritmos de Inteligencia Artificial. 17
3.4.1. Creación de rutas. 17
3.4.2. Comportamiento del PNJ 18
3.5. Almacenamiento y recuperación de una partida. 21
3.5.1. Almacenamiento 21
3.5.2. Recuperación. 22

Conclusiones **23**

Bibliografía **24**

Índice de tablas y figuras

1.1. Primera parte del proyecto. 3
1.2. Segunda parte del proyecto. 3

2.1. Plataformas para el desarrollo de aplicaciones. 5
2.2. Ingresos de software para videojuegos portables. 6

3.1. Sprites en J2ME. 8
3.2. Componentes de J2ME y su relación con otras tecnologías Java 8
3.3. Ciclo de vida de un MIDlet 9
3.4. Métodos principales del MIDlet. 10
3.5. Los 4 estados del *game loop*. 11
3.6. Herencias de la clase "*GameCanvas*". 11
3.7. Origen de coordenadas en dispositivos móviles. 12
3.8. Ejecución de cada secuencia del *game loop*. 13
3.9. Métodos encargados de gestionar los *gestos* 14
3.10. Pantallas de inicio del software 14
3.11. Pantalla del "Menú de Opciones" y "Menú Principal". 15
3.12. Pantalla del "Menú Secundario" y "Juego" 16
3.13. Pantalla de "Créditos" 16
3.14. Pantalla del "Tutorial" 17
3.15. Rutas 18
3.16. Diagrama de la Inteligencia Artificial. 19
3.17. Pantalla de "Guardar una partida" y ""Eliminar una partida guardada". 21
3.18. Proceso por el cual se almacena/recupera una partida 22

Capítulo 1

Introducción

La computadora, concebida hace mucho y creada para procesar números ha invadido las actividades humanas y a asumido una nueva personalidad con capacidad grafica y de sonido.
Los juego de video, también conocidos como software de entretenimiento o videojuegos constituyen, hoy en día, la industria más importante dentro del mundo del entretenimiento, superando a la industria cinematográfica.
Una de las actividades más interesantes, desde el punto de vista del desarrollo de software, son los videojuegos, caracterizados por el uso de tecnologías de punta y liderando en investigación, en campos tan importantes como la computación grafica, interfaces, usabilidad, simulaciones, e inteligencia artificial entre otros.
A su vez se han convertido en una forma de expresión cultural muy fuerte, con personajes y figuras mundialmente reconocidas y estilos definibles de acuerdo al modo de juego.
Aunque aparentemente, los videojuegos, parezcan "tonterías para jugar y nada más", suponen muchas veces un reto y contienen una complejidad superior comparados con aplicaciones cotidianas como una aplicación de "escritorio" o "web", ya que los videojuegos ya sean "stan-alone" o "web" contienen además procesamiento grafico e interfaces superiores.
Hace algunos años nadie tenía aceleradores 3D, y ahora, relativamente, todas las computadoras modernas tratan de usar las capacidades de las tarjetas aceleradoras.
Los videojuegos juegan un papel importante que hace evolucionar el hardware y el software de las maquinas.

1. Objetivos

1.1. Objetivo general

Este trabajo tiene por objetivo Programar un videojuego 2D por turnos, del tipo estrategia militar pura o también conocido como War-Games, para plataformas móviles con interfaz touch screen.

1.2. Objetivos generales

* Describir la historia de una guerra a desarrollarse en el juego.
* Editar los componentes visuales y sonoros del juego (personajes, escenarios, elementos *GU I* [2]y *HUD*[3] y sonidos).
* Programar el control de los componentes del juego, (personajes, elementos *GUI* y *HUD* y sonidos), mediante la interfaz touch screen por medio de *J2ME*[4].

sistema de comportamiento del *PNJ*.

[1] Pantalla Táctil
[2] Interfaz grafica de usuario [10]
[3] En informática, principalmente en los videojuegos, se llama *HUD* (del inglés: "*Head-Up Display*") a la información que en todo momento se muestra en pantalla durante la partida, generalmente en forma de iconos y números. El *HUD* suele mostrar el número de vidas, puntos, nivel de salud y armadura, mini-mapa, etc., dependiendo del juego. [11]
[4] Java 2, Edición micro [12]
[5] Un personaje no jugador, término a menudo abreviado con la sigla *PNJ*, es un personaje controlado por un "director de juego" en el curso de una partida de rol. En los videojuegos de rol, un *PNJ* es generalmente parte del programa, y no es controlado por un humano. [11]

1.2. Marco teórico

Uno de los principales retos asociados con el desarrollo de sistemas de entretenimiento, como lo es un videojuego, es la inteligencia artificial para cada uno de los personajes encargados de ambientar el juego, una búsqueda de rutas para establecer los movimientos de cada elemento dinámico del juego y la misma organización del código fuente en el que se programa el juego ya que esto influye en la complejidad de cómo entender el videojuego a desarrollar desde un punto de vista abstracto para el programador.
Existen diversas técnicas para establecer un mecanismo adecuado para generar tanto la inteligencia artificial, la búsqueda de rutas y la animación para cada personaje del videojuego, cada uno de ellos poseen una cierta complejidad, desde el punto de vista computacional, tanto en su elaboración como en el consumo de recursos.
De igual manera, existen varios lenguajes de programación con los que se puede desarrollar un videojuego, prácticamente en todos, estos van desde juegos programados en lenguaje ensamblador, pasando por el famoso juego *PAC-MAN*[6], hasta los modernos juegos de última generación.
En este trabajo se menciona la descripción en general del trabajo realizado y los resultados obtenidos al desarrollar el videojuego, que fue el objetivo de este proyecto.

1.2 1. Justificación.

En la UAM-Azcapotzalco se han desarrollado diversos proyectos por parte de los alumnos tratando temas sobre los videojuegos; pasando del análisis sobre cómo la inteligencia artificial puede basarse en el aprendizaje hasta implementar interfaces para usuario en dispositivos móviles. Además se han efectuado congresos y presentaciones tratando el tema de los videojuegos.
Un ingeniero en computación debe de tener la visión y la convicción de que su área de trabajo no se limita solamente a computadoras de escritorio o sistemas informáticos convencionales. La variedad de dispositivos en los que puede actuar un ingeniero en computación es muy amplia y una de estas áreas es la del cómputo móvil y la creación de videojuegos.
El cómputo móvil cambiará la forma en que se realizarán las actividades laborales, académicas, de investigación y entretenimiento, en el caso de los videojuegos, como en su momento lo hizo el cómputo, lo que demandará a las organizaciones de toda índole migrar de los servicios electrónicos a los móviles. El mercado laboral demandará investigadores y profesionales capaces de enfrentar y resolver los retos que plantea la computación móvil. [4]
Los videojuegos hoy en día tienen diversas plataformas que ya no dependen solamente de las grandes consolas. Ahora se pueden desarrollar juegos para teléfonos móviles como el iPhone, iPod touch, iPad, entre otros y hay que pensar en ellos pues representan una gran oportunidad de desarrollo para el país. [5]
Pensar que los videojuegos están aislados de la educación y que no son temas de desarrollo e interés para una universidad es un error ya que éstos favorecen el desarrollo de determinadas habilidades de atención, concentración especial, resolución de problemas, creatividad, etc. [6] Se aseveraba que los videojuegos provocaban efectos adversos a nivel intelectual, sin embargo, todos los estudios realizados hasta la fecha coinciden en la ausencia de dichos efectos adversos, de este modo se está en condiciones de afirmar que los jugadores de videojuegos suelen ser sujetos de mayor nivel intelectual que los no jugadores, a la vez que presentan diferencias en su estilo de procesar la información. No se debe buscar una relación causal entre el tipo de videojuego y el nivel intelectual; de existir tal relación, posiblemente sea a la inversa [7]. Los juegos de estrategia, además de entretener, son un tipo de juego en los que la capacidad de razonar y la memoria juegan un papel más importante que la suerte, un factor determinante en otros juegos, y por lo tanto, estimulan el intelecto [8].
Debido a esto, se propuso programar un videojuego de estrategia del tipo *War-Game* en una plataforma móvil, con una interfaz *touch screen*, para brindar un mejor control de los elementos del sistema.
Un ingeniero en computación posee todas las habilidades para poder realizar un proyecto de esta índole, ya que se necesita conjuntar conocimientos sobre programación orientada hacia dispositivos móviles, inteligencia artificial, conocimientos matemáticos y diseño de algoritmos. Estas conocimientos se obtienen a lo largo de la formación como ingeniero en computación, cursando asignaturas tales como: Programación Orientada a Objetos, Análisis y Diseño de Algoritmos, Inteligencia Artificial, así como todas aquellas que involucran matemáticas de cualquier tipo.

[6] *PACMAN* es un videojuego arcade creado por el diseñador de videojuegos *Toru Iwatani* de la empresa *Namco*, y distribuido por *Midway* Games al mercado estadounidense a principios de los años 1980.

A futuro, como continuación de este proyecto, se procedería a implementar diversas funciones para este juego tales como: la posibilidad de multijugador por medio comunicación *Bluetooth*[7] o comunicación *HTTP*[8] para usuarios que deseen jugar en línea. Otra opción sería usando *HTTPS*[9], incrementando la seguridad en el envió de la información. Otra opción para una continuación de este proyecto podría ser el modelado 3D, parcial o totalmente del juego.

1.2.2. Metodología.

En primer lugar se hizo una investigación partiendo de conocimientos previos para poder desarrollar y detallar las tareas que dieron forma a este proyecto, para este propósito han sido las que tienen como meta el desarrollo de este sistema para plataforma móvil basado en *J2ME*.
Con base teórica y en trabajos anteriores basados tanto en videojuegos, dispositivos móviles y algoritmos se desarrollaron los algoritmos de búsquedas de caminos e inteligencia artificial, basados más que nada en grafos.
Se decidió el uso del algoritmo A*(A-start), una modificación del algoritmo conocido como *Dijkstra* por los buenos resultados obtenidos en proyectos previos relacionados, desarrollados durante mis estudios de licenciatura. También se eligió implementar la inteligencia artificial mediante un grafo describiendo cada estado posible en los cuales se podría encontrar el *PNJ*.
Se implementaron estos algoritmos en el software de entretenimiento, que además de contener todas las características y obtención de los parámetros que estos algoritmos requieren, posee módulos para la animación, control del sonido, control de vibración y por ultimo lectura y escritura de archivos para poder guardar avances del videojuego.
A continuación se muestran los calendarios de trabajo, en donde se resumen las actividades realizadas a lo largo de este proyecto:

Para la primera parte del proyecto, correspondiente a la U.E.A. Proyecto Terminal de Ingeniería en Computación

Tabla 1.1: Primera parte del proyecto

ACTIVIDAD	Numero de actividades			
	1	2	3	4
Recopilación y Edición de componentes visuales (*sprites*[10], elementos *GUI* y *HUD*).				
Recopilación y Edición de sonidos.				
Programar el control de los elementos del juego mediante la interfaz *touch screen*.				
1ra. Parte del Reporte Final (posible borrador).				

Para la última parte del proyecto, correspondiente a la U.E.A. Proyecto Terminal de Ingeniería en Computación II

Tabla 1.2: Segunda parte del proyecto

ACTIVIDAD	Numero de actividades				
	1	2	3	4	5
Diseñar y programar el sistema de comportamiento del *PNJ*.					
Integrar el comportamiento del *PNJ* al videojuego.					
Elaborar pruebas finales.					
Corrección de errores, en su caso.					
2da. Parte del Reporte Final.					

[7] La tecnología inalámbrica Bluetooth es una tecnología de corto alcance para las comunicaciones destinadas a sustituir los cables de conexión portátil y / o dispositivos fijos, manteniendo altos niveles de seguridad. Las características clave de la tecnología Bluetooth son la robustez, bajo consumo y bajo costo. La especificación *Bluetooth* define una estructura uniforme para una amplia gama de dispositivos para conectarse y comunicarse entre sí.[14]
[8] *HTTP* es un protocolo estándar de transferencia de datos, *(Hypertext Transfer Protocol HTTP)* por sus siglas en ingles. [9]
[9] *HTTPS* es el protocolo estándar que la mayoría de los navegadores utilizan para comunicarse de forma segura. Es lo mismo que *HTTP*, salvo que la comunicación se realiza mediante una conexión segura utilizando el protocolo *SSL*. [9]
[10] Elemento visual que puede ser una única imagen o componerse de varias que se van alternando hasta crear una animación. [11]

Capítulo 2

Antecedentes

La historia de los videojuegos tiene su origen en la década de 1940 cuando, tras el fin de la Segunda Guerra Mundial, las potencias vencedoras construyeron las primeras supercomputadoras programables como el *ENIAC*[11], de 1946 Los primeros intentos por implementar programas de carácter lúdico no tardaron en aparecer, y se fueron repitiendo durante las siguientes décadas. Los primeros videojuegos modernos aparecieron en la década de los 60, y desde entonces el mundo de los videojuegos no ha cesado de crecer y desarrollarse con el único límite que le ha impuesto la creatividad de los desarrolladores y la evolución de la tecnología. En los últimos años, se asiste a una era de progreso tecnológico dominada por una industria que promueve un modelo de consumo rápido donde las nuevas superproducciones quedan obsoletas en pocos meses, pero donde a la vez un grupo de personas e instituciones han iniciado el estudio formal de la historia de los videojuegos.

Al igual que ocurriera con el cine y la televisión, el videojuego ha logrado alcanzar en apenas medio siglo de historia el estatus de medio artístico, y semejante logro no ha tenido lugar sin una transformación y evolución constante del concepto mismo de videojuego y de su aceptación.

Nacido como un experimento en el ámbito académico, logró establecerse como un producto de consumo de masas en tan sólo diez años, ejerciendo un formidable impacto en las nuevas generaciones que veían los videojuegos con un novedoso medio audiovisual que les permitiría protagonizar en adelante sus propias historias [20].

El presente capítulo se dedica a presentar brevemente los antecedentes sobre el tema que el trabajo presentado aborda. Se divide en dos partes:

Primeramente se muestran algunos trabajos relacionados realizados por alumnos de la universidad y por ultimo una presentación muy breve sobre este tema en el sector comercial.

2.1. Proyectos relacionados

Se pueden mencionar dos proyectos terminales que de algún modo se relacionan a éste, éstos son:

1. *"Implementación de una interfaz de usuario de la plataforma de educación a distancia Moodle utilizando asistentes personales (PDAs)"* [1], el cual se relaciona con mi proyecto en el área de la computación móvil al implementarse en *PDAs*[12], sin embargo, no hace uso del lenguaje *J2ME*, sino que modifica directamente el código de la plataforma Moodle[13] para lograr que sea visible a través de un *PDA*.
2. *"Máquina de aprendizaje en un juego de ajedrez [2]",* el cual se relaciona con mi proyecto respecto a la inteligencia artificial del sistema, se diferencia en que este sistema no es para plataformas móviles.
3. *"Plataforma de trabajo para desarrollo de aplicaciones en dispositivos móviles y, como caso de estudio, juego de ajedrez" [21],* el cual se relaciona con mi proyecto respecto a su desarrollo para dispositivos móviles.
4. *"Implementación de motor de juego para juegos tipo Tower Defense" [22],* el cual se relaciona con mi proyecto respecto a que desarrolla un juego para plataforma pc, se diferencia principalmente por tratarse para de un juego para PC y por estar desarrollado en *XNA*[14].

[11] ***ENIAC*** es un acrónimo de ***E****lectronic* ***N****umerical* ***I****ntegrator* ***A****nd* ***C****omputer* (Computador e Integrador Numérico Electrónico), utilizada por el Laboratorio de Investigación Balística del Ejército de los Estados Unidos.

[12] *PDA*, del inglés ***P****ersonal* ***D****igital* ***A****ssistant (Asistente Digital Personal)*

[13] *Moodle* es un Sistema de Gestión de Cursos de Código Abierto (*Open Source Course Management System, CMS*), conocido también como Sistema de Gestión del Aprendizaje (*Learning Management System, LMS*) o como Entorno de Aprendizaje Virtual (*Virtual Learning Environment, VLE*). Es una aplicación web gratuita que los educadores pueden utilizar para crear sitios de aprendizaje efectivo en línea.[13]

[14] Es un conjunto de herramientas con un entorno de ejecución proporcionado por Microsoft que facilita el desarrollo de juegos de ordenador y de gestión.

2.2. Sector comercial.

En poco tiempo se ha visto la tendencia de los dispositivos móviles a incorporarse al negocio del entretenimiento mediante videojuegos y actualmente estas aplicaciones se pueden descargar directamente al dispositivo mediante tiendas virtuales propiedad de las mismas compañías fabricantes de los ya conocidos "*Smartphone*[15]", "*Tablets*[16]", etc. y ya son muy conocidas varias de estas tiendas, ejemplos de ellas pueden ser: Apple Store, Android Market, .BlackBerry App World entre otras, solo para listar a las más famosas.
La compañía *Millennial Media* ha publicado un estudio en el que se analizan los ingresos de algunas de las distintas tiendas virtuales para los dispositivos móviles, concluye que tanto *Android*[17] como *iPhone*[18] quedan parejos en cuanto a ingresos. En el informe correspondiente al mes de mayo del 2011 el **45%** de los ingresos fueron generados por la plataforma de *Apple*[19], mientras que el *Android* se sitúa muy cerca con un **43%**. La tercera en esta lista es la plataforma de *BlackBerry*[20], con un **9%**, mientras que en el 3% restante se engloban el resto, entre ellas *Windows Phone 7*[21], *Symbian*[22] y *Palm*[23]. [23].

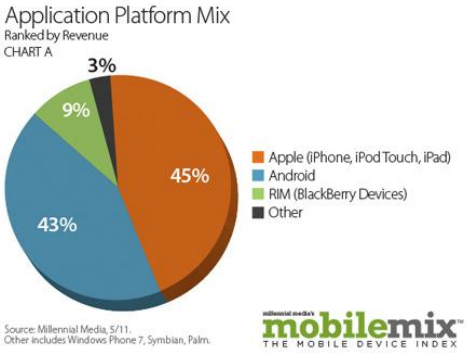

Figura 2.1: Resultados de una encuesta realizada por *Millennial Media*, con algunos resultados interesantes. EL 43% afirmo que desarrolla aplicaciones para *Android*, aunque en otras preguntas los resultados son claramente favorables para *iOS*[24], siendo considerada como una mejor plataforma para el desarrollo y donde más ingresos se obtienen. [23]

[15] El teléfono inteligente (*smartphone* en inglés) es un término comercial para denominar a un teléfono móvil que ofrece más funciones que un teléfono celular común. Casi todos los teléfonos inteligentes son móviles que soportan completamente un cliente de correo electrónico con la funcionalidad completa de un organizador personal. Una característica importante de casi todos los teléfonos inteligentes es que permiten la instalación de programas para incrementar el procesamiento de datos y la conectividad. Estas aplicaciones pueden ser desarrolladas por el fabricante del dispositivo, por el operador o por un tercero.
[16] Una *tablet* PC o tableta ('ordenador personal en tableta') es una computadora portátil con la que se puede interactuar a través de una pantalla táctil o multitactil. Para trabajar con la computadora, el usuario puede utilizar una pluma también conocida como *stylus* o los dedos, sin necesidad de teclado físico ni ratón.
Hoy en día el termino *tablet* se usa principalmente para referirse a los aparatos también conocidos como *gadgets* controlados principalmente Mediante una pantalla táctil, pero no con la intención principal de ejecutar algún sistema operativo de computadora en general sino para la ejecución en sí de distintas aplicaciones compatibles con el sistema operativo que el dispositivo posee.
[17] *Android* es un sistema operativo basado en el núcleo del sistema operativo *Linux* diseñado originalmente para dispositivos móviles, tales como teléfonos inteligentes, pero que posteriormente se expandió su desarrollo para soportar otros dispositivos tales como *tablets*, reproductores *MP3*, *netbook*, PC, televisores, lectores de libros electrónicos e incluso, se han llegado a ver en, microondas y lavadoras.
[18] *iPhone* es una familia de teléfonos inteligentes multimedia con conexión a internet, pantalla táctil capacitiva y escasos botones físicos, diseñado por la compañía *Apple Inc*.
[19] *Apple Inc*. es una empresa multinacional estadounidense con sede en *Cupertino*, California, que diseña y produce equipos electrónicos y software.
[20] *BlackBerry* es una línea de teléfonos inteligentes que integran el servicio de correo electrónico móvil. *BlackBerry* fue desarrollado por la compañía canadiense llamada *Research In Motion (RIM)*. Aunque incluye aplicaciones típicas (libreta de direcciones, calendario, listas de tareas, etc., así como capacidades de teléfono en los modelos más nuevos). Es fundamentalmente conocido por su capacidad para enviar y recibir correo electrónico de Internet accediendo a las redes móviles de compañías de teléfono celular que brindan este servicio. Usa el sistema operativo *BlackBerry OS*.
[21] *Windows Phone 7* es un sistema operativo desarrollado por la empresa multinacional estadounidense *Microsoft*, como sucesor de la plataforma *WindowsMobile*. Está pensado para el mercado de consumo generalista en lugar del mercado empresarial.
[22] *Symbian* es un sistema operativo que fue producto de la alianza de varias empresas de telefonía móvil, entre las que se encuentran *Nokia*.
[23] *Palm OS* es un sistema operativo que fue creado por *PalmSource. Inc* para computadoras de mano (*PDAs*).
[24] *iOS* (anteriormente denominado *iPhone OS*) es un sistema operativo móvil desarrollado por *Apple Inc*.

2.2.1. Plataforma futura de los videojuegos

Cada usuario poseedor de algún *Smartphone, tablet, etc.* ha experimentado el dinamismo y la explosión que actualmente los videojuegos, y en general cualquier aplicación, en los dispositivos móviles experimentan. Los fabricantes tradicionales de videojuegos para consolas enfrentan un nuevo problema por una nueva perspectiva de movilidad que están teniendo los videojuegos y la evolución acelerada del hardware y del software de los dispositivos móviles, abriendo así nuevas plataformas en las que es posible el desarrollo de aplicaciones de alta calidad. Esto sitúa a estos fabricantes tradicionales en un dilema: unirse a la tendencia móvil y arriesgar su negocio o esperar que la rica experiencia del videojuego en casa triunfe ante el movimiento [24].
Actualmente se considera que cualquier dispositivo móvil inteligente integrado a las compañías de telecomunicaciones generan un gran flujo de efectivo, sin embargo, desde la perspectiva de un desarrollador de sistemas de software esto supone un modelo nuevo de negocio y muchas ganancias al lograr colocar nuevas aplicaciones en los dispositivos móviles que inundan el mercado actual, considerando la potencia de algunos de estos nuevos dispositivos, el desarrollo de aplicaciones, para este propósito, se hace más fáciles y de mayor calidad.
Latinoamérica será una de las regiones con más crecimiento en el consumo de Internet móvil incluyendo la descarga de *Apps*[25], juegos y el uso de redes sociales, cada vez más gente reemplazará su teléfono móvil básico por uno dispositivo inteligente, a tal punto que todo el mundo usará teléfonos inteligentes. Este cambio radical no significa que las compañías tradicionales de videojuegos tengan que salir del mercado, pero sí que tienen que ingeniárselas cada vez más desde el surgimiento de los *smartphone*.
Los juegos en línea fueron los primeros en destronar del máximo lugar a las consolas de videojuegos en el podio del divertimento digital. Pero, esa caída no comenzó hace mucho, parece haberse profundizado de la mano de los dispositivos móviles [25].

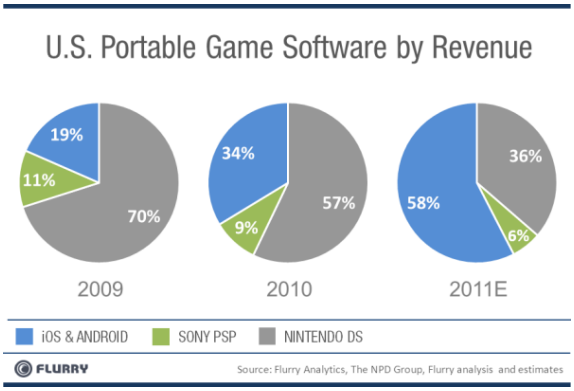

Figura 2.2: El gráfico muestra el porcentaje de los ingresos generados en EE.UU. para los juegos portátiles de 2009 a 2011. Se tiene que tomar en cuenta que los proyectos de noviembre y diciembre de 2011, sin en base a su relación con los 10 primeros meses del año, como se observa en 2009 y 2010. Empezando por la izquierda, para el año 2009, se calcula $ 2.7 billones en ingresos totales en EE.UU para juegos portátiles. Para 2010 y 2011, estimamos $ 2,5 mil millones y $ 3.3 millones, respectivamente [26].

Se debe reconocer que decidir jugar a los videojuegos en dispositivos móviles puede llegar a ser una experiencia limitada para algunos usuarios, frente a jugar con consolas portables, esto depende en gran manera de las expectativas esperadas de cada uno de ellos; son dos experiencias diferentes. Actualmente se puede considerar que los juegos ideales para móviles son los categorizados como "casuales": partidas rápidas y sencillas, juegos que ocupan poco espacio y gráficos hermosamente adornados pero no a tal grado de generar numerosos campos en 3D. Esto puede cambiar conforme pasa el tiempo y el desarrollo tecnológico evolucione pero sobre todo dependerá en mayor medida del uso que cada usuario le dé a los dispositivos móviles.

[25] Una "*app*" o aplicación es un tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajos en específico.

Capítulo 3

Descripción técnica

Para realizar este videojuego de estrategia por turnos, se hizo uso del lenguaje de programación *Java edición micro* (*J2ME)*.
Este videojuego no posee muchos efectos gráficos, por ser un videojuego táctico se justifica que; sólo cuenta con elementos representativos para dar una idea al usuario del estado del mapa en cada turno en el cual se desarrolla cada nivel del videojuego. El juego presentado es caracterizado por 3 niveles y 2 unidades de combate disponibles y jugables para el usuario. Existen otros 2 tipos de unidades en cada nivel del mapa pero estas están disponibles solo para la ejecución de la inteligencia artificial, es decir, juegan el papel del ejército rival, por lo que estas son las unidades denominadas *PNJ*. Cada nivel del juego está asociado con una misión que se debe de llevar a cabo por parte del usuario para poder tener éxito en el nivel y poder continuar al siguiente.
La dinámica del juego es:

1. *El usuario o PNJ elije una unidad o construcción*
2. *Se elije un punto dentro del mapa, hacia donde se desea se mueva la unidad seleccionada anteriormente o, si es el caso, alguna acción de la construcción*
3. *Se confirma la acción.*
4. *Se inhabilitara la unidad o construcción, ya que solo se permite una sola acción por unidad o construcción cada turno.*
5. *Se repite el proceso hasta que se agotan las unidades o construcciones capaces de realizar alguna acción.*

El videojuego cuenta con un set de opciones disponibles para el usuario, que permiten la configuración del juego como pueden ser: vibración, volumen, etc. Se permite también poder guardar la partida y continuar alguna partida guardada anteriormente o comenzar con un nuevo juego. El software incluye un pequeño tutorial donde se explica al usuario la dinámica del juego, los elementos que constituyen el mapa y para qué sirven así como también guardar partidas o retomar partidas guardadas con anterioridad.
En general este capítulo describe el desarrollo del software, comentando las herramientas usadas para su construcción y la descripción de algunos elementos de los que se compone el código fuente, relevantes para la operación correcta de este software.

3.1. Arte

El arte del videojuego se basa en imágenes con formato *PNG*[26] de distinta resolución según la imagen que se trate, El *PNG* es un formato gráfico cada vez más usado en lugar de *GIF*[27]. Se muestra correctamente en los navegadores, su uso está libre de derechos y permite una alta compresión así como una reproducción progresiva de imágenes con hasta 16,7 millones de colores [32].
Se decidió utilizar este formato por su escalabilidad al momento de reducir o variar el tamaño de la imagen, prácticamente la imagen al ser modificada no pierde nitidez, además de soportar fondos transparentes facilitando así su incorporación a este software. Al igual que el *GIF*, al ser *PNG* un formato sin pérdida de calidad, produce archivos excesivamente grandes para la reproducción de fotografías o cuadros comparados con los equivalentes archivos *JPG*[28]. Es posible reducir considerablemente el tamaño de imágenes con una disminución casi inapreciable de la calidad [32].
La principal desventaja del formato *PNG* es que no permite crear pequeñas animaciones como el formato *GIF*. Existe un formato complementario llamado *MNG*[29] que soporta animación pero aun no existe un estándar oficial del mismo. Por lo que no se debe de llegar a la conclusión que las animaciones presentes en el juego son producto de las mismas imágenes, más bien se hace uso de una herramienta de programación llamada *sprites*. Las imágenes de las cuales se compone cada *sprite* son mostradas cada una en un intervalo de tiempo de terminado, con esto se logra las animaciones mostradas en la ejecución del software.

[26] *PNG* (*Portable Network Graphics*) es un formato gráfico basado en un algoritmo de compresión sin pérdida para bitmaps.
[27] GIF (*Compuserve GIF*) es un formato grafico utilizado ampliamente en la *World Wide Web*, tanto para imágenes como para animaciones.
[28] JPG son las siglas de *Joint Photographic Experts Group*, el nombre del grupo que creó este formato**.** jpg es un formato de compresión de imágenes, tanto en color como en escala de grises, con alta calidad.
[29] El *MNG* (pronunciado *ming*) es un formato de fichero, libre de derechos, para imágenes animadas. Las iníciales significan ***M****ultiple-image* ***N****etwork* ***G****raphics.*

En cuanto a la creación y edición de las imágenes *PNG*, se pueden emplear sin problemas los principales editores de imágenes existentes como *CorelDRAW*[30], *Photopaint*[31] *o Paint Shop Pro*[32], *GIMP*[33] *o Adobe Photoshop*[34]. En particular se uso el editor *Adobe Photoshop* ya que se contaba con cocimientos previos.

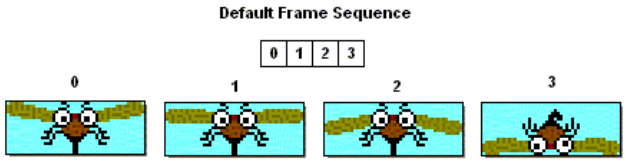

Figura 3.1: Ejemplo de una secuencia de un *Sprite* define una lista ordenada de fotogramas. Así que hay una correlación directa entre el índice de la secuencia y el índice del fotograma [33].

3.2. Lenguaje de desarrollo

El juego está programado con el lenguaje *Java*[35] específicamente con la tecnología *J2ME* especialmente diseñada y optimizada para dispositivos móviles. Esta plataforma de desarrollo se basa con conceptos de "configuración" y "perfil".
La "configuración": Describe las características mínimas en cuanto a la configuración hardware y software. La configuración que usa *J2ME* es la *CLDC* (*Connected Limited Device Configuration*) y existe, también, otro tipo de configuración llamada *CDC* (*Connected Device Configuration*).
La *CLDC* es una especificación general para un amplio abanico de dispositivos, que van desde *PDAs* a teléfonos móviles y otros dispositivos de mediana capacidad, la *CDC* es una especificación para dispositivos más avanzados como televisores, impresoras, scanner, etc. El "perfil" *MID* (conocido por *MIDP*, *MID Profile* o *Perfil Mobile Information Device)*. Un perfil define las características del dispositivo de forma más específica.

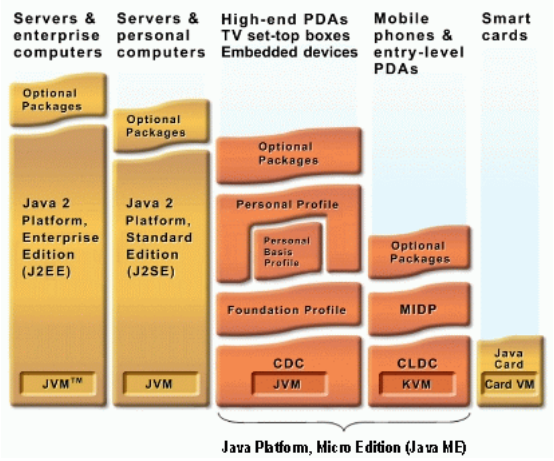

Figura 3.2: Visión general de los componentes de la tecnología *Java ME* y cómo se relaciona con las otras tecnologías *Java* [28].

[30] *CorelDRAW* es una aplicación informática de manejo vectorial, relativamente básico y sencillo de usar que hace parte del paquete de aplicaciones *CorelDRAW* Graphics Suite ofrecida por la corporación *Corel*.
[31] *Corel Photo-Paint* es un programa informático de edición de gráficos rasterizados o, también llamados Imágenes de Mapa de bits, que viene incluido en la suite de aplicaciones conocida como *CorelDRAW Graphics Suite* de la corporación *Corel*.
[32] *Paint Shop Pro (PSP)* es un editor de gráficos rasterizados y editor de gráficos vectoriales para PC que funciona sobre el sistema operativo *Microsoft Windows*.
[33] *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. Forma parte del proyecto *GNU* y está disponible bajo la Licencia pública general de *GNU*.
[34] *Adobe Photoshop* se trata esencialmente de una aplicación informática en forma de taller de pintura y fotografía que trabaja sobre un *"lienzo"* y que está destinado para la edición, retoque fotográfico y pintura a base de imágenes de mapa de bits (o *gráficos rasterizados*).
[35] *Java* es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Con respecto a la memoria, su gestión no es un problema ya que ésta es gestionada por el propio lenguaje y no por el programador [27].

3.2.1. MIDP y CLCD

El *MIDP* es un elemento clave de la Plataforma *Java 2, Mobile Edition (J2ME)*. Cuando se combina con la *CLDC*, *MIDP* proporciona un estándar para los dispositivos móvil más populares de la actualidad, teléfonos celulares y *PDAs* [29]. El *CLDC* está diseñado para llevar la plataforma *Java* a dispositivos con limitada capacidad de procesamiento, memoria y capacidad de gráficos, tales como teléfonos celulares, localizadores de gama baja, organizadores personales, etc. [31].
Actualmente existen en *J2ME* tres versiones de *MIDP* esta son la *MIDP 1.0*, *MIDP 2.0* y la *MIDP 2.1*, respecto al *CLCD* existen 2 versiones disponibles la *CLCD 1.0* y la *CLCD 1.1*.
La versión *MIDP 2.0* ofrece una nueva *API*[36] más avanzada, interfaz de usuario mejorada y funcionalidad para multimedia y videojuegos [30]. La versión *CLCD 1.1* es una versión mejorada del *CLDC 1.0* incluyendo soporte para matemáticas de punto flotante [31].
Para elaborar este software se uso la versión *MIDP 2.0* y la versión *CLCD 1.1* por las características descritas anteriormente las cuales permitieron contar con todas las herramientas necesarias para el desarrollo de este proyecto.

3.2.2. Ciclo de vida de un MIDlet

Cada ciclo de vida de un *MIDlet* es manejado por el Sistema de Gestión de Aplicaciones (*AMS* por sus siglas en ingles), se trata del software en el dispositivo que gestiona la descarga, instalación, ejecución y eliminación de aplicaciones y otros recursos en el dispositivo móvil.
Una vez que el usuario selecciona el *MIDlet* para su ejecución, el *AMS* crea una instancia de este, entonces, comienza su ciclo de vida en el estado de "PAUSA". El *AMS* mueve el estado del *MIDlet* al estado "ACTIVO"; notifica esta transición mediante la invocación del método "*startApp()"* del *MIDlet*. Asimismo, sus métodos "*pauseApp()*" y "*destroyApp()*" se encargan de notificar al *MIDlet* cuando se está en el estado "PAUSA" o "DESTRUIDO", respectivamente.
Normalmente se presta poca atención al ciclo de vida de un *MIDlet*, y como consecuencia no es posible manejar correctamente las transiciones de estado. Por ejemplo, el *MIDlet* puede ser suspendido –ya que el *AMS* hizo una pausa - cuando se recibe una llamada o el usuario sin querer puede terminar la ejecución del *MIDlet*. Así mismo los requisitos específicos de la plataforma pueden obligar a la aplicación a manejar estos casos, e incluso para volver al estado en que estaba antes de la interrupción, como si nunca hubiera salido.
La implementación de una máquina de estados finitos puede simplificar el manejo del ciclo de vida de un *MIDlet* [34].

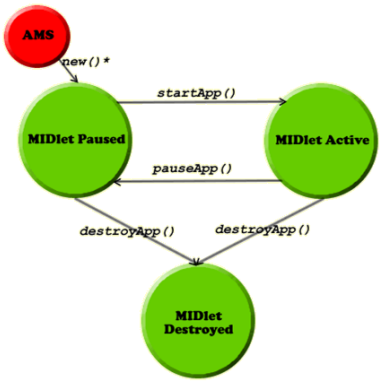

Figura 3.3: El *AMS* crea una nueva instancia de la aplicación; entonces el ciclo de vida del *MIDlet* empieza.

[36] Interfaz de programación de aplicaciones o *API* (del ingles *Application Programming Interface*) es el conjunto de funciones y procedimientos o métodos, en la programación orientada a objetos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas (también denominadas comúnmente "librerías").

La ejecución del software del presente trabajo se basa completamente en este ciclo de vida del *MIDlet*, de hecho esta implícitamente, como se menciono anteriormente, en la programación de la aplicación. Tradicionalmente un software *stan-alone*, programado con *Java SE*, contiene implícitamente un método principal de ejecución, este es el "*main(String [] args)*".

A diferencia de esto las aplicaciones programas con la tecnología de *J2ME* contienen no un método principal, mas bien, estas aplicaciones necesitan tres métodos para su correcta ejecución, siendo el principal el método "*starApp()*", y precisamente estos mismos métodos están programados en este videojuego, así se garantiza su correcta ejecución. A continuación se presenta este fragmento de código, donde es posible apreciar estos métodos descritos anteriormente.

```
/**
 * Constructor.
 */
public Amelie() {
    Amelie.theMIDlet = this;
}

/**
 * Metodo para iniciar la apliacion.
 */
public void startApp() {
    Amelie.theDisplay = Display.getDisplay(this);
    Amelie.getDisplay().setCurrent(new VideoJuego());
}

/**
 * Metodo para pausar la apliacion.
 */
public void pauseApp() {
}

/**
 * Metodo para destruir y liberar los recursos de la
 * aplicacion.
 * @param unconditional -Bandera enviada por el sistema para
 * avisar que se debe de
 * destruir la aplicacion forsosamente.
 */
public void destroyApp(boolean unconditional) {
    this.notifyDestroyed();
}
```

Figura 3.4: En este trozo de código se aprecia los tres principales métodos que deben de formar parte del cuerpo del *MIDLet*; el método "*startApp()*", "*pauseApp()*", y el método "*destroyApp(...)*", adicionalmente se muestra el constructor del *MIDLet*.

Como se puede observar en el método "*startApp()*" una vez ejecutado crea una instancia a la clase "*VideoJuego*", que aparte del *MIDLet* "*Amelie*", la clase "*VideoJuego*" es la principal clase de todo el proyecto, con respecto a su ejecución.
En la clase "*VideoJuego*" es donde se encuentra el "*GAME LOOP*[37]" del videojuego, es en este "*GAME LOOP*" donde la organización y la secuencia de ejecución del videojuego se lleva a cabo con cierto orden, convirtiendo así a el videojuego como una ejecución de secuencias debidamente organizadas facilitando así la estructura de cada una de las clases posteriores.

[37] El *"GAME LOOP*" o bucle de juego es el encargado de "dirigir" en cada momento que tarea se está realizando, leer entradas, procesar entradas, actualizar pantalla, etc. [15]

3.3. Game loop

El "*game loop"* es la estructura principal de este juego, en el se organiza cada acción del juego, esta estructura es la encargada de planificar las acciones del videojuego respecto en cada uno de los "estados" que componen al software creado.
El "*game loop*" se compone por:

1. *Inicializar clases secundarias.*
2. *Planificar las acciones, según el estado del videojuego (Ciclos).*
3. *Renderizar los elementos visuales del videojuego, según el estado del videojuego (Render).*
4. *Dibujar los elementos renderizados en la pantalla del dispositivo (Actualizacion de Pantalla).*

Se debe de diferenciar el estado de "*Renderizado*" y el estado en el cual se "*Actualiza la pantalla*", tratándose el primero sobre el proceso computacional en el cual se genera la imagen final generada a partir de varias imágenes (que componen los elementos visuales del videojuego) y el segundo el cual se encarga de plasmar la imagen generada en el proceso de renderizacion en la pantalla del dispositivo, es decir, actualizar la pantalla (Ver figura 3.5).

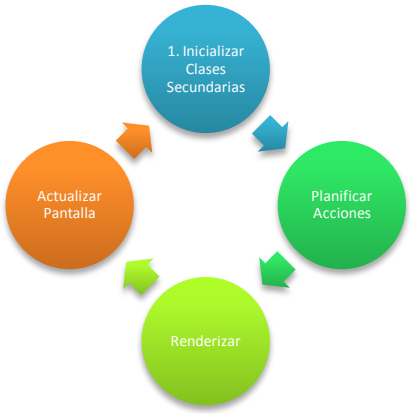

Figura 3.5: Representación del cada uno de los 4 estados de los que se compone el *game loop.*

La clase "*VideoJuego*" hereda 3 métodos de la clase "*GameCanvas*[38]", (Ver figura 3.5), con los cuales es posible atender las peticiones del usuario, estos son:

- *pointerPressed(int x, int y)*
- *pointerDragged(int x, int y)*
- *pointerReleased(int x, int y)*

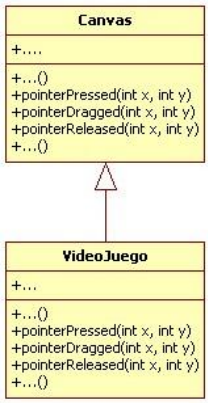

Figura 3.6: La clase "*VideoJuegos*" hereda métodos de la clase "*GameCanvas*", y los sobrescribe para poder manejar los "*gestos*" de las pulsaciones del usuario.

[38] La clase GameCanvas es una clase base para escribir aplicaciones que se necesitan manejar eventos de bajo nivel y emitir llamadas a gráficos para dibujar en la pantalla. Es probable que las aplicaciones de juegos hagan uso intensivo de esta clase.

Como es posible observar estos metodos resiven como argumento dos variables enteras, "*x, y*", los valores que seran asignadas a estas variables son las coordenadas cartecianas correspondiantes a la coordenada donde el usuario toca la pantalla del dispositivo, (Ver figura 3.7). Los valores asignados a esta variables varian según el metodo que es invocado:

- *pointerPressed(int x, int y): Recibe las coordenadas, en pixles, donde el usuario toca la pantalla.*
- *pointerDragged(int x, int y): Reside las coordenadas, en pixles, por donde el usuario desplaza el gesto.*
- *pointerReleased(int x, int y): Recibe las coordenadas, en pixles, donde el usuario deja de tocar la pantalla.*

Con estos métodos es suficiente para poder interpretar todas las acciones del usuario respecto a los *gestos* que genera sobre la pantalla del dispositivo.

Figura 3.7: Origen de coordenadas y dirección en pantallas de dispositivos móviles.

3.3.1. Estados del videojuego

Como ya se había mencionado el "*game loop"* diseñado para este proyecto se compone por:

1. *Inicializar clases secundarias.*
2. *Planificar las acciones, según el estado del videojuego (Ciclos).*
3. *Renderizar los elementos visuales del videojuego, según el estado del videojuego (Render).*
4. *Dibujar los elementos renderizados en la pantalla del dispositivo (Actualización de Pantalla).*

Se describió brevemente el funcionamiento del "*game loop*" (Ver imagen 3.5), ahora se describe los "estados" del videojuego y como estos influyen en la ejecución del "*game loop*".
La creación de los "estados" que componen al videojuego se baso en el agrupamiento de diferentes bifurcaciones de ejecución que componen al software creado, así se logro; agrupando y clasificando estas bifurcaciones en "estados", facilitar la organización del código.
Los "estados" de este del videojuego se componen de:

1. *PRESENTACION.*
2. *MENU_PRINCIPAL.*
3. *MENU_START_GAME.*
4. *SHOW_CREDITS.*
5. *TUTORIAL.*

El "*estado*" en el que se encuentre el videojuego determina la ejecución, en particular de cada elemento, que forman al "*game loop*", entonces, si el "*estado = PRESENTACION*" y el "*game loop*" se encuentra en la secuencia de "*Inicializar clases secundarias.*", entonces, el modulo encargado de esta secuencia perteneciente al "*game loop*" ejecutara la inicialización de los distintos módulos que comprenden a el estado "*PRESENTACION*".
Esto es exactamente igual para las otras secuencias del "*game loop*" (Ver figura 3.8).

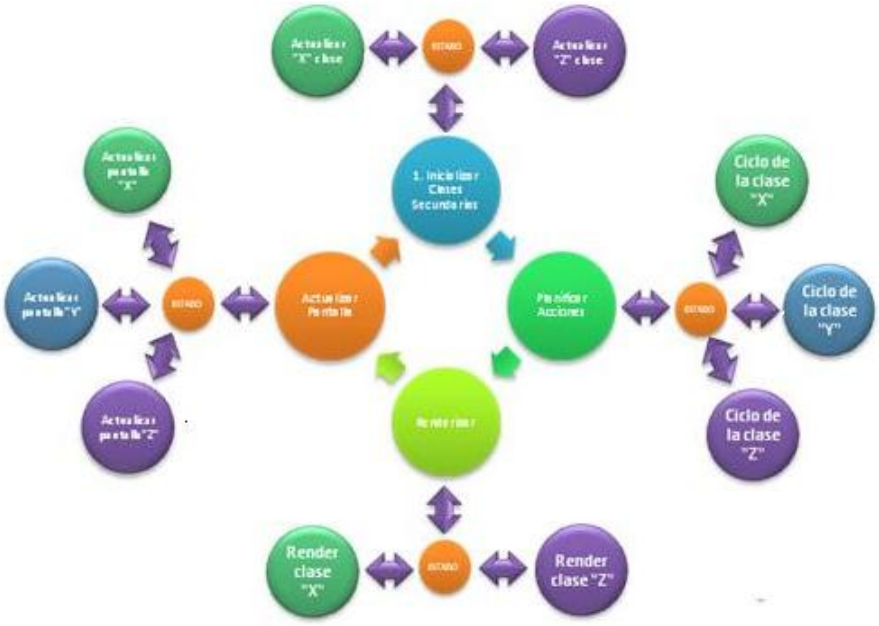

Figura 3.8: Representación del cada uno de los 4 estados de los que se compone el *game loop* mas una representación de cómo influye el estado del videojuego para la ejecución de cada secuencia del *game loop.*

El "*estado*" en el que se encuentre el videojuego no solo determina la ejecución, en particular de cada elemento, que forman al "*game loop*", sino también, afecta a la ejecución de los métodos que hereda la clase "*VideoJuego*" de la clase "Game*Canvas*", es decir, los métodos:

- *pointerPressed(int x, int y)*
- *pointerDragged(int x, int y)*
- *pointerReleased(int x, int y)*

Ya que estos métodos, de igual manera que las secuencias del "*game loop*", deben de ser capaces de saber en que "estado" se encuentra el videojuego para poder interpretar adecuadamente los *gestos* generados por el usuario (Ver figura 3.9).
Estos métodos, heredados de la clase *GameCanvas* a la clase *VideoJuego*, son sobreescritos para poder implementarlos como es debido para esta aplicación.
Interiormente en cada uno de estos métodos se debe de redirigir las coordenadas resididas como parámetros al conjunto de clases perfectamente definidas por medio del "estado" del videojuego.
La ejecución de estos métodos conlleva a afectar directamente la ejecución de la secuencia "*Ciclos*" del *game loop* ya qué las peticiones del usuario capturadas por estos métodos deben de ser atendidas y estas acciones afectan la ejecución del videojuego.

3.3.1.1. Estado: Presentación

El estado del videojuego denominado "*presentación*" se encarga de cargar en memoria el sonido y las imágenes del inicio del juego, las cuales corresponden a la imagen de bienvenida, y la imagen de carga. (Ver figura 3.10).

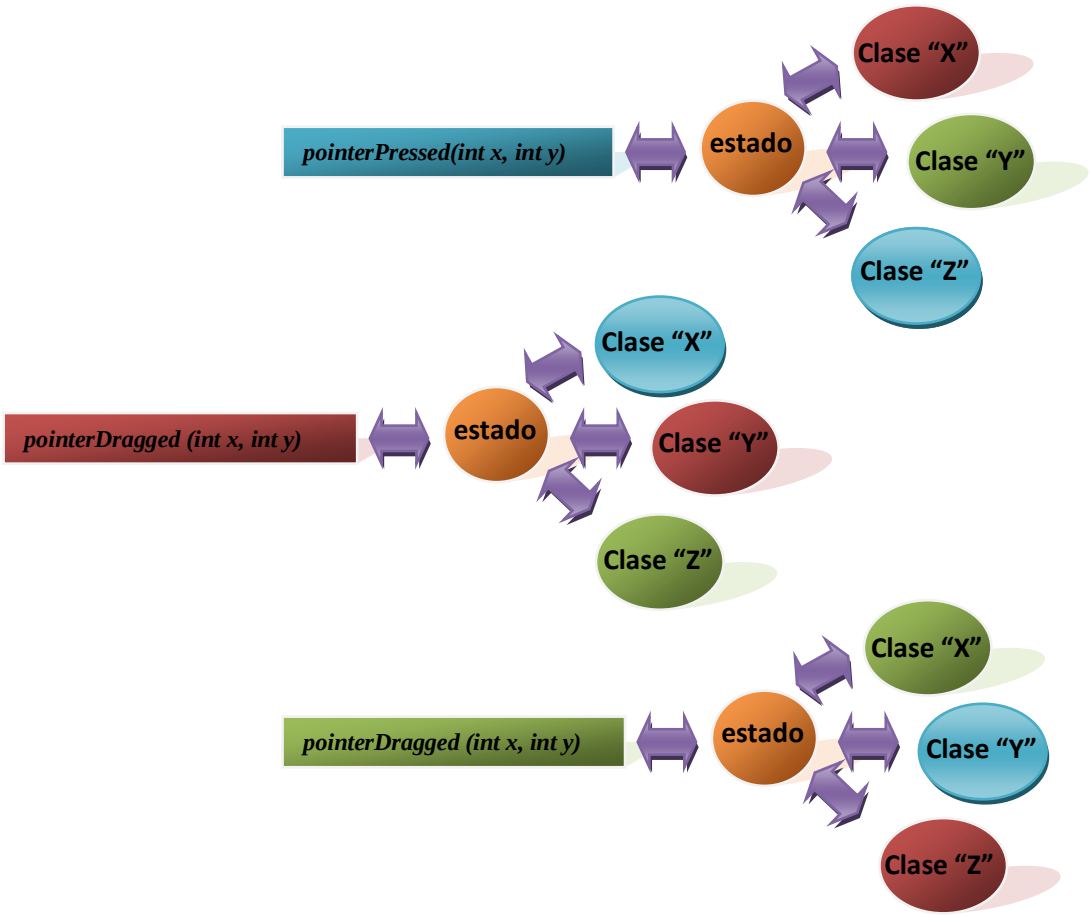

Figura 3.9: El estado del videojuego afecta la ejecución de los métodos encargados de gestionar los *gestos* proporcionados por el usuario.

Pantalla de "Bienvenida"

Pantalla de "Carga"

Figura 3.10: Pantallas de inicio del software, pertenecientes al estado "*Presentación*" del videojuego

3.3.1.2. Estado: Menú Principal

El estado del videojuego denominado "*Menú Principal*" se encarga de cargar en memoria el sonido y las imágenes del menú principal y las del menú de opciones. (Ver figura 3.11). Además es, en este estado del software, donde el usuario puede empezar a interactuar con el videojuego.
Pantalla del Menú Principal: Esta pantalla ofrece al usuario 4 opciones, estas son:

1. *"Comenzar el juego"*
2. *"Opciones del juego"*
3. *"Tutorial del juego"* y
4. *"Salir del juego".*

Pantalla Menú de Opciones: Esta pantalla ofrece al usuario 5 opciones, estas son:

1. *"Aumentar/Disminuir Volumen"*
2. *"Habilitar/Deshabilitar Vibración*
3. *"Borrar partida"* y
4. *"Regresar a la pantalla de menú principal"* y
5. *"Silenciar completamente el sonido".*

Pantalla del "Menú Principal" Pantalla del "Menú de Opciones"

Figura 3.11: Pantallas de inicio del software, pertenecientes al estado "*Menú Principal*" del videojuego. Los círculos morados indican las opciones correspondientes para cada pantalla descritas anteriormente.

3.3.1.3. Estado: MENU_START_GAME

El estado del videojuego denominado "*Menú start game*" se encarga de cargar en memoria el sonido y las imágenes propias del juego y del menú secundario. (Ver figura 3.12). En este estado del videojuego el usuario decide si empieza una partida nueva del juego o carga una partida guardada anteriormente, posteriormente inicia propiamente el juego según la opción elegida anteriormente por el usuario.
Pantalla del Menú Secundario: Esta pantalla ofrece al usuario 2 opciones, estas son:

1. *"Comenzar una partida nueva del juego"*
2. *"Cargar una partida del juego salvada anteriormente"*

Pantalla del juego: Esta pantalla se compone varios elementos, estas son:

1. *"Elementos HUD (Salud de las unidades, Objetivos, etc.)"*
2. *"Elementos GUI (el pequeño menú de opciones y menú de unidades a la izquierda)"*
3. *"El tablero del Juego"* y
4. *"Sprites (Todos los personajes: Tanques y Soldados)"*

Pantalla del Menú Secundario

Pantalla del juego

Figura 3.12: Pantallas pertenecientes al estado "*Menú start game*" del videojuego. Los círculos morados indican los elementos correspondientes para cada pantalla descritas anteriormente.

3.3.1.4. Estado: SHOW_CREDITS

El estado del videojuego denominado "*Show_Credits*" se encarga de cargar en memoria el texto de los créditos del videojuego. (Ver figura 3.13). En este estado del videojuego el usuario solo puede esperar a que pasen todos los créditos del videojuego, para luego regresar al menú principal, es decir, el software regresa al estado "*Menú Principal*".
Se muestran de donde se obtuvieron distintos elementos del juego como por ejemplo la música o algunas imágenes, además, se muestran las licencias bajo las cuales los elementos del videojuego estas restringidos.

Pantalla de créditos

Figura 3.13: Pantalla pertenecientes al estado "*Show Credits*" del videojuego.

3.3.1.5. Estado: TUTORIAL

El estado del videojuego denominado "*Tutorial*" se encarga de cargar en memoria las imágenes para mostrar al usuario una pantalla donde se muestra un pequeño tutorial que explica los distintos elementos del videojuego. (Ver figura 3.14).
En este estado del videojuego se ofrece un pequeño menú en donde el usuario decide si quiere seguir "adelante" o "retroceder" en el tutorial, además, de tener la opción de "regresar" para regresar al menú principal, es decir, el software regresa al estado "*Menú Principal*".
Pantalla del Tutorial: Esta pantalla ofrece al usuario 3 opciones y un elemento que forma el tutorial:

1. *"Siguiente"*
2. *"Salir"*
3. *"Anterior"* y
4. *"Elemento formado por una serie de imágenes para mostrar el tutorial".*

Pantalla del tutorial

Figura 3.14: Pantalla pertenecientes al estado "*Tutorial*" del videojuego. Los círculos morados indican los elementos correspondientes a la pantalla descrita anteriormente.

3.4. Algoritmos de Inteligencia

Pac-Man fue el primer juego que mucha gente recuerda haber jugado con una Inteligencia Artificial simple. Hasta ese momento había muchos clones de *Pong*[39] con un rival controlado por la computadora (que básicamente sigue el balón hacia arriba y hacia abajo) y un sinnúmero de juegos basados en el estilo de un *Space Invaders*[40]. Sin embargo, *Pac-Man* tenia personajes definidos, enemigos que parecía conspirar en su contra, se movían alrededor del nivel de la misma manera que el lo hacía, haciéndole la vida difícil.
Pac-Man se basó en una técnica de Inteligencia Artificial muy simple: una máquina de estados. Cada uno de los cuatro monstruos (más tarde llamados fantasmas) era ya sea que persiguen a *Pac-Man* o huyen de él. Para cada estado, en cada cruce, se tomaba una ruta semi-aleatoria. En el modo de cazar o persecución de *Pac-Man*, cada uno de los fantasmas tenía la oportunidad de perseguir al jugador o elegir una dirección al azar. En el modo de huida o bien huyeron o eligieron una dirección al azar. "Todo muy sencillo y muy 1979".
El modelo de la Inteligencia Artificial no cambio mucho hasta mediados de los 90s. Los personajes controlados por la computadora antes de 1990 eran tan sofisticados como un fantasma *Pac-Man.*
La Inteligencia Artificial en la mayoría de los juegos modernos está dirigida a tres necesidades básicas:

1. *La capacidad de mover a los personajes*
2. *La capacidad de tomar decisiones acerca de dónde se mueven,* y
3. *La capacidad de pensar estratégicamente y tácticamente*

A pesar de que hemos pasado de utilizar el estado de la Inteligencia Artificial basada en maquinas de estados (que todavía se utilizan en la mayoría de los juegos, porque así lo requieran) a una amplia gama de técnicas, todos cumplen los mismos tres requisitos básicos [35].

3.4.1. Creación de rutas

Cada unidad de combate disponible para el juego (tanque o soldado) necesita un punto origen y un punto destino dentro del tablero del mapa para establecer una ruta de desplazamiento del punto origen al punto destino sobre el mapa correspondiente a cada nivel del juego.
Para poder especificar una ruta de desplazamiento de una unidad hacia un punto destino se hizo uso del algoritmo *A**, (*A-start*), con el cual se puede encontrar una ruta con una eficiencia aceptable, en cuanto al camino más corto.
El mapa o tablero de cada nivel del juego está compuesto de una cierta cantidad de celdas, (Ver figura 3.15), cada una de las unidades solo pueden desplazarse cierta cantidad de celdas en el mapa, iluminadas al momento de elegir alguna

[39] *Pong* (o *Tele-Pong*) fue un videojuego de la primera generación de videoconsolas publicado por Atari, creado por Nolan Bushnell y lanzado el 29 de noviembre de 1972. *Pong* está basado en el deporte de tenis de mesa (o *ping pong*)..
[40] *Space Invaders* es un videojuego de arcade diseñado por Toshihiro Nishikado y lanzado al mercado en 1978. En un principio fue fabricado y vendido por Taito Co en Japón, para posteriormente ser licenciado para producción y distribución en Estados Unidos por Midway, división de Bally Technologies. *Space Invaders* es uno de los primeros juegos "enemigos arriba usuario abajo". Es uno de los videojuegos más importantes de la historia. Su objetivo es eliminar oleadas de alienígenas con un cañón láser y obtener la mayor cantidad de puntos posible. Para el diseño del juego, Nishikado se inspiró en *Breakout*, La guerra de los mundos y Start Wars.
Aunque es un juego simple para los estándares actuales, fue uno de los precursores de los videojuegos modernos y ayudó a expandir la industria del sector, desde una mera novedad a una industria global. Fue exitoso y popular desde su lanzamiento, tanto que llegó a una breve escasez de monedas de cien yenes en Japón. Al año 2007, Taito ha ganado US$500 millones en utilidades.

unidad para moverla, así se evitara la exploración de una gran cantidad de celdas que componen al tablero del mapa así como una sobre carga de procesamiento para la plataforma en el cual se ejecuta el videojuego.
La creación de rutas se efectúa tanto para las unidades del *PNJ* como para las unidades del jugador, ya que el jugador especifica un punto destino en el mapa para cierta unidad escogida, al igual para las unidades del *PNJ*, la inteligencia artificial establece un punto destino para cierta unidad. La creación de la ruta para esta unidad dependerá de la ejecución del algoritmo *A**, conociendo el punto de origen y el punto destino en la matriz del mapa se puede establecer una ruta de desplazamiento para cada una de las unidades, donde cada celda del mapa es un elemento de la matriz, El algoritmo también permite verificar y así evitar las colisiones que pudieran surgir entre unidades y entre la unidad y el terreno, por ejemplo: los tanques evitan siempre las montañas y el mar para planificar su ruta. Todo esto permite establecer una ruta aceptable para cada una de las unidades de combate que componen a este juego.

Figura 3.15: Se observan las celdas iluminadas sobre las cuales la unidad se puede desplazar, los puntos rojos son la ruta establecida por el algoritmo *A*;* desde la posición actual de la unidad (origen) hasta la posición destino.

El algoritmo *A** hace uso de una división en el mapa por medio de cuadrados, rectángulos, pentágonos, etc., para este caso, dado que el mapa se trata de una matriz, la división del mapa se hizo por medio de cuadrados, a cada celda del mapa se le asigna un respectivo peso *F*, que depende de la suma de la variable *G* llamada "coste de movimiento", de un origen hacia un cuadrado adyacente, y una variable *h* llamada "coste de movimiento estimado", esta *h* es comúnmente llamada heurística. El método que se uso para calcular *h* se llama el "método *Manhattan*", donde se calcula el número total de cuadros recorridos horizontalmente y verticalmente para alcanzar el cuadrado destino desde el cuadro origen, sin hacer uso de movimientos diagonales. Luego se multiplica el total por 10. Se llama método *Manhattan* porque es como calcular el número de manzanas que hay desde un lugar a otro, donde no puedes acortar atravesando en diagonal una manzana. Cabe señalar que cuando se calcula *h*, se ignora cualquier obstáculo que intervenga. Es una estimación de la distancia que queda, no de la distancia real, es por eso que se llama heurística. Entonces el peso *F* se establece de la siguiente manera:
$F = G + H$

Donde:

- *G* = el coste de movimiento para ir desde el punto de origen a un cierto punto adyacente, siguiendo el camino generado para llegar allí.
- *h* = el coste de movimiento estimado para ir desde el punto de origen hasta un cierto punto destino [18].

3.4.2. Comportamiento del PNJ

Es importante mencionar como se establece el comportamiento para el *PNJ*, en los videojuegos la inteligencia artificial puede implementar diversos procedimientos que en conjunto crean una simulación de actos apropiados y naturales para el espectador; para soluciones eficaces se emplean técnicas como: maquinas de estados finitos, búsqueda de caminos, Heurística, Algoritmos genéticos, etc. [17]. El videojuego presentado, además de ocupar el algoritmo *A** para la búsqueda de caminos, ocupa una "maquina de estados finitos" para poder definir el comportamiento de las unidades del *PNJ*.

Una maquina de estados finitos provee lo necesario para simular una acción inteligente por parte de las unidades designadas al *PNJ*. A continuación de listan las principales acciones que la inteligencia artificial debe de ser capaz de realizar:

1. Atacar alguna unidad del usuario o,
2. Conquistar alguna construcción y
3. Ser capaz de crear nuevas unidades.

La máquina de estados finitos se compone de 12 estados en total, los cuales se muestran a continuación:

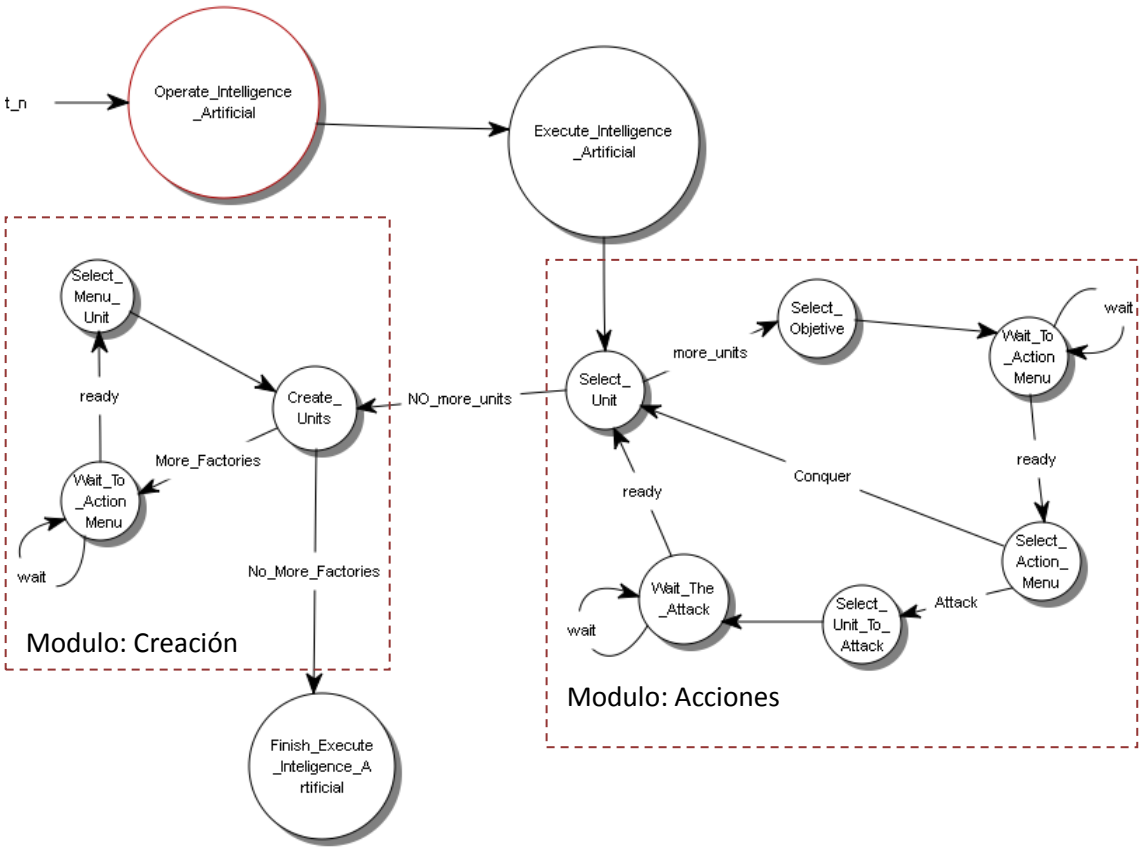

Figura 3.16: Esta máquina de estados es ejecutada cada vez que el usuario termina su turno de jugar. Por medio de esta máquina se define el comportamiento de la inteligencia artificial; tanto para mover los personajes *PNJ* como también para la creación de nuevas unidades.

El diseño de la inteligencia artificial se baso en las tareas repetitivas que un usuario hacia al mover cada uno de sus unidades, decidiendo así que lo más apropiado para modelar la inteligencia artificial de este videojuego era una maquina de estados ya que es repetitiva, bien definida y provee una buena simulación de automatización.
A continuación se describen cada uno de los 12 estados que componen a esta máquina de estados finitos; la cual define a la inteligencia artificial del videojuego:

1. *Operate_Inteligence_Artificial:* En este estado, siendo el primero en ejecutarse, es el encargado de localizar cada una de las unidades *PNJ* que se encuentran disponibles en cada uno de los niveles del juego, además, también localiza todas las "fabricas" que las unidades *PNJ* han conquistado, esto con el fin de poder producir nuevas unidades.
2. *Execute_Inteligence_Artificial:* Este estado es el encargado de, ya habiendo localizado a todas las unidades y fabricas pertenecientes al *PNJ*, planificar las acciones para cada una de las unidades *PNJ*, estas acciones son ATACAR a alguna unidad enemiga y CONQUISTAR alguna construcción ya sea enemiga o libre. Cada acción de cada personaje se ejecuta secuencialmente, es decir, si existe una cantidad N de unidades *PNJ* entonces se ejecutaran N acciones una tras de otra.

Los siguientes 5 estados pertenecen a un modulo llamado “Acciones”, los estados pertenecientes a este modulo son los encargados de ejecutar la “acción” de cada una de las unidades denominadas *PNJ*, como se menciono anteriormente estas acciones son ATACAR o CONQUISTAR.

3. *Select_Unit:* En este estado se verifica si existe alguna unidad clasificada como *PNJ* que no allá efectuado alguna acción, en caso de hallar por lo menos una unidad aun con la posibilidad de efectuar alguna acción este estado se encarga de “seleccionar” a dicha unidad, se referirá a esta unidad como X, y pasar al siguiente estado del modulo “Acciones”, al contrario de no hallar alguna unidad aun habilitada para efectuar alguna acción, entonces el flujo de la máquina de estados es dirigida al primer estado del modulo “Creación”.
4. *Select_Objetive:* Este estado es el encargado de, ya habiendo seleccionado alguna unidad X habilitada para realizar alguna acción, se selecciona el objetivo de dicha unidad, ya un objetivo para atacarlo o para conquistarlo.
5. *Wait_To_Action_Menu:* Por motivos de procesamiento propios del dispositivo se debe de generar un retraso en la ejecución del programa para dar tiempo a la unidad X a llegar al objetivo, es este estado el encargado de generar este retraso para después pasar al siguiente estado del modulo “Acciones”.
6. *Select_Action_Menu:* Habiendo alcanzado la unidad X su objetivo se debe de confirmar la acción a realizar en caso de confirmar la acción de CONQUISTAR se regresara al estado “*Select_Unit”* para seleccionar la siguiente unidad, en caso de ser ATACAR el estado siguiente será *Select_Unit_To_Attack.*
7. *Select_Unit_To_Attack:* dirigirse la unidad X a atacar algún objetivo se debe de confirmar a que unidad va a tacar ya que la unidad X puede encontrarse ante el caso de tener a mas de una unidad a sus costados por lo que tendrá la posibilidad de elegir a que unidad atacar. Este estado es el encargado de seleccionar a dicha unidad que sufrirá el ataque de la unidad X.
8. *Wait_The_Attack:* Al atacar a alguna unidad se efectúa una animación de dicho ataque. Este estado es el encargado de generar un retraso para poder dar oportunidad a la ejecución de esta animación, terminada la animación se efectúa a pasar al *Select_Unit* para seleccionar alguna otra unidad a efectuar alguna acción o a pasar al modulo denominado “CREACION”.

La creación de nuevas unidades se define en 3 estados que pertenecen al modulo “CREACION” de la máquina de estados. El modulo denominado “CREACIÓN” es el encargado crear unidades nuevas en función de la cantidad de fabricas que hayan sido conquistadas por unidades denominadas PNJ.

9. *Create_Units:* En este estado se verifica si existe alguna fabrica conquistada por las unidades *PNJ* que no allá creado ya alguna unidad, en caso de hallar por lo menos una fabrica con la posibilidad de crear alguna unidad nueva se “selecciona” a dicha fabrica, y pasar al siguiente estado del modulo “CREACION”, al contrario de no hallar alguna fabrica habilitada para crear alguna unidad el flujo de la máquina de estados es dirigida al estado encargado de finalizar la ejecución de la inteligencia artificial..
10. *Wait_To_Action_Menu:* Por el procedimiento que se lleva a cabo para crear alguna unidad nueva, se crea un retraso en la ejecución y es este estado el encargado de generar este retraso. El procedimiento que se lleva a cabo para la creación de una nueva unidad consiste en verificar si se cuenta con la cantidad de EFECTIVO necesario para poder comprar alguna unidad nueva, en caso de contar con EFECTIVO suficiente para tener la posibilidad de poder adquirir más de una unidad nueva; se decide aleatoriamente.
11. *Select_Menu_Unit:* Terminado el proceso para la elección de una nueva unidad, esta decisión se confirma y es este estado el encargado de seleccionar la unidad a ser creada. Posteriormente se pasa al estado *Create_Units* en el cual se evalúa si existe la posibilidad de crear otra unidad o se pasa a finalizar la ejecución de la inteligencia artificial.

Al finalizar la ejecución de los módulos “ACCION” y “CREACIÓN” se debe de dar por terminado la ejecución de la inteligencia artificial de manera correcta para una próxima ejecución de esta máquina de estados.

12. *Finish_Execute_Inteligence_Artificial:* Este es el encargado de inicializar diversas variables que fueron modificadas en cada uno de los estados anteriores, con el propósito de un correcto funcionamiento en la próxima ejecución de esta máquina de estados.

Estos 12 estados explicados anteriormente son los que componen a la máquina de estados finitos que definen el comportamiento de las unidades del *PNJ*, y están reflejados en la programación de la inteligencia artificial del videojuego.

3.5. Almacenamiento y recuperación de una partida

Este videojuego se basa en la interpretación de una serie de matrices las cuales representan de forma abstracta los elementos que componen al escenario (unidades, construcciones, terreno, salud de las unidades, etc.).
Esta aplicación provee la opción al usuario de guardar la partida y también es posible eliminar una partida guardada mediante la opción de "Borrar partida".
Tanto la opción de guardar como la de borrar una partida son operaciones basadas en una colección de registros que provee directamente *J2ME;* almacenando en estos registros todas las matrices que representan al videojuego o eliminando estos registros, respectivamente.

Figura 3.17: Se muestran las 2 opciones que el usuario puede tener acceso: "Guardar una partida" y "Eliminar una partida guardada".

En las aplicaciones móviles basadas en *J2ME* se pueden almacenar datos, estos datos son almacenados en lo que se llama `Records`, un *Record* o registro es un dato individual de cualquier tipo de dato (*string, array*, imagen, etc.).
Para utilizar esta capacidad entra el concepto de *RMS* que no es más que el objeto que nos provee el almacenamiento y asigna un identificador único, algo importante es que un *record* o registro no es lo mismo que los *RMS.*
Con los *records* solamente es posible guardar un dato de cualquier tipo, por lo que se debe crear un *record* por cada dato, para solucionar esto y dar un poco más de capacidad se utilizan los "*Recod Stores*", que en conceptos simples no es más que una colección de records.
Para utilizar "*Records Stores"* o *records* se considero las capacidades del dispositivo para el cual se desarrollo este video juego, ya que se debe de tener cuidado al crear un *Record Store* que almacene muchos datos.
Los *RMS* son considerados como base datos locales en una aplicación móvil, pero esto no tiene que ver nada con una base datos como SQLPostgres, MySQL, etc. Otro aspecto importante es que el *record store* generado para almacenar la partida de este videojuego no es posible de observar al explorar el sistema de archivos del dispositivo móvil, el *RMS* es almacenado en la misma aplicación y solo la aplicación que genero el *record store* puede acceder a él.
A nivel de API un record store es representado por una instancia de la clase `javax.microedition.rms.RecordStore`, y todas las clases para el manejo de RMS están definidas en `javax.microedition.rms`.

3.5.1. Almacenamiento

A continuación se describe el proceso con el cual se guarda en un *Record Store* los datos del juego (Ver figura 3.18).

1. Se obtienen los datos a ser guardados: *nivel, el ancho y alto de las matrices, las ganancias del usuario y del PNJ, la matriz de construcciones, la matriz referente a la salud de las construcciones tanto del usuario como las del PNJ, la matriz que describe el terreno, la matriz que describe la posición y el tipo de las unidades, la matriz referente a la salud de las unidades tanto las del usuario como las del PNJ y por último la matriz que describe las unidades deshabilitadas y las habilitadas a realizar alguna acción.*
2. Cada uno de los datos recuperados son asignados a un objeto de la clase de flujo de datos definido por el lenguaje de programación *JAVA* "`DataOutputStream`", por medio del método:
   "`DataOutputStream.writeInt(dato)`".
3. Se extrae del objeto de la clase "`DataOutputStream`" un arreglo de *bytes:*
   "`DataOutputStream.toByteArray()`".

4. El arreglo "`dateToSave`" es agregado al *Recod Store* para ser almacenado pasándole como parámetros los datos a ser almacenados, el índice de dónde empezar a leer los datos del arreglo y el índice de donde terminaran de leer el arreglo: "`RecordStore.addRecord(dataToSave, 0, dataToSave.length)`".

3.5.2. Recuperación

A continuación se describe el proceso con el cual se recuperan los datos del *Record Store* el cual almacena la información de alguna partida guardada por el usuario anteriormente (Ver figura 3.18).

1. Se recuperan todos los datos almacenados en el *Record Store* y son almacenados en un arreglo de Byte, posteriormente ese arreglo de bytes es manejado a través de un objeto de la clase "`DataInputStream`".
2. Se obtienen cada uno de los datos necesarios para la reconstrucción del nivel: *nivel, el ancho y alto de las matrices, las ganancias del usuario y del PNJ, la matriz de construcciones, la matriz referente a la salud de las construcciones tanto del usuario como las del PNJ, la matriz que describe el terreno, la matriz que describe la posición y el tipo de las unidades, la matriz referente a la salud de las unidades tanto las del usuario como las del PNJ y por último la matriz que describe las unidades deshabilitadas y las habilitadas a realizar alguna acción*, a través del método "`DataInputStream.readInt()`".
3. Cada uno de los datos recuperados son asignados a sus respectivas clases dentro del código fuente del videojuego y con ellos es inicializado el juego y así este se iniciara justamente como el usuario lo había guardado.

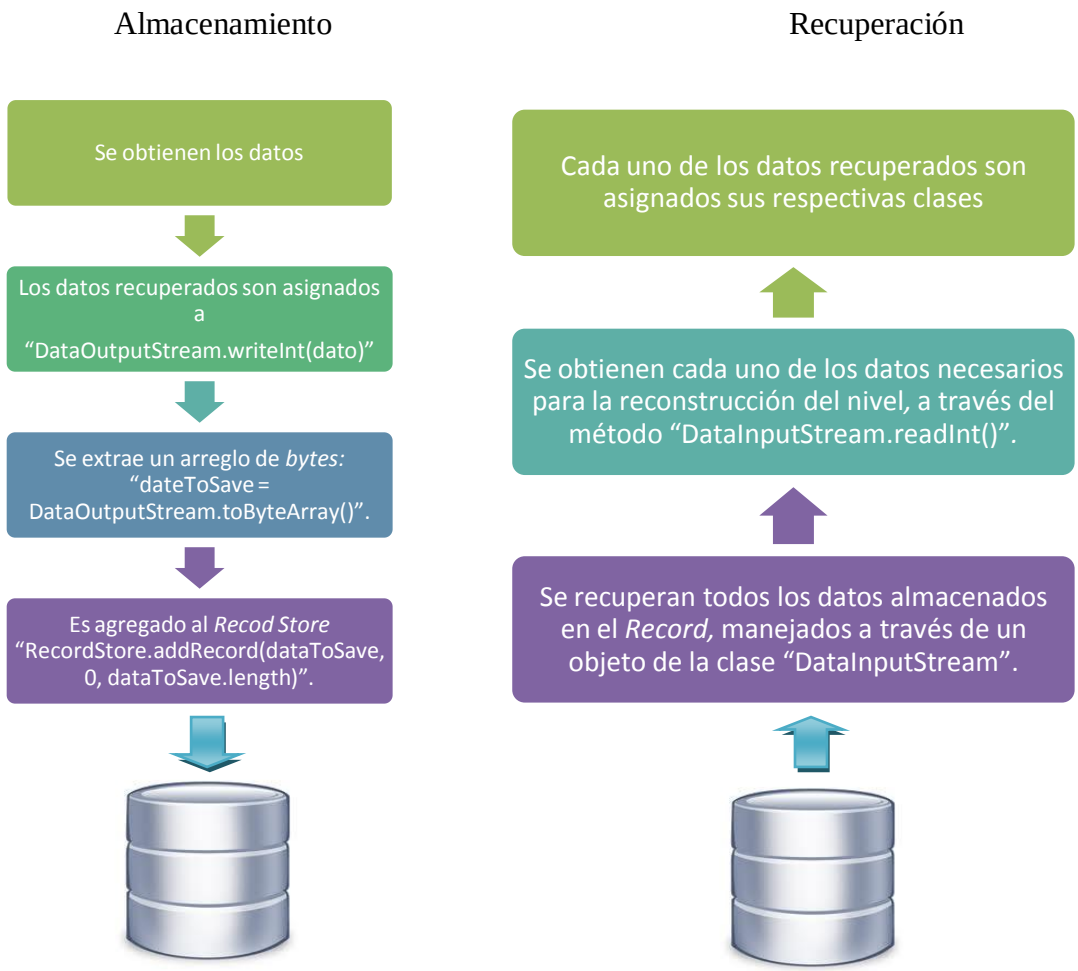

Figura 3.18: Se muestra el proceso con el cual se almacena o recupera algún estado del videojuego, salvado por el usuario.

Conclusiones

El continuo avance de la tecnología y las necesidades básicas de comunicación, han permitido el crecimiento de los dispositivos móviles, como un elemento de apoyo en la cotidianidad de las personas, en la agilización y optimización de los procesos de las organizaciones e instituciones educativas.

Es por eso que el término computación móvil, se ha convertido en una situación familiar en nuestra sociedad.

La funcionalidad de la computación móvil, se ha convertido en un fuerte apoyo a las características que brinda la computación tradicional, y no como se piensa en algunos sectores que viene a reemplazarla. Se habla de apoyo porque permite realizar procesos que la computación tradicional no realiza de forma óptima y más aun, si reconocemos que en algunos ambientes el factor de la movilidad juega un papel muy importante.

Los teléfonos celulares, los cuales han tenido un crecimiento vertiginoso y evolutivo, llegan cada día a formar parte de la vida de las personas, estos poseen una gran acogida debido a las diversas aplicaciones que traen preinstalados, con la posibilidad de incluir muchos programas o aplicaciones cuando el usuario desee.

Los videojuegos son un ejemplo de una de tantos tipos de aplicaciones que están a disposición del usuario para poderlo instalar en un dispositivo móvil, estos son capaces de brindar un entretenimiento y al mismo tiempo generar ganancias para algunos sectores comerciales brindando funcionalidades extras, difundiendo publicidad, entre tantas mas.

Este proyecto fue enfocado en la elaboración de un videojuego de estrategia. El sistema se probo y demostró una eficiencia aceptable tanto en rendimiento como en usabilidad, aprovechando el computo móvil como plataforma haciendo uso del *touchscreen* e implementando todas las funcionalidades con las que cuenta un videojuego comercial como son: bandas sonoras, efecto de vibración, respaldo de avances y manejo de animaciones apropiadas para el tipo de videojuego.

Específicamente, se desarrollo un software de entretenimiento implementando algoritmos tanto para la animación de imágenes como también para el control de bandas sonoras que ambientan el videojuego. Además, de la elaboración de un sistema que emula aun jugador siendo este la inteligencia artificial del videojuego.

El sistema cuenta también con diversas pantallas y menús en función de la pantalla en la que se encuentre el flujo del programa. Respecto a los niveles del videojuego se elaboraron 3 distintos mapas y de ellos depende la complejidad del videojuego, gracias a que la inteligencia está en función de una maquina de estados finitos es capaz de adaptarse a cualquiera de estos 3 distintos mapas ya que su comportamiento es repetitivo, sin embargo, las acciones que la inteligencia artificial realizan no se perciben como tales, por el numero de movimientos que la inteligencia puede llegar a ejecutar además del dinamismo que cada uno de los tableros sufre a consecuencia de las acciones del usuario la inteligencia artificial realiza distintas acciones logrando así una percepción más inteligente y menos repetitiva respecto a la ejecución de la IA.

Para la elaboración de este software se utilizo un lenguaje de Programación Orientado a Objetos, *Java Micro Edition* (*J2ME*), desde el sistema operativo Windows XP. Este software fue diseñado para dispositivos móviles con resolución de 640 por 360 pixeles, *display touch sceen* y maquina virtual de java *MIDP 2.0*. Se probó en los dispositivos Nokia 5230, 5530 y 5800.

Bibliografía

[1] Abarca Alfonzo, Kitzia. "*Implementación de una interfaz de usuario de la plataforma de educación a distancia Moodle utilizando asistentes personales (PDAs).* ", proyecto terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco, México, 2007

[2] Sánchez Sánchez, Guillermo Augusto. "*Máquina de aprendizaje en un juego de ajedrez."* , proyecto terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco, México, 2008

[3] *Solo Programadores.* Martinez, Alberto Guitierrez. 124, s.l. : REVISTAS PROFESIONALES S.L, Mayo 2005.

[4] Universia. *Impartirá IPN primer programa de posgrado en sistemas computacionales móviles*. [En línea] 04 de Noviembre de 2009. [Citado el: 14 de Junio de 2010.] http://noticias.universia.net.mx/tiempo-libre/noticia/2009/11/04/119329/impartira-ipn-primer-programa-posgrado-sistemas-computacionales-moviles.html.

[5] Jiménez, Arturo. *Móviles, nicho para videojuegos mexicanos*. [En línea] MundoEjecutivo.com.mx, Jueves 03 de Junio de 2010. [Citado el: 14 de Junio de 2010.] http://www.mundoejecutivo.com.mx/negocios-finanzas/industrias/moviles-nicho-para-videojuegos-mexicanos.html.

[6] Balerdi, Félix Etxeberria. *Videojuegos y educación.* [En línea] REVISTA ELECTRÓNICA Universidad de Salamanca. [Citado el: 15 de Junio de 2010.] http://campus.usal.es/~teoriaeducacion/rev_numero_02/n2_art_etxeberria.htm.

[7] Martí, Juan Alberto Estallo. *Psicopatología y Videojuegos*. [En línea] Departaento de Psicopatología - Universidad de Barcelona, Junio de 1997. [Citado el: 15 de Junio de 2010.] http://www.ub.es/personal/videoju.htm.

[8] ABC. *Organización superdotados aconseja juegos estrategia para estimular intelecto*. [En línea] ABC, 06 de Marzo de 2010. [Citado el: 17 de Junio de 2010.] http://www.abc.es/agencias/noticia.asp?noticia=299335.

[9] Hamer, Carol. *Creating Mobile Games Using Java ME Platform to Put the Fun into Your Mobile Device and Cell Phone.* New York : Technology in Action Press, 2007.

[10] Harvey M. Deitel, Paul J. Deitel. *Como Programar en Java.* s.l. : Pearson, Prentice Hall, 2004.

[11] Prieto Martín, Manuel Jesús. *Desarrollo de juegos con J2ME : Java 2 micro edition.* s.l. : Alfaomega, 2005.

[12] *¿Qué es J2ME?* . [En línea] [Citado el: 13 de Junio de 2010.] http://www.java.com/es/download/faq/whatis_j2me.xml.

[13] *Moodle*. [En línea] [Citado el: 15 de Junio de 2010.] http://moodle.org/.

[14] *Prensa y analistas* - Bluetooth. [En línea] [Citado el: 15 de Junio de 2010.] http://www.bluetooth.com/Spanish/Technology/Pages/default.aspx.

[15] Serrano, Alberto García. mailxmail.com. [En línea] 7 de Septiembre de 2004. [Citado el: 18 de Junio de 2010.] http://www.mailxmail.com/curso-programacion-juegos-moviles-j2me/game-loop-1.

[16] *Netbeans*. [En línea] ORACLE. [Citado el: 18 de Junio de 2010.] http://netbeans.org/index_es.html.

[17] *Scribd* [En línea] 30 de Noviembre de 2008. [Citado el: 01 de Julio de 2010]. http://www.scribd.com/doc/8550246/Inteligencia-artificial-programacion-videojuegos

[18] Patrick Lester. *A* Pathfinding para Principiantes* [En línea] 2 de Marzo de 2003 [Citado el: 30 de Noviembre de 2008]. http://www.policyalmanac.org/games/articulo1.htm.

[20] Wolf, Mark J.P. *The Medium of the Video Game* (1ª edición). University of Texas Press.

[21] Luna Navarrete, Edson, " Plataforma de trabajo para desarrollo de aplicaciones en dispositivos móviles y, como caso de estudio, juego de ajedrez", proyecto terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco, México, 2010

[22] Montaño Ayala, Oscar Xahil, " Implementación de motor de juego para juegos tipo Tower Defense", proyecto terminal, División de CBI, Universidad Autónoma Metropolitana Azcapotzalco, México, 2011.

[23] *Hipertextual ebooks* [En línea] 16 de Junio, 2011. [Citado el: 21 de Noviembre, 2011]. http://alt1040.com/2011/06/app-store-y-android-market-empatan-en-generacion-de-ingresos

[24] altonivel [En línea] 20 de Agosto, 2010. [Citado el: 21 de Noviembre, 2011]. http://www.altonivel.com.mx/5590-juegos-en-moviles-o-consolas.html

[25] CRONISTA.COM [En línea] 17 de Noviembre, 2011. [Citado el: 21 de Noviembre, 2011]. http://www.cronista.com/tecnologia/Los-juegos-para-celulares-ya-mueven-mas-dinero-que-los-de-la-Play-y-la-Wii-20111117-0011.html

[26] Flurry Blog [En línea] 09 de Noviembre, 2011. [Citado el: 21 de Noviembre, 2011]. http://blog.flurry.com/

[27] ¿Qué es la tecnología Java y por qué lo necesito? [En línea] [Citado el: 24 de Noviembre, 2011]. http://www.java.com/es/download/faq/whatis_java.xml

[28] About J2ME [En línea] [Citado el: 24 de Noviembre, 2011]. http://www.oracle.com/technetwork/java/javame/about-java-me-395899.html

[29] Mobile Information Device Profile (MIDP); JSR 37, JSR 118 Overview [En línea] [Citado el: 25 de Noviembre, 2011]. http://www.oracle.com/technetwork/java/overview-140208.html

[30] What's New in MIDP 2.0 [En línea] [Citado el: 25 de Noviembre, 2011]. http://www.oracle.com/technetwork/java/whatsnew-138562.html

[31] Connected Limited Device Configuration (CLDC), JSR 139 [En línea] [Citado el: 25 de Noviembre, 2011]. http://java.sun.com/products/cldc/

[32] Unidad de Investigación ACCESO [En línea] 20 de Abril, 2011. [Citado el: 26 de Noviembre, 2011]. http://acceso.uv.es/accesibilidad/artics/web/01-png.htm

[33] *MID Profile* [En línea] 2006. [Citado el: 27 de Noviembre, 2011]. http://docs.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/javax/microedition/lcdui/game/Sprite.html

[34] *Managing the MIDlet Life-Cycle with a Finite State Machine* [En línea] Agosto, 2004. [Citado el: 27 de Noviembre, 2011]. http://developers.sun.com/mobility/midp/articles/fsm/