UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

DESARROLLO DE JUEGO EDUCATIVO RPG EN TELÉFONOS MÓVILES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL
EN COMPUTACIÓN

MATÍAS DANIEL ESPINOZA VIVANCO

PROFESOR GUÍA:
JAIME SÁNCHEZ ILABACA

MIEMBROS DE LA COMISIÓN:
LUIS GUERRERO BLANCO
NELSON ANTRANIG BALOIAN TATARYAN

SANTIAGO DE CHILE
OCTUBRE 2009

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN
POR: MATÍAS DANIEL ESPINOZA VIVANCO
FECHA: 02/10/2009
PROF. GUÍA: Sr. JAIME SÁNCHEZ ILABACA

"DESARROLLO DE JUEGO EDUCATIVO RPG EN TELÉFONOS MÓVILES"

Históricamente las actividades lúdicas han estado presentes en diversas culturas. Desde un principio, las actividades lúdicas fueron asociadas al entretenimiento y la diversión, pero este concepto fue cambiando cuando éstas comenzaron a ser utilizadas como metodología de enseñanza. El videojuego es una expresión del medio digital y por ende es parte de la cultura en el presente, razón por la cual debe ser parte de la formación de las personas del siglo XXI.

Resulta interesante explorar los potenciales educativos de los videojuegos, donde el estudiante aprende de las experiencias ganadas a través de las tareas realizadas dentro del juego, rompiendo el esquema tradicional de enseñanza. Esto apunta claramente en el sentido de las expectativas de los niños de hoy, ya que ellos están inmersos en un mundo rodeado de tecnología y elementos multimedia, potenciados por la alta interactividad entre estos. Estas expectativas son un factor muy importante, lo cual hace ver al sistema tradicional de educación muchas veces irrelevante en la vida de los niños de hoy. En esta búsqueda se pretendió enganchar a los niños, a aprender a través de un videojuego educativo RPG, contribuyendo a mejorar las formas tradicionales de aprendizaje.

En consecuencia, se exploró en el uso de teléfonos móviles para fines educacionales, a través del diseño, desarrollo y evaluación de un videojuego educativo del tipo RPG, en el cual se abordaron contenidos de ciencias. El videojuego fue pensado para un uso individual, donde el usuario puede recorrer libremente un mundo virtual que se compone de elementos tales como escenarios, personajes e ítems; en el juego es posible interactuar con personajes virtuales, además de tomar y utilizar objetos con algún fin. Una característica importante que fue incorporada, consistió en el desarrollo de un motor de videojuegos, cuya función consiste en controlar los videojuegos diseñados. Este motor de videojuegos facilita la creación y edición de nuevos videojuegos, y utilizando este motor se diseñaron y realizaron los videojuegos que luego fueron probados por usuarios finales.

En la actualidad existe una amplia gama de teléfonos móviles con diversas características. Debido a esto, se fijaron características base en los teléfonos móviles, definiendo de esta forma el hardware objetivo, con lo que se buscó abarcar transversalmente la mayor cantidad de teléfonos móviles que soportan la implementación de la solución desarrollada, con la intención de maximizar la cantidad de usuarios que puedan utilizar la aplicación obtenida.

Los resultados dieron cuenta de la obtención de una herramienta educativa que en general resultó fácil de usar, que gustó y que logró motivar a los alumnos que trabajaron con ella. Esto, en conjunto con la posibilidad de crear nuevos videojuegos utilizando el motor de juegos implementado y la portabilidad de la solución obtenida, la que permite abarcar una gran cantidad de dispositivos, abren la posibilidad de explotar masivamente el potencial educativo de esta herramienta a futuro.

La actitud con la que enfrentas los desafíos es tan importante como los resultados obtenidos; la forma como ganas o pierdes es el reflejo de la esencia de tu persona.

Dedicado…
…a mis padres, por brindarme siempre su apoyo y cariño.
…y a mi novia, por ser la estrella que ha iluminado mi camino.

Muchas Gracias…
…a mis compañeros de trabajo del C5, en especial a Mauricio, Héctor, Ruby, Carolina, Natalia y Angelo por su apoyo y aporte en este trabajo.
…a todos los niños del colegio José Joaquín Prieto Vial que jugaron con el software.

Índice

1. Antecedentes 7
1.1. Introducción 7
1.2. Motivación 8
1.3. Definición del problema 9
1.4. Objetivos 10
1.4.1. Objetivo General 10
1.4.2. Objetivos Específicos 10
2. Marco Teórico 11
2.1. Resolución de problemas 11
2.1.1. Pasos en la resolución de problemas 11
2.2. Videojuegos de Rol 13
2.3. Hiperhistorias 15
2.4. Dispositivos móviles 17
2.4.1. Teléfonos móviles 18
2.5. Java 2 Micro Edition como lenguaje de desarrollo 19
2.6. Programación orientada a objetos 22
2.7. Patrones de diseño 24
2.8. Metodologías de usabilidad 25
2.9. Trabajos referentes 26
3. Requerimientos 27
3.1. Alcances 27
3.2. Desafíos 28
4. Diseño de la solución 30
4.1. Idea 30

4.2. Componentes 32
4.3. Relación entre los componentes del juego 34
4.4. Diseño de los elementos gráficos del videojuego 34
4.5. Relación entre el diseño de las imágenes y las componentes 36
4.6. Modelo de Interacciones 37
4.7. Integración de la resolución de problemas 40
4.8. Jugabilidad 41
5. Implementación 44
5.1. Software de desarrollo 44
5.2. Hardware objetivo 44
5.3. Arquitectura de la solución 46
5.3.1. Utilización de patrón de diseño MVC 46
5.3.2. Modelo 48
5.3.3. Controlador 51
5.3.4. Vista 52
5.4. Interfaces del usuario 54
5.4.1. Visualización de los componentes en pantalla 54
5.4.2. Botones de entrada 55
5.4.3. Comportamiento de las componentes 56
5.5. Construcción del Prototipo 1 57
5.6. Construcción del Prototipo 2 59
6. Evaluación 65
6.1. Fundamentación Teórica 65
6.2. Evaluación previa 66
6.2.1. Objetivos 66

6.2.2. Muestra 66
6.2.3. Instrumentos 66
6.2.4. Procedimiento 67
6.2.5. Resultados 67
6.3. Evaluación del prototipo 1 67
6.3.1. Objetivos 68
6.3.2. Muestra 68
6.3.3. Instrumentos 69
6.3.4. Procedimiento 69
6.3.5. Resultados 71
6.3.6. Análisis 72
6.4. Evaluación del prototipo 2 72
6.4.1. Objetivos 73
6.4.2. Muestra 73
6.4.3. Instrumentos 73
6.4.4. Procedimiento 74
6.4.5. Resultados 75
6.4.6. Análisis 77
7. Conclusiones y trabajo futuro 79
8. Referencias 81
9. Anexos 84
9.1. Pauta de evaluación de usuario final 84
9.2. Pauta adaptada de evaluación de usuario final 85
9.3. Pauta de observación 1 y encuesta feedback de usuario 86
9.4. Pauta de observación 2 86

9.5. Interfaz iLogica 87

9.6. Función setDatos() del prototipo 1 89

9.7. Función setDatos() del prototipo 2 90

1. Antecedentes

1.1. Introducción

Históricamente las actividades lúdicas han estado presentes en diversas culturas. Desde un principio las actividades lúdicas fueron asociadas al entretenimiento y la diversión, pero este concepto fue cambiando cuando éstas comenzaron a ser utilizadas como metodología de enseñanza [1]. El videojuego es una expresión del medio digital y por ende es parte de la cultura en el presente, razón por la cual debe ser parte de la formación de las personas del siglo XXI [2].

Los juegos tienen un potencial educativo importante, además de motivar, permiten a las personas aprender y desarrollar destrezas, habilidades y estrategias [1]. Las posibilidades de aplicación de videojuegos para el aprendizaje son amplias. Algunos aspectos donde los videojuegos pueden contribuir al aprendizaje son: conocimiento de alfabetización digital, habilidades de juego a través de la resolución de problemas, habilidades de comprensión y habilidades académicas, entre otras [1] [2].

Como señalamos anteriormente son diversas las potencialidades educativas que se pueden explotar de los videojuegos. El hecho de involucrar al jugador en un mundo virtual donde se realizan tareas que incluyen el aprendizaje de las características de este nuevo mundo y de las acciones asociadas a él, con el fin de progresar u obtener cosas nuevas, dejan en evidencia las características educativas explotables en los videojuegos.

En este sentido si un videojuego lo enfocamos al aprendizaje de un área específica podemos lograr que los niños puedan aprender contenidos a través de las experiencias ganadas con el desarrollo de tareas dentro del juego. De esta forma la enseñanza va más allá de la manera tradicional como la conocemos y el conocimiento se adquiere a través de ganar experiencia en las situaciones del juego, lo cual puede ayudar a los estudiantes a ser más efectivos en situaciones fuera del aula donde podrían aplicar estos conceptos. El concepto de “aprender haciendo” es una de las ventajas educativas que ofrecen los videojuegos. Evidencias de esto se pueden encontrar en juegos como Full Spectrum Warrior and America’s Army desarrollados por el ejército de Estados Unidos donde se busca introducir a civiles a una mirada militar del mundo; otros ejemplos son juegos para aprender historia (Making History) e ingeniería (Time Engineers) [14][15][16].

En un videojuego RPG (del inglés role-playing game) un jugador desempeña un rol a través de un personaje dentro de un mundo virtual a lo largo de una historia, la cual sigue un curso de acuerdo a sus decisiones y acciones; el personaje puede ganar experiencia, ítems y otras características a lo largo del juego, de acuerdo a las tareas que desempeña en este mundo virtual [13]. Y precisamente las características de este tipo de videojuego pueden aprovechar las potencialidades educativas señaladas anteriormente, a través de la adquisición de conocimiento con el desarrollo de las tareas dentro del juego. A través de un videojuego RPG se pueden desarrollar actividades de

aprendizaje tales como la imitación, la retroalimentación, el entrenamiento, la práctica, la revisión de casos, los desafíos incrementales y la inmersión, entre otras, abarcando el desarrollo de contenidos tales como habilidad, criterio, comportamiento, lenguaje y comunicación [47].

Tanto los computadores, como las PDAs[1], los teléfonos móviles y otros artefactos tecnológicos, forman parte de la manera en que se transforma el trabajo, las relaciones sociales, la elaboración del conocimiento, el aprendizaje y la educación [2]. En particular los teléfonos móviles se han masificado muy fuertemente durante el último tiempo, aumentando las aplicaciones desarrolladas para estos y en consecuencia ampliándose sus usos. La utilización de teléfonos móviles en la educación se puede considerar como parte de la evolución del uso de estos dispositivos.

En nuestro país existen evidencias de experiencias de uso de tecnologías en educación [7][8][9][10][11]. En particular tenemos casos del uso de PDAs en educación a través aplicaciones lúdicas y trabajo colaborativo con alumnos [10][11].

Para el presente trabajo de título se exploró el uso de teléfonos móviles en educación a través del desarrollo, en estos dispositivos, de un videojuego educativo del tipo RPG.

1.2. Motivación

La motivación para este trabajo se basó en 3 factores: la manera en cómo influye un videojuego sobre los procedimientos y estrategias de aprendizaje, la movilidad que proporciona al usuario la tecnología utilizada como plataforma, y el grado de masividad de esta última.

Resulta interesante explorar los potenciales educativos de los videojuegos, donde el estudiante aprende de las experiencias ganadas a través de las tareas realizadas dentro del juego, rompiendo el esquema tradicional de enseñanza. Esto apunta claramente en el sentido de las expectativas de los niños de hoy, ya que ellos están inmersos en un mundo rodeado de tecnología y elementos multimedia, potenciados por la alta interactividad entre estos. Estas expectativas son un factor muy importante, lo cual hace ver al sistema tradicional de educación muchas veces irrelevante en la vida de los niños de hoy. En esta búsqueda se pretendió enganchar a los niños, a aprender a través de un juego RPG, contribuyendo a mejorar las formas tradicionales de aprendizaje [17][18][19][20][47].

[1] PDAs (del inglés Personal Digital Assistant, es un computador de mano originalmente diseñado como agenda electrónica con un sistema de reconocimiento de escritura; hoy día se puede usar como una computadora doméstica: ver películas, crear documentos, juegos, correo electrónico, navegar por Internet, reproducir archivos de audio, etc.)

Por otra parte es vital determinar la manera de abordar correctamente la influencia de un videojuego sobre el aprendizaje. Se ha identificado que esto depende de 3 aspectos que afectan directamente la motivación del usuario: presentar desafíos a través de metas, crear curiosidad a través de diversas alternativas y generar fantasía mental. Hay evidencias de que controlando adecuadamente estos aspectos, se puede contribuir efectivamente en áreas del aprendizaje tales como el desarrollo personal y social, el lenguaje y alfabetización, el desarrollo matemático, el desarrollo creativo, el conocimiento y la comprensión del mundo, entre otros [1].

El factor movilidad también es importante. Los dispositivos livianos y transportables como las PDAs y los teléfonos móviles pueden ser utilizados tanto dentro como fuera del aula. Además, estos dispositivos disponen de lo necesario para manejar la información, generar colaboración y fomentar la construcción del conocimiento en horario escolar y/o extraescolar.

Finalmente, otro de los factores importantes al aplicar tecnología en educación es el costo de la plataforma para la cual se desarrollan estas aplicaciones, lo que afecta en consecuencia la cantidad de personas que pueden beneficiarse a través del uso de las aplicaciones desarrolladas.

Muchas aplicaciones educativas desarrolladas para PC y Pocket PC marginan a usuarios de menores recursos que no disponen de estas tecnologías. En este sentido los teléfonos móviles aparecen como una alternativa más económica y accesible. A esto se suma que la masividad de los teléfonos móviles ofrece una relación alumno/dispositivo cercana al 1:1, lo que representa una gran oportunidad para desarrollar aplicaciones orientadas a la educación basada en este tipo de tecnología.

El hecho de encausar estos factores con fines educacionales a través del desarrollo de un videojuego educativo RPG es lo que genera valor agregado. Esto se debe a que en general los videojuegos para teléfonos móviles solamente tienen fines lúdicos y no es común encontrar videojuegos con fines educativos con las características desarrolladas en esta memoria.

1.3. Definición del problema

En este trabajo se quiso responder las siguientes preguntas:

- ¿Es posible desarrollar un videojuego educativo para teléfonos móviles que apoye efectivamente las actividades escolares de los alumnos?
- ¿Es posible motivar a alumnos a través de una herramienta educativa para teléfonos móviles?
- ¿Es posible obtener aplicación educativa desarrollada para teléfonos móviles para a una cantidad considerable de usuarios?
- ¿Es factible facilitar la edición del contenido del videojuego para distintos objetivos curriculares?

1.4. Objetivos

1.4.1. Objetivo General

El objetivo general de este trabajo fue:

- Construir un juego educativo RPG para teléfonos móviles que apoye el aprendizaje escolar.

1.4.2. Objetivos Específicos

Los objetivos específicos de este trabajo fueron:

- Diseñar e Implementar un motor de videojuegos educativos RPG para teléfonos móviles.
- Diseñar e Implementar un videojuego educativo RPG que funcione sobre un motor de videojuegos para teléfonos móviles.
- Evaluar la usabilidad de un videojuego RPG para teléfonos móviles en sus etapas de desarrollo.

Con ello también se quiso:

- Obtener una herramienta que apoye el aprendizaje escolar a través de la estimulación de la capacidad de resolución de problemas del usuario.
- Lograr que la aplicación sea portable a la mayor cantidad de modelos de celulares.
- Facilitar la edición y creación de distintos videojuegos, en sus contenidos y objetivos educativos.

2. Marco Teórico

2.1. Resolución de problemas

Un problema suele ser un asunto del que se espera una rápida y efectiva solución. Los problemas pueden corresponder a distintos ámbitos como por ejemplo las matemáticas, la filosofía y la investigación científica, entre otros.

Conceptualmente no es lo mismo hacer un ejercicio que resolver un problema. En los ejercicios se aplica un algoritmo o proceso rutinario de forma más o menos mecánica. Resolver un problema, en cambio, significa dar una explicación coherente a un conjunto de datos relacionados dentro de un contexto, para lo cual se hace una pausa, se reflexiona y hasta puede que se ejecuten pasos originales que no se habían ensayado antes para llegar a la respuesta. Esta característica de dar una especie de paso creativo en la solución, sin importar que tan pequeño sea es lo que distingue un problema de un ejercicio [27][28].

2.1.1. Pasos en la resolución de problemas

George Polya introduce en sus trabajos una serie de pasos para resolver un problema. Él propuso que en cada uno de estos pasos es conveniente plantearse algunas preguntas con respecto al problema. A continuación se muestran los pasos propuestos por Polya, con las respectivas preguntas que deberían plantearse en cada uno de estos [28][29]:

- **Comprensión e Identificación del problema**

  1. *"¿Entiendes todo lo que dice?*
  2. *¿Puedes replantear el problema en tus propias palabras?*
  3. *¿Distingues cuáles son los datos?*
  4. *¿Sabes a qué quieres llegar?*
  5. *¿Hay suficiente información?*
  6. *¿Hay información extraña?*
  7. *¿Es este problema similar a algún otro que hayas resuelto antes?"* [28]

- **Concepción de un plan para la resolución del problema:** ¿Puedes usar alguna de las siguientes estrategias? (Una estrategia se define como un artificio ingenioso que conduce a un final).

  1. *"Ensayo y Error (Conjeturar y probar la conjetura).*
  2. *Usar una variable.*
  3. *Buscar un Patrón*
  4. *Hacer una lista.*
  5. *Resolver un problema similar más simple.*

6. *Hacer una figura.*
7. *Hacer un diagrama*
8. *Usar razonamiento directo.*
9. *Usar razonamiento indirecto.*
10. *Usar las propiedades de los Números.*
11. *Resolver un problema equivalente.*
12. *Trabajar hacia atrás.*
13. *Usar casos*
14. *Resolver una ecuación*
15. *Buscar una fórmula.*
16. *Usar un modelo.*
17. *Usar análisis dimensional.*
18. *Identificar sub-metas.*
19. *Usar coordenadas.*
20. *Usar simetría."* [28]

- **Realización de un plan para la resolución del problema**

1. *"Implementar la o las estrategias que escogiste hasta solucionar completamente el problema o hasta que la misma acción te sugiera tomar un nuevo curso.*
2. *Concédete un tiempo razonable para resolver el problema. Si no tienes éxito solicita una sugerencia o haz el problema a un lado por un momento (¡puede que se te prenda el foco cuando menos lo esperes!).*
3. *No tengas miedo de volver a empezar. Suele suceder que un comienzo fresco o una nueva estrategia conducen al éxito."* [28]

- **Evaluación de la solución del problema**

1. *"¿Es tu solución correcta?*
2. *¿Tu respuesta satisface lo establecido en el problema?*
3. *¿Adviertes una solución más sencilla?*
4. *¿Puedes ver cómo extender tu solución a un caso general?"* [28]

El proceso puede ser visto más claramente a través del diagrama de la figura 1.

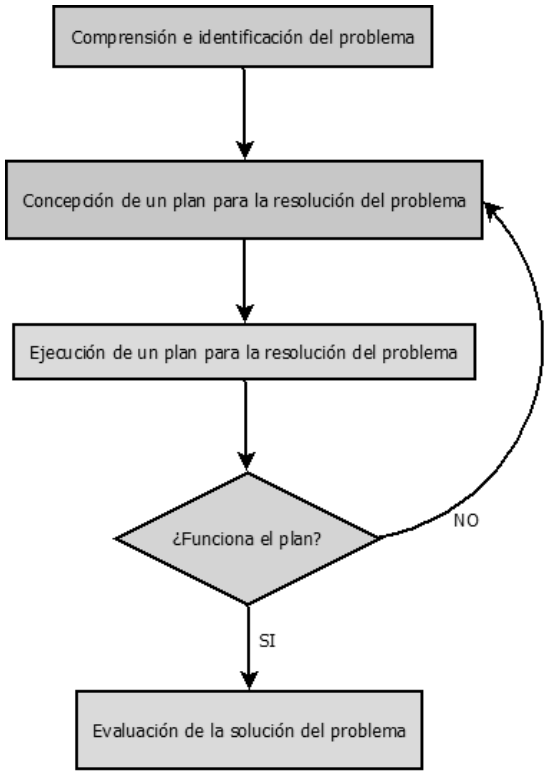


Figura 1: Método de Cuatro Pasos de Polya para la Resolución de Problemas.

2.2. Videojuegos de Rol

Un **videojuego** (del inglés video game) es un programa de computación creado para el entretenimiento, basado en la interacción entre una o varias personas y un aparato electrónico (ya sea un ordenador, un sistema arcade, una videoconsola, un dispositivo handheld o actualmente un teléfono celular) que ejecuta dicho videojuego. En muchos casos, estos videojuegos recrean entornos y situaciones virtuales en los que el jugador puede controlar a uno o varios personajes (o cualquier otro elemento de dicho entorno), para conseguir uno o varios objetivos por medio de ciertas reglas determinadas.

Un **juego de rol** (del inglés role-playing game) es un juego interpretativo-narrativo en el cual los jugadores asumen el rol de un personaje a lo largo de una historia o trama, para lo cual interpretan sus diálogos y describen sus acciones.

Un **videojuego de rol** es un género de videojuegos que usa elementos del juego de rol tradicional, llevando al usuario a un mundo virtual donde a través de un personaje protagonista (el cual es controlado por el usuario), interpreta un papel con el cual ha de mejorar sus habilidades, ganar experiencia, ítems y otras características a medida que interactúa con el entorno y otros personajes a lo largo de una historia. Esta historia sigue un curso de acuerdo a sus decisiones y acciones [13].

En la actualidad los **videojuegos de rol** se pueden encontrar para distintas consolas, como también para PC y para diferentes tipos dispositivos móviles como los teléfonos móviles y las PDAs. En este tipo de videojuego destacan títulos como World of Warcraft, The Elder Scrolls 3 Morrowind, Diablo, Star Wars Knights of the Old Republic, Ultima Series, Fallout, Planescape Torment, Baldur's Gate Series, Final Fantasy Series, Fable**,** Might & Magic 6 The Mandate of Heaven, y Neverwinter Nights, entre otros [49][50]. También existen videojuegos educativos de rol que son producto de adaptaciones de otros videojuegos, para fines educacionales, como es el caso de Revolution (adaptación de Neverwinter Nights Gold) y Arden (adaptación de Neverwinter Nights Diamond), entre otros [51]. Entre los videojuegos de rol desarrollados para teléfonos móviles destacan títulos tales como Age of Heroes III Orc's Retribution, Wolfenstein RPG, The Idhun Chronicles, The Sims 2, Orcs and Elves 2, entre otros [52].

Un **videojuego de rol** está compuesto por distintos elementos. A continuación se introducen algunos conceptos que identifican los elementos gráficos más relevantes que podemos encontrar en un videojuego de rol:

- ***Protagonista***: corresponde al personaje controlado por el jugador (este elemento es llamado también como Personaje Jugador).

- ***Personaje***: corresponde a los personajes que no están bajo el control directo del jugador en un juego de rol; su comportamiento (que se compone generalmente por acciones específicas y parlamentos) se encuentra por lo general prescrito y es automático, y se gatilla a partir de las acciones del ***Protagonista*** (este elemento es llamado también como Personaje no Jugador**)**.

- ***Ítem***: corresponden a elementos que pueden ser coleccionados y/o ser utilizados para lograr algún fin durante el juego.

- ***Decorativo***: corresponde a elementos gráficos que tienen la función de "decorar" el mundo virtual mejorando la percepción del entorno de la historia; también sirven para restringir los movimientos del ***Protagonista***, ya que el ***Protagonista*** no puede moverse sobre ellos, debido a que se producen colisiones con estos elementos (este elemento es llamado también como Sólido).

- ***Escenario***: los distintos escenarios conforman el mundo virtual completo donde el ***Protagonista*** puede desplazarse. Sobre la superficie de cada escenario encontramos ***Personajes***, ***Ítems***, ***Decorativos*** y al ***Protagonista*** (solo si este último se está desplazando sobre el escenario).

Finalmente es importante introducir el termino **motor de juego** (del inglés *game engine*) el cual es un término que hace referencia a una serie de rutinas de programación que permiten el diseño, la creación y la representación de un videojuego.

De esta forma el manejo relacional de los elementos gráficos y parlamentos asociados a personajes, ítems u otros elementos ocurre a través de un motor de juego.

2.3. Hiperhistorias

Los autores Sánchez & Lumbreras en su trabajo[30] introducen una definición de hiperhistoria: *"Una Hiperhistoria es la combinación de un mundo virtual, donde el aprendiz puede navegar, un conjunto de objetos sobre los cuales el usuario puede realizar ciertas operaciones y un conjunto de personajes que pueden ser manipulados por el aprendiz. Una Hiperhistoria incluye a uno o varios contextos navegables (ambiente hipermedial) y objetos y personajes (entidades). Los objetos y protagonistas pueden tener su propio comportamiento y actuar en forma autónoma, eventualmente en forma concurrente con el comportamiento del protagonista. El aprendiz (cuando manipula uno o más protagonistas) puede también interactuar con otros personajes, a fin de resolver un problema particular o lograr una meta*.

Considerando la naturaleza del aprendiz y el dominio de aplicación de las Hiperhistorias, resulta obvio que si el aprendiz es familiar con el mundo virtual, se vuelva fácil por él usarlo. Ambientes familiares, como el barrio del aprendiz, la escuela, la cancha, el parque, etc., pueden ser metáforas interesantes para construir mundos virtuales" [30].

"Las Hiperhistorias se navegan usando una metáfora basada en los Hipertextos y constituyen un buen ejemplo de herramientas multimediales educativas. Además, proporcionando diferentes interfaces por el mismo mundo virtual es posible adaptar el ambiente a los aprendices con necesidades cognitivas especiales. Como resultado, se obtiene que sobre el mismo mundo virtual sea posible construir diferentes Hiperhistorias para satisfacer una diversidad de demandas. Para lograr esto, se requiere que el diseño contemple una clara separación del contenido de la historia con respecto a la presentación y manejo de la interfaz" [30].

A continuación se introduce una definición de los términos hipertexto e hipermedio utilizados por las definiciones anteriormente mostradas: "*El hipertexto ha sido definido como un enfoque para manejar y organizar información, en el cual los datos se almacenan en una red de nodos conectados por enlaces. Los nodos contienen textos y si contienen además gráficos, imágenes, audio, animaciones y video, así como código ejecutable u otra forma de datos se les da el nombre de hipermedio, es decir, una generalización de hipertexto."* [32]

2.3.1. Diseño de Hiperhistorias

Los autores Sánchez & Lumbreras en sus trabajos sobre hiperhistorias [30][31] recalcan una serie de requerimientos a cumplir necesarios para poder introducir las

guidelines para el diseño de hiperhistorias elaboradas por ellos. A continuación se citan estos elementos:

Las hiperhistorias implican el cumplimiento de ciertos requerimientos tales como:

- "*Separación de la interfaz del contenido de la historia.*
- *Composición, modularidad y herencia entre entidades.*
- *Soporte de eventos concurrentes.*
- *Independencia entre la especificación y el lenguaje de implementación.*
- *La interfaz usuaria implementada en un lenguaje diferente de implementación de la historia.*
- *Objetos con conducta dinámica y autónoma.*
- *Comunicación sincrónica y asincrónica entre entidades.*" [30][31]

Con estos requerimientos sobre la mesa los autores Sánchez & Lumbreras presentaron los siguientes guidelines para el diseño de hiperhistorias [31]:

- **La idea:**
  - *"1) Describir alternativas posibles en el curso de la historia, teniendo en cuenta que eventos originan los cambios en el desarrollo, con una representación conceptual adecuada de los ambientes y de la actitud y/o actividad de otras entidades (fase de escritura de la Hiperhistoria, utilizando lenguaje coloquial)"* [31].

- **La caracterización:**
  - *"2) Determinar qué contextos existen en la historia, su conectividad y las entidades que allí existirán (fase de modelado del ambiente, utilizando notación gráfica para dibujar contextos y links, y especificando clases de contextos con la notación establecida en el modelo)"* [31].

  - *"3) Asignar características relevantes a cada personaje, y utilizar estas consistentemente para reflejar diferentes habilidades o point of view (Fase de caracterización de personajes, se describe informalmente qué cosas puede hacer y/o actitud de cada uno de ellos, en especial si el usuario puede, para diferentes sesiones, utilizar diferentes personajes)"* [31].

  - *"4) Obtener atributos de cada entidad y comportamiento de ella para cada bloque. (Fase de modelado formal de entidades, se indica qué eventos puede recibir, cómo va a responder y qué atributos se van a modificar, utilizando el formalismo de descripción de entidades visto en el modelo propuesto)"* [31].

- **La codificación:**

    - *"5) Codificar el comportamiento de cada entidad, asociándolo a diferentes bloques de comportamiento y anidándolos si es necesario. Resolver problemas de timing y sincronización (Fase de compresión y timing, se utilizan eventos que involucran el timer y sincronización a través de flags y envío/recepción de mensajes)"* [31].

- **El remodelado:**

    - *"6) Enriquecer la historia con características de entidades de comportamiento autónomo, basándose en el uso de entidades que poseen actividades temporizadas guiadas, teniendo en cuenta que las otras entidades pueden/deben poder aceptar estos nuevos eventos. (Fase de enriquecimiento; se extiende la Hiperhistoria recreando los pasos 4) y 5) para las nuevas entidades y extendiendo la funcionalidad para las entidades preexistentes)"* [31].

    - *"7) Especificar canales de comunicación explícitos entre objetos que admiten interconexión por canales, de tal manera de describir la mecánica de interacción entre ellos"* [31].

- **Interfaz:**

    - *"8) Realizar un boceto del aspecto que tendrá cada contexto y del aspecto que ofrecerá cada entidad que está en la interfaz"* [31].

2.4. Dispositivos móviles

Un dispositivo móvil es un aparato suficientemente pequeño como para facilitar considerablemente su utilización y transporte (en su mayoría pueden ser transportados en el bolsillo). Normalmente estos aparatos pueden ser sincronizados a través de un PC, permitiendo actualizar sus aplicaciones y datos. [21][22]

Los dispositivos móviles se caracterizan por poseer algunas capacidades de procesamiento y tener una memoria limitada; además pueden tener una conexión permanente o intermitente a una red y aunque están diseñados específicamente para una función, pueden llevar a cabo otras funciones más generales. En general poseen una pantalla y botones pequeños, aunque algunos carecen totalmente de botones y se manejan con pantallas táctiles. [22][48]

Algunos de estos dispositivos son los siguientes:

- *"Paginadores.*
- *Comunicadores de bolsillo.*

- *Teléfonos con pantalla para Internet (Internet Screen Phones).*
- *Sistemas de navegación de automóviles.*
- *Sistemas de entretenimiento.*
- *Sistemas de televisión e Internet (WebTV).*
- *Teléfonos móviles.*
- *Organizadores y asistentes personales digitales (Personal Digital Assistant).”* [22]

2.4.1. Teléfonos móviles

A continuación se introducen las características particulares de los teléfonos móviles.

El teléfono móvil o celular es un dispositivo inalámbrico electrónico que permite tener acceso a la red de telefonía celular o móvil. Se denomina celular debido a las antenas repetidoras que conforman la red, cada una de las cuales es una célula. Su principal característica es su portabilidad, que permite comunicarse desde casi cualquier lugar. Aunque su principal función es la comunicación de voz, como el teléfono convencional, su rápido desarrollo ha incorporado otras funciones como son cámara fotográfica, agenda, acceso a Internet, reproducción de video e incluso GPS y reproductor mp3.

Es importante tener claramente identificados los pros y los contras del uso de este tipo de tecnología. A continuación se presenta un resumen de sus ventajas y desventajas:

Las ventajas que presenta un teléfono móvil como tipo de dispositivo móvil son varias: [23][48]

- Muy extendido.
- Económico.
- Fuente de alimentación desenchufada apropiada.
- Apropiado para trabajar mientras se transporta.
- Portabilidad muy apropiada, son ligeros y fáciles de transportar.
- Apropiado para trabajar mientras se camina.
- Apropiado para el uso en lugares incómodos.
- Tiempo de arranque del dispositivo apropiado.

Por el contrario, también muestran varios inconvenientes: [23][48]

- Capacidad de entrada de datos inapropiada.
- Soporte de multimedia insatisfactorio.
- Capacidad de almacenamiento de memoria volátil insatisfactorio.
- Capacidad de almacenamiento deficiente.
- Poder de procesamiento inapropiado, poca potencia de proceso.

- Tamaño de pantalla insatisfactorio.
- Capacidad de uso de MANET (Mobile Ad-hoc Network) insatisfactorio.
- Interacción avanzada difícil.

2.5. Java 2 Micro Edition como lenguaje de desarrollo

La plataforma **Java 2 Micro Edition** (**J2ME**) es una familia de especificaciones que definen varias versiones minimizadas de la plataforma **Java 2 Standard Edition** (**J2SE** que es la plataforma estándar de Java usada en sistemas de escritorio y servidor); estas versiones minimizadas pueden ser usadas para programar aplicaciones en dispositivos electrónicos, como pueden ser teléfonos celulares, PDAs, tarjetas inteligentes, etc. Estos dispositivos presentan en común que no disponen de abundante memoria ni mucha potencia en el procesamiento, y que tampoco necesitan de todo el soporte que brinda **J2SE** [5][24]. De esta forma J2ME es una versión reducida de Java 2 que se adapta a las características físicas de los pequeños dispositivos [25].

El lenguaje de programación Java permite escribir un programa una vez y poder ejecutarlo en multitud de ordenadores, con diferentes plataformas sin tener que compilarlo de nuevo. Esa es una gran ventaja y una característica muy deseable en el entorno de los pequeños dispositivos, por lo que se ha exportado esa filosofía a estos aparatos. Así, mediante **J2ME** se podrán escribir aplicaciones para una gran variedad de dispositivos pequeños diferentes [25].

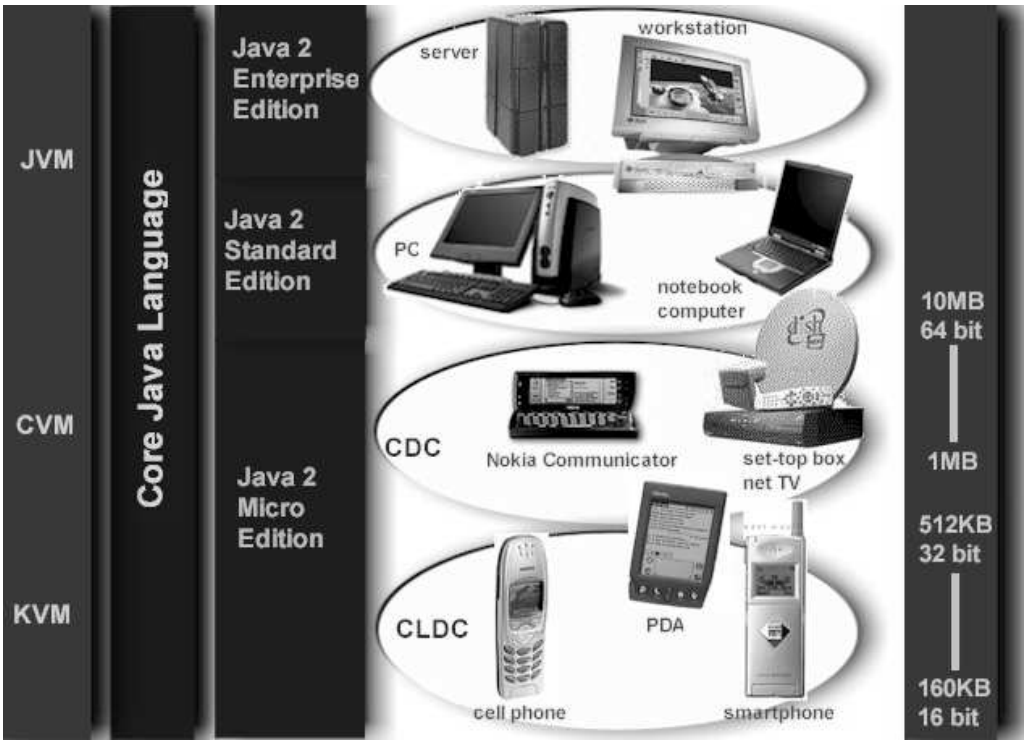


Figura 2: La plataforma Java y la relación entre sus ediciones y dispositivos [33].

J2ME está dividida en **configuraciones**, **perfiles** y **APIs opcionales**, los que al ser combinados se optimizan para la memoria, potencia de proceso y capacidades de E/S de una categoría de dispositivos determinada [34][35].

Una **API** (del inglés Application Programming Interface), es una interfaz de programación de aplicaciones la cual corresponde a un conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Una **máquina virtual** es un software que emula a un ordenador y puede ejecutar programas como si fuese un ordenador real.

Las **configuraciones** son especificaciones que se componen de una **máquina virtual** y un conjunto base de **APIs** que pueden ser usadas en cierta clase de dispositivos. La maquina virtual puede ser completa, como la describe la especificación o algún derivado de ella [34][35].

Actualmente **J2ME** tiene disponibles dos **configuraciones**:

- **Connected Limited Device Configuration (CLDC):** *"está diseñada para dispositivos con conexiones de red intermitentes, procesadores lentos y memoria limitada: teléfonos móviles, asistentes personales (PDAs), etc. Es habitual que estos dispositivos tenga CPUs de 16 o 32 bits y un mínimo de entre 128 y 256 KB de memoria disponible para la implementación de la plataforma Java y sus aplicaciones asociadas. Está basada en la máquina virtual K (K Virtual Machine, KVM)."*[34]

- **Connected Device Configuration (CDC):** *"está diseñada para dispositivos que tienen más memoria, procesadores más rápidos y un ancho de banda mayor, como Set-top boxes, pasarelas residenciales, asistentes personales de gran capacidad, etc. Incluye una máquina virtual Java completa (Java Virtual Machine, JVM) y un subconjunto de APIs de la arquitectura J2SE mucho mayor. Se orienta a dispositivos con CPU de 32 bits y un mínimo de 2 MB de memoria disponible para la plataforma Java y aplicaciones asociadas".*[34]

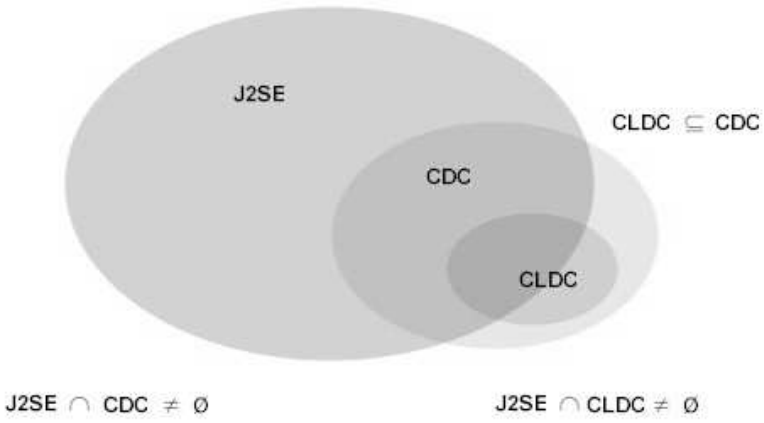


Figura 3: relación entre las configuraciones de J2ME (CDC y CDLC) y J2SE [36].

Un **perfil** trabaja sobre una **configuración** pero agrega **APIs** específicas para hacer un entorno completo de construcción de aplicaciones. Usualmente incluyen **APIs** para el ciclo de vida de las aplicaciones, interfaz de usuario y almacenamiento persistente [34][35].

En la actualidad existen los siguientes perfiles asociados a J2ME:

- **Mobile Information Device Profile (MIDP):** *"está diseñado para teléfonos móviles y PDAs con capacidades básicas. Ofrece la funcionalidad básica para las aplicaciones móviles, incluyendo la interfaz de usuario, conectividad a redes, almacenamiento local de datos y gestión del ciclo de vida de las aplicaciones. Al combinarlo con la configuración CLDC, MIDP proporciona un entorno de ejecución Java completo que incrementa la capacidad de los dispositivos móviles y que reduce el consumo de memoria y energía." [34]*

- **Foundation Profile**, **Personal Profile** y **Personal Basis Profile** son perfiles pensados para la configuración CDC.[34]

Las **APIs** opcionales sirven para poder ampliar la plataforma J2ME, combinándolas con CLDC y CDC junto con sus perfiles. "*Estos paquetes se han creado para responder a requisitos concretos de mercado y ofrecen un conjunto de APIs estándares para utilizar tanto tecnologías existentes como emergentes; entre estas se incluyen Bluetooth, servicios Web, mensajería wireless, capacidades multimedia o conectividad a bases de datos. Dado que son modulares, los fabricantes de dispositivos pueden incorporarlos según los vayan necesitando para mejorar las características soportadas."* [34]

Finalmente, es importante mencionar que a través de J2ME se aportan las mismas características técnicas del lenguaje Java disponibles en J2SE:

- **Extensión dinámica**: *"es la habilidad de un programa Java para descargar código en tiempo de ejecución, yendo a buscar nuevos ficheros de clases sustituyendo las ya existentes o simplemente añadiéndolos a las aplicaciones".* [26]

- **Seguridad:** *"Java ofrece un entorno de ejecución seguro para programas con acceso a red. La máquina virtual de Java lleva a cabo una verificación estricta del código antes de la ejecución, asegurando que éste no trata de saltarse las protecciones impuesta por el lenguaje, utilizar punteros que accedan directamente a memoria o usar el objeto equivocado."* [26]

- **Portabilidad**: *"cada dispositivo dispone de un hardware con características peculiares que hace difícil encontrar un conjunto de bibliotecas que permitan desarrollar programas más o menos independientes del soporte físico. La máquina virtual de Java asegura esta portabilidad."* [26]

- **Fiabilidad**: *"requiere la obligación de la estructuración del código en paquetes, fuertes verificaciones de compilación y ejecución (fuerte tipado, comprobación de límites en vectores, pruebas de desbordamiento de pila, etc.). Dispone un mecanismo eficiente para la gestión de excepciones y de memoria (elimina los punteros, asignación dinámica de memoria transparente al usuario y su posterior liberación -de esta manera se evitan errores)."*[26]

- **Código reutilizable**: *"debido a la orientación a objetos de Java, se consiguen características como la facilidad en el desarrollo, la reutilización del código y la mayor calidad del código".* [26]

2.6. Programación orientada a objetos

La programación orientación a objetos puede ser entendida como una forma de pensar basada en abstracciones de conceptos existentes en el mundo real (u **objetos**) aplicada a la programación. [37]

Un **objeto** es un elemento que surge del análisis del dominio de un problema a resolver. En programación un **objeto** corresponde a la instancia de una **clase**. [37]

Un objeto tiene las siguientes características:

- **Estado**: almacena los efectos de las **operaciones** ejecutadas por el **objeto**. [37]

- **Comportamiento**: es el efecto observable de una **operación** ejecutada por el **objeto**, incluyendo su resultado. [37]

- **Identidad**: es la propiedad que diferencia un **objeto** de otros. Es lo que se preserva en el **objeto** a pesar de que el estado cambie. [37]

Una **clase** es un tipo de molde o plantilla que define los valores que almacena un **objeto** en sus variables de instancia, y acciones u operaciones que se le puede aplicar a través de sus métodos. [37]

Un **atributo** es una descripción de un compartimiento de un tipo especificado en una **clase**. Cada **objeto** de esa **clase** mantiene un valor de ese tipo en forma independiente. [37]

Una **operación** es una especificación de una transformación o consulta que un **objeto** puede ser llamado a ejecutar. Tiene un nombre y una lista de **parámetros**. [37]

Un **método** es la implementación de una **operación** para una determinada **clase**. En un **método** se especifica el **algoritmo o procedimiento** que genera el resultado de la **operación**. [37]

Una **interfaz** es un conjunto de **operaciones** al que se le aplica un nombre. Una interfaz no define un estado para las instancias de estos elementos, ni tampoco asocia un método a las operaciones que declara. Cuando una clase implementa una interfaz, el conjunto de operaciones que define la interfaz caracteriza parte del comportamiento de un elemento en las instancias de dicha clase. [37]

A continuación se nombran las características más importantes de la programación orientada a objetos

- **Abstracción**: corresponde a las características esenciales de un **objeto**, donde se capturan sus comportamientos. [37]

- **Encapsulamiento**: se reúne bajo un mismo nivel de abstracción a todos los elementos que pueden considerarse pertenecientes a una misma entidad. [37]

- **Principio de ocultación**: cada tipo de objeto expone una interfaz a otros **objetos** que específica cómo pueden interactuar con los **objetos** de la **clase**. [37]

- **Polimorfismo:** es la capacidad de asociar diferentes **métodos** a la misma **operación**. De esta forma si tenemos comportamientos diferentes, asociados a objetos distintos, estos pueden compartir el mismo nombre, y al ser llamados por ese nombre se utilizará el comportamiento correspondiente al **objeto** que se esta usando. [37]

- **Herencia**: las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los **objetos** heredan las propiedades y el comportamiento de todas las **clases** a las que pertenecen. La herencia

organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de **objetos** preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a ser implementados. [37]

- **Recolección de basura**: (del inglés Garbage Collection) es la técnica por la cual el ambiente de **objetos** se encarga de destruir automáticamente, y por tanto desasignar de la memoria, los **objetos** que hayan quedado sin ninguna referencia a ellos. [37]

2.7. Patrones de diseño

Los patrones de diseño consisten en soluciones abstractas para problemas de diseño recurrentes [38][39].

Un patrón de diseño entrega una solución para un problema en un contexto [38][40]. Así los patrones de diseño constituyen la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

No es conveniente forzar el uso de los patrones de diseño ya que se puede caer en errores. Es recomendable utilizar los patrones cuando tenemos problemas iguales o similares a los que soluciona el patrón, considerando también que en un caso particular puede no ser aplicable.

2.8. Metodologías de usabilidad

La usabilidad corresponde a la aceptación práctica de las interfaces de un sistema, considerando todos sus aspectos con los que el ser humano puede interactuar. [41]

La usabilidad surge de la utilidad de un sistema (o medio para conseguir un objetivo [3]), la cual se divide en una componente de funcionalidad y otra basada en el modo en que los usuarios pueden usar dicha funcionalidad. Esta última componente es la que interesa a la hora de realizar estudios. [41]

De esta forma podemos definir la usabilidad como la medida en la cual un sistema puede ser usado por usuarios específicos para conseguir objetivos específicos en un contexto de uso especificado, a través de los siguientes atributos: [41]

- **Aprendizaje**: corresponde a la facilidad con que un sistema es aprendido de manera que el usuario realice rápidamente una tarea. [41]

- **Eficiencia**: corresponde a la eficiencia de un sistema en su uso, de manera que una vez que se ha aprendido a utilizar, pueda generarse una mayor productividad. [41]

- **Recuerdo**: corresponde a si un sistema es fácil de recordar, de manera que un usuario casual sea capaz de retornar al sistema, después de un periodo sin utilizarlo, sin tener que aprender todo de nuevo. [41]

- **Errores**: corresponde a si un sistema tiene una baja tasa de errores, de forma tal que los usuarios cometan pocos errores durante su uso y si ellos cometen errores, puedan salir fácilmente de ellos, por lo que errores catastróficos no debieran ocurrir. [41]

- **Satisfacción**: corresponde a su un sistema es placentero al utilizarlo, de forma que los usuarios están subjetivamente satisfechos cuando lo usan y les gusta. [41]

La usabilidad generalmente es medida por los usuarios finales del sistema, pero también existen revisiones realizadas por usuarios expertos. [41]

Los usuarios usan un sistema realizando tareas específicas. Con esto podemos medir la usabilidad a través del análisis de la interacción de un usuario, antes, durante y después que esta ocurra con el sistema, utilizando distintas metodologías de evaluación de usabilidad. [41]

2.9. Trabajos referentes

A continuación se describen los trabajos que contribuyeron como precedente para el diseño y desarrollo de esta memoria:

- ***Link***: es un juego RPG para PC que cuenta con audio 3D para usuarios no videntes y videntes. Es un juego que se compone de búsquedas asociadas a conceptos de ciencias, las cuales tienen como premios ítems, objetos o bien nuevas búsquedas dentro del juego. [9]

- ***Viaje a Natomía***: es un juego RPG para PC que está basado en el juego Link pero que agrega el factor colaborativo. En este juego un grupo de alumnos realiza un viaje a través del cuerpo humano, donde cada alumno recorre un sistema del cuerpo humano distinto y tienen por objetivo descifrar una enfermedad, para lo cual deben buscar ítems y utilizarlos correctamente. [44]

- ***Buinzoo*** y ***Museo***: son softwares educativos que aplican la metodología de resolución de problemas a través de una trivia de preguntas de alternativas. En ambos softwares se guía al alumno en la realización de un recorrido específico del zoológico Buinzoo y del Museo Nacional de Historia Natural respectivamente, ya que en estos lugares se encuentra la información que les permite responder las preguntas correctamente, entregándole instrucciones y ofreciendo pistas que lo ayudan a descifrar y/o encontrar la información relevante para este propósito. [43]

- ***Jugando a ser científico***: es un software educativo que impone los pasos del método científico en una investigación realizada por los alumnos en base a una simulación del proceso de crecimiento de 3 especies de plantas (cereal, hortaliza, fruta) y sobre las cuales se aplican 3 factores que afectan directamente el proceso de crecimiento (agua, luz, temperatura). Este software dirige al alumno en la aplicación de cada uno de los pasos del método científico (pregunta, hipótesis, predicción, diseño, resultados y conclusiones). [45]

3. Requerimientos

3.1. Alcances

En este trabajo se realizó lo siguiente:

- Diseño y desarrollo de **un motor de videojuegos educativos RPG para teléfonos móviles,** con el cual es posible desarrollar videojuegos con distintos ambientes y contenidos, siguiendo una hiperhistoria diseñada para un mundo virtual determinado.

- Diseño y desarrollo de **videojuegos educativos RPG de prueba** que se ejecutan sobre el **motor de videojuegos** desarrollado. En estos **videojuegos de prueba** se ejecutan las funcionalidades generales que fueron desarrolladas en el **motor de videojuegos** (por esta razón resultó equivalente referirse a las funcionalidades del motor de videojuegos o a las del videojuego de prueba). Para estos videojuegos se diseñaron hiperhistorias donde se abordaron contenidos de ciencias.

- Evaluación de la usabilidad de los **videojuegos educativos RPG** construidos sobre el **motor de videojuegos desarrollado**.

Como la aplicación desarrollada consiste en un motor de videojuegos, su diseño de software fue pensado para facilitar la incorporación y edición de elementos en el juego. Entre estos elementos encontramos los componentes del juego, las imágenes de estos componentes y las interacciones asociadas a estos componentes disponibles cuando el Protagonista del juego se encuentra con ellos.

A continuación se nombran las características generales que presenta un **videojuego** construido sobre el **motor de videojuegos educativos RPG** desarrollado en este trabajo, las cuales reflejan las funcionalidades ofrecidas por el motor de videojuegos:

- El videojuego es de uso individual y el usuario puede recorrer libremente el mundo virtual que ofrece el videojuego a través de sus escenarios disponibles.

- Visualmente el videojuego presenta al usuario un mundo virtual compuesto por ***Escenarios***, donde el ***Protagonista*** (controlado por el usuario) puede desplazarse libremente, y donde podemos encontrar en su superficie a los ***Personajes***, ***Ítems*** y ***Decorativos***. Otros elementos que se hacen visibles son los **parlamentos** de los ***Personajes***, los cuales se encontraran como información y/o pregunta referente a algún problema que requiere la solución del usuario.

- En el videojuego cuando el ***Protagonista*** colisiona con otros elementos de la superficie del ***Escenario*** puede encontrar los siguientes tipos de interacción disponibles con el elemento colisionado:

    - Recibir información.
    - Intercambiar ítems.
    - Recoger ítems.
    - Responder preguntas de alternativas.
    - Responder preguntas numéricas.

- En el juego cuando el ***Protagonista*** colisiona con un ***Personaje***, un ***Ítem*** o con un ***Decorativo*** puede gatillar a lo más una interacción por colisión con ese elemento, entre varias disponibles secuencialmente a medida que cumple ciertas condiciones que permiten cambiar de una interacción a la siguiente (también es posible que no tenga ninguna interacción disponible para gatillar). En la siguiente tabla se muestra las posibilidades de interacción que tienen sentido cuando un Protagonista colisiona con los distintos elementos del juego:

Interacción	Personaje	Ítem	Decorativo
Recibir Información	Si	Si	Si
Intercambiar Ítems	Si	No	No
Recoger Ítems	Si	Si	Si
Responder preguntas de alternativas	Si	No	No
Responder preguntas numéricas	Si	No	No

Tabla 1: Interacciones disponibles para los distintos elementos.

- El videojuego idealmente propone desafíos a los usuarios enfrentándolos a la resolución de problemas que tenga sentido en el mundo virtual del juego y enfrentando al usuario a tomar decisiones, lo que genera por ejemplo obtener nuevos ítems, que sirven para ser utilizados en una situación futura. Esto claramente depende de cómo está diseñada la hiperhistoria.

En el capítulo 4 se detalla en profundidad cómo fueron pensadas todas estas características.

3.2. Desafíos

El diseño y desarrollo de un juego educativo RPG en este trabajo de memoria presentó los siguientes desafíos:

- Diseño de la jugabilidad de la aplicación. La jugabilidad se refiere la calidad del juego en términos de sus reglas de funcionamiento y de su diseño como juego. En este sentido se pretende obtener un juego que por una parte cumpla con los objetivos previstos, pero que a su vez sea un juego entretenido que motive y que haga que los niños quieran usarlo.

- Diseño y elaboración las interfaces del juego: escenarios, personajes, ítems.

- Diseño y elaboración de hiperhistorias.

- Diseño y desarrollo de un software que se implementó en base a todos los desafíos anteriormente nombrados, y que tiene la característica de facilitar la edición de los elementos que componen el juego. Esto corresponde a la construcción del motor de juegos y del juego.

- Evaluaciones de usabilidad satisfactorias.

4. Diseño de la solución

A continuación se describe el diseño de la solución desarrollada la cual consiste en un motor para videojuego educativos RPG, para el cual se define una estructura y un set de funcionalidades generales disponibles para sus componentes base, lo que facilita la incorporación de diferentes historias, sentidos y objetivos del juego, utilizando los mismos componentes.

Para lograr esto se clasificaron los componentes del juego de acuerdo a sus funciones, estableciendo además sus reglas o restricciones de interrelación, las cuales serán vistas más adelante. Además esta clasificación facilitó varios aspectos de implementación que también serán vistos más adelante.

El diseño además contempló la inclusión de un modelo de interacción cuya función es encargarse de la hiperhistoria del juego y plasmar la metodología de resolución de problemas al mismo tiempo.

4.1. Idea

Como se ha señalado anteriormente, la aplicación que se quiere obtener consiste en un motor de videojuegos RPG. En un videojuego construido sobre este motor de videojuegos RPG, un personaje controlado por el usuario (llamado en adelante ***Protagonista***) puede recorrer libremente un **mundo virtual**.

Este **mundo virtual** está conformado por distintos ***Escenarios***, el cual a su vez está compuesto por personajes no jugadores (llamados en adelante ***Personajes***), ***Ítems*** con alguna función y otros elementos que decoran el entorno (llamados en adelante ***Decorativos***).

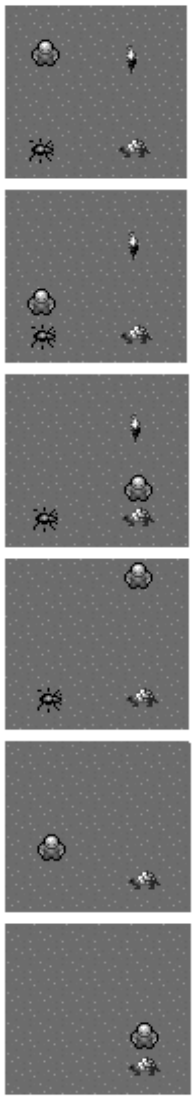
El ***Protagonista*** puede interactuar con los ***Personajes*** y a través de ellos es posible recibir *información*, recibir una *pregunta* que debe resolver, recibir ***Ítems*** e intercambiar ***Ítems*** (como parte de una acción donde el ***Protagonista*** resuelve un problema entregando algún(os) ***Ítem(s)*** y recibe como premio otros ***Ítems***), y también la posibilidad recoger (o capturar) un ***Personaje***.

El ***Protagonista*** también puede interactuar directamente con ***Ítems*** sueltos en el ***Escenario***, donde estos ***Ítems*** pueden ser recogidos y/o se puede recibir algún tipo de *información* referente al ***Ítem***.

Finalmente, el ***Protagonista*** también puede interactuar con los elementos ***Decorativos***, de los cuales puede recibir algún ***Ítem*** o recibir algún tipo de *información*.

Las ***Interacciones*** ocurren sólo si se cumplen ciertas restricciones asociadas, en caso contrario se notifica a través de un texto algo relativo a lo que está sucediendo o referente a la acción que se espera que sea realizada antes de que ocurra dicha

Interacción. Estas restricciones están asociadas a la posesión de algunos ***Ítems*** y posiblemente también en alguna cantidad determinada de dichos ***Ítems***.



1. Inicialmente tenemos un protagonista (persona), un ítem (antorcha) y dos personajes (cucaracha y tortuga).

2. El protagonista no puede capturar a la cucaracha porque no tiene la antorcha.

3. La tortuga tiene una pregunta que hacer al protagonista, pero muestra un diálogo alternativo cuando el protagonista no tiene la cucaracha.

4. El protagonista toma la antorcha directamente porque no tiene restricciones para poder recogerla.

5. El protagonista captura la cucaracha porque tiene en su poder la antorcha.

6. La tortuga hace la pregunta al protagonista porque tiene en su poder la cucaracha.

Figura 4: ejemplo de una secuencia de interacciones entre el protagonista y los personajes e ítems.

A modo de ejemplo se ilustra en la figura 4 una secuencia de ejemplos donde ocurren algunas interacciones entre un ***Protagonista*** y los ***Personajes*** e ***Ítems***. En este esquema tenemos un ***Protagonista*** (representado por el dibujo de una persona), un ítem (representado por el dibujo de una antorcha) y dos personajes (representados por los dibujos de una cucaracha y una tortuga respectivamente). La cucaracha tiene como interacción el ser capturada (o recogida) con la restricción de que el protagonista debe tener la antorcha en su poder; por esta razón cuando el protagonista no tiene la antorcha, al interactuar con la cucaracha esta no permite ser capturada. La antorcha a su vez tiene como interacción ser recogida y no tiene ninguna restricción asociada, por

esta razón cuando el protagonista interactúa con la antorcha la recoge directamente. Una vez recogida la antorcha si el protagonista interactúa con la cucaracha nuevamente, la puede capturar. Paralelamente la tortuga tiene como interacción hacer una pregunta que el protagonista debe responder sujeto a la restricción que el protagonista debe tener a la cucaracha capturada; entonces se producen dos situaciones: cuando el protagonista no tiene la cucaracha la tortuga presenta un diálogo alternativo y cuando el protagonista tiene la cucaracha la tortuga hace la pregunta al protagonista.

4.2. Componentes

A partir de la idea inicial se identificaron las distintas componentes básicas con las que debe contar el videojuego. A continuación se describe cada una de las componentes identificadas:

- ***Protagonista***: corresponde al personaje controlado por el usuario. En el ejemplo de la figura 4 está representado por la imagen de una persona.

- ***Escenario***: los distintos escenarios conforman en conjunto el mundo virtual. Sobre la superficie de un ***Escenario*** un ***Protagonista*** se desplaza libremente. En el ejemplo de la figura 4 se puede ver una porción (o clip) de un ***Escenario***.

- ***Gráfica***: corresponde a todos los elementos presentes en la superficie del ***Escenario*** con las que el ***Protagonista*** colisiona y a todos los elementos invisibles en la superficie del ***Escenario*** que están asociados a alguna acción del ***Protagonista*** con alguna de las ***Gráficas*** presentes en la superficie del ***Escenario***. En el ejemplo de la figura 4 podemos ver como ***Gráficas*** presentes sobre la superficie del ***Escenario*** a la antorcha, la cucaracha y la tortuga; una ***Gráfica*** invisible corresponde a elementos que se obtienen a través de otros elementos que si son visibles en la superficie del ***Escenario***. Una ***Gráfica*** puede corresponder a dos tipos:

    - ***Animación*****:** corresponde a un set de secuencias animadas, con o sin desplazamiento asociado.

    - ***Capa*****:** es una composición de 1 o más imágenes que no tienen desplazamiento.

- ***Concepto***: es lo que le da significado a la componente ***Gráfica***. Todas las componentes ***Gráficas*** tienen asociado un ***Concepto***. Los ***Conceptos*** pueden corresponder a:

    - ***Personaje***: sirve para identificar a una ***Gráfica*** como un habitante del mundo virtual. Se caracterizan por tener un nombre y una descripción (u oficio). Además un ***Personaje*** tiene asociada una ***Gráfica*** del tipo

Animación con desplazamiento, lo que constituye una característica distintiva de otros componentes inertes del juego.

- ***Ítem***: sirve para identificar a una ***Gráfica*** como un elemento que el ***Protagonista*** puede recoger en el ***Escenario*** o bien recibir de algún ***Personaje*** y utilizar en el juego con algún objetivo. Se caracterizan por tener un nombre y una descripción (o función). Además un ***Ítem*** tiene asociada una ***Gráfica*** del tipo ***Capa***.

- ***Decorativo***: sirve para identificar a todas las ***Gráficas*** que decoran el entorno virtual y con las cuales el ***Protagonista*** colisiona, restringiendo su movimiento y no permitiendo que se desplace sobre ellos (los ***Personajes*** con desplazamiento también colisionan con este tipo de elementos y no se desplazan sobre ellos). Pueden tener un nombre y una descripción solo si es necesario. Además, un ***Decorativo*** tiene asociada una ***Gráfica*** del tipo ***Capa***.

- ***Interacción***: corresponde a la acción disponible que tiene una ***Gráfica*** en un momento dado cuando el ***Protagonista*** colisiona con esta. Las ***Interacciones*** pueden ser de distintos tipos entre las cuales tenemos:

  - ***Informar:*** es una acción que está asociada a una ***Gráfica*** de cualquiera de los 3 tipos de ***Conceptos***, donde entrega información a través de un texto.

  - ***Describir:*** es una acción que está asociada a una ***Gráfica*** de cualquiera de los 3 tipos de ***Conceptos***, es similar a ***Informar***, pero se entrega información a través de un texto que identifica el ***Concepto*** y lo *describe*.

  - ***Responder:*** es una acción asociada en general a la ***Gráfica*** de un ***Personaje***, el cual presenta una ***Pregunta*** al ***Protagonista***. Esta ***Pregunta*** puede ser de 2 tipos:

    - **Alternativas:** el usuario elige la respuesta entre varias alternativas.

    - **Numérica:** el usuario ingresa la respuesta numérica.

  - **Recoger:** es una acción que está asociada a una ***Gráfica*** de cualquiera de los 3 tipos de ***Conceptos***. Como resultado se presenta un texto que informa al ***Protagonista*** el ***Concepto*** que acaba de recoger, y se añade a un bolso donde el ***Protagonista*** almacena sus ***Conceptos*** coleccionados. La ***Gráfica*** desaparece del ***Escenario*** luego de ser recogida.

- **Transar:** es una acción que está asociada en general a la ***Gráfica*** de un ***Personaje***. El ***Protagonista*** debe entregar cierta cantidad de cada uno de los ***Conceptos*** solicitados y a cambio el ***Personaje*** entrega otros ***Conceptos*** en alguna cantidad.

4.3. Relación entre los componentes del juego

A continuación se describe la forma en que se relacionan las componentes y la cardinalidad de estas relaciones:

- *Un **Protagonista** está en un **Escenario** y en un **Escenario** hay cero o un **Protagonista***: esto se debe a que el juego es de uso individual; en caso de evolucionar el juego a multijugador en un ***Escenario*** podrían haber de cero a muchos ***Protagonistas*** pero sólo uno estaría controlado por el usuario.

- *Un **Escenario** tiene una a muchas **Gráficas** y una **Gráfica** pertenece solo a un **Escenario**.*

- *Un **Concepto** puede ser representado por una a muchas **Gráficas** y una **Gráfica** tiene solo un **concepto**.*

- *Una **Gráfica** tiene una a muchas **Interacciones** y una **Interacción** pertenece solo a una **Gráfica***.

En el diagrama Entidad-Relación de la figura 5 se muestran estas relaciones.

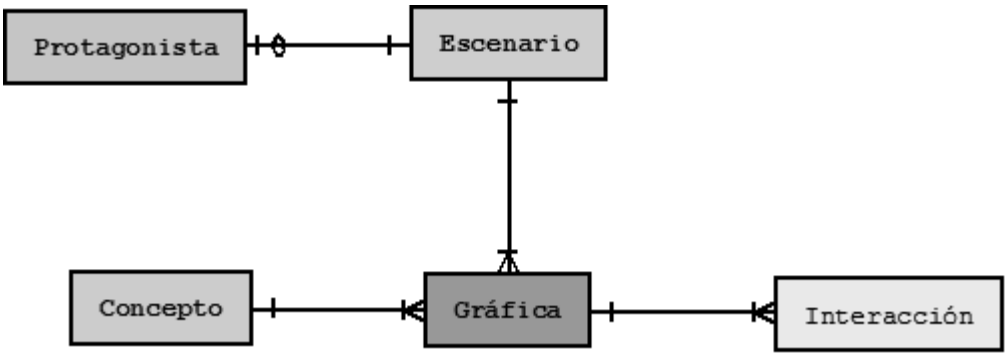


Figura 5: diagrama entidad relación de las componentes del videojuego.

4.4. Diseño de los elementos gráficos del videojuego

Como se señalo anteriormente distintos componentes del juego requieren de elementos gráficos con ciertas características acordes a la visibilidad que se quiere lograr con ellos. Por esta razón debemos definir estos elementos gráficos, cuya clasificación se muestra a continuación:

- ***Frame***: es una imagen que puede ser vista como un cuadro de 2 dimensiones de tamaño ‘x’ e ‘y’. En nuestro caso los ***Frames*** son de tamaño ‘x=16’ e ‘y=16’.

- ***Sprite***: es un conjunto de ‘n’ ***Frames*** en un orden determinado y un tamaño fijo e idéntico para todos ellos. Estos ***Frames*** se van cambiando secuencialmente cada cierto intervalo de tiempo ‘t’, desde el primero al último, en un mismo espacio de 2 dimensiones con un tamaño idéntico al de sus ***Frames***, dando la apariencia de una animación. Un ***Sprite*** tiene una ubicación en un espacio de 2 dimensiones, por esta razón al combinar la secuencia animada con el desplazamiento en este espacio se logra la apariencia de desplazamiento animado (ver figura 6).

- ***TiledLayer***: es un conjunto de ‘n’ ***Frames*** dispuestos en una matriz donde cada una de sus celdas es del tamaño de un ***Frame***. No es necesario que todas las celdas de esta matriz estén asignadas con un ***Frame***. También es posible asignar celdas de la matriz con ***Sprites*** en lugar de ***Frames***. Un ***TiledLayer*** tiene una ubicación en un espacio de 2 dimensiones (ver figura 7).

Figura 6: ejemplo de Sprite.

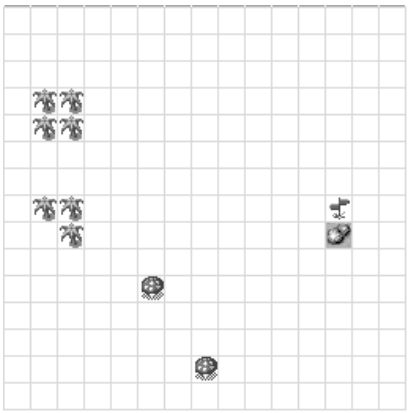


Figura 7: ejemplo de TiledLayer.

Es posible sobreponer un ***TiledLayer*** encima de otro, en este caso los espacios vacíos del ***TiledLayer*** que están encima son transparentes, lo que permite ver las celdas del ***TiledLayer*** que están abajo, ubicadas en la misma proyección de la celda superior.

También es posible sobreponer ***Sprites*** encima de un ***TiledLayer***. En este caso el espacio ocupado por el ***Sprite*** cubre su proyección sobre el ***TiledLayer***. Cuando el ***TiledLayer*** es una superficie con todas sus celdas asignadas y el ***Sprite*** tiene desplazamiento se logra el efecto de movimiento de este ***Sprite*** sobre la superficie del ***TiledLayer***.

4.5. Relación entre el diseño de las imágenes y las componentes

Como ya se definieron los elementos gráficos, ahora corresponde establecer como están relacionados con los componentes del juego. A continuación se especifica esta relación por cada una de las componentes del juego:

- ***Escenario***: un ***Escenario*** tiene asociado un ***TiledLayer*** como fondo (ver figura 8). Este ***TiledLayer*** tiene una posición fija (no tiene desplazamiento) y sobre él ubicamos a todos los demás componentes asociados al ***Escenario*** (***Protagonista*** y ***Gráficas*** correspondientes a los distintos tipos de ***Concepto***).

- ***Protagonista***: un ***Protagonista*** tiene un conjunto de ***Sprites*** asociados a sus acciones disponibles (ver figura 9). Las acciones disponibles consisten en caminar en las direcciones arriba, abajo, izquierda y derecha, con un desplazamiento que mueve al ***Protagonista*** en la dirección apuntada por la acción. En el ejemplo de la figura 2 se puede ver a la figura que representa a una persona, que puede caminar (secuencia animada + desplazamiento) por el mapa en las direcciones antes mencionadas.

- ***Gráfica*** según ***Concepto***: el tipo de elemento gráfico asociado a una ***Gráfica*** depende del ***Concepto*** que representa dicha ***Gráfica***:

    - ***Animación*** de un ***Personaje***: una ***Animación*** de un ***Personaje*** tiene un conjunto de ***Sprites*** referentes a sus acciones. Las acciones pueden ser “caminar” en las direcciones arriba, abajo, izquierda, derecha, con un desplazamiento que mueve al ***Protagonista*** en la dirección apuntada por la acción caminar o bien la acción puede ser “estar esperando” sin desplazamiento asociado.

    - ***Animación*** de un ***Ítem***: una ***Animación*** de un ***Ítem*** tiene un ***Sprite*** que por una parte representa al ***Ítem*** y además muestra una acción que quiere representar, sin desplazamientos asociados.

    - ***Capa*** de un ***Ítem***: una ***Capa*** de un ***Ítem*** tiene un ***TiledLayer*** que representa al ***Ítem***, sin desplazamientos asociados.

- o ***Capa*** de un ***Decorativo***: una ***Capa*** de un ***Decorativo*** tiene un ***TiledLayer*** que representa al ***Decorativo***, sin desplazamientos asociados.

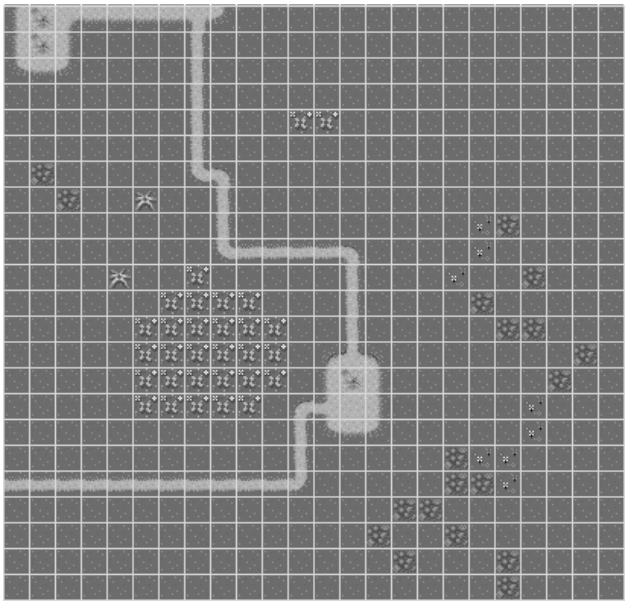


Figura 8: ejemplo de TiledLayer usado como fondo de escenario.

Figura 9: ejemplo de Sprites de un Protagonista.

4.6. Modelo de Interacciones

Anteriormente se indicaron los tipos de ***Interacción*** posibles cuando un ***Protagonista*** colisiona con una ***Gráfica***. Estos tipos de ***Interacción*** pueden ser ***Informar***, ***Describir***, ***Preguntar*** (pregunta de ***Alternativas*** o ***Numérica***), ***Recibir*** y ***Transar***. Si planteamos estas ***Interacciones*** desde el punto de vista de una hiperhistoria y de la resolución de problemas, encontramos características comunes en todos estos tipos de ***Interacción***.

En este modelo cada ***Interacción*** corresponde a un **nodo** de la hiperhistoria.

Una restricción importante añadida al modelo consiste en que cada ***Gráfica*** (sin importar el tipo de ***Concepto*** que representa) puede ejecutar a lo más una ***Interacción*** entre un conjunto de ***Interacciones*** posibles para dicha ***Gráfica,*** al momento en que el ***Protagonista*** colisiona con ella.

Si una ***Gráfica*** quiere ser considerada como un elemento interactivo debe tener una secuencia ordenada de ***Interacciones***, con al menos una ***Interacción*** posible. Esta secuencia de ***Interacciones*** tiene un orden preestablecido en cada ***Gráfica***, pero solo se ofrece una ***Interacción*** al ***Protagonista***, la que denominaremos ***Interacción Actual***.

El cambio de ***Interacción*** es secuencial, pasando a una ***Interacción Siguiente*** en el orden preestablecido solo cuando corresponda hacerlo; esto está sujeto a la restricción de que una ***Interacción Actual*** tiene solo una ***Interacción Siguiente***. Estos cambios de ***Interacción*** corresponde hacerlos solo cuando se realiza satisfactoriamente una ***Interacción***. La ***Interacción*** en una ***Gráfica*** puede ser siempre la misma (el cambio apunta hacia la misma ***Interacción***) o puede cambiar a otra ***Interacción*** de alguno de los tipos antes nombrados (el cambio apunta a otra ***Interacción***). Este cambio ocurre al terminar una ***Interacción*** y se hace visible para el usuario, ya sea por el parlamento que se presenta al término de una ***Interacción*** satisfactoria o bien al interactuar nuevamente con la misma componente ***Gráfica*** presentándose una ***Interacción*** distinta, la que corresponde a la ***Interacción Siguiente***.

Para que una ***Interacción*** pueda ser realizada se imponen ***Condiciones*** que el usuario debe cumplir. Estas ***Condiciones*** están relacionadas con los ***Conceptos*** que el ***Protagonista*** recolecta en el juego, y corresponden a:

- Tener un ***Concepto.***
- Tener una cantidad “X” del ***Concepto.***
- Tener una cantidad menor que “X” del ***Concepto.***
- Tener una cantidad menor o igual que “X” del ***Concepto.***
- Tener una cantidad mayor que “X” del ***Concepto.***
- Tener una cantidad mayor o igual que “X” del ***Concepto.***

Donde X es un número natural mayor o igual que 0.

Si las ***Condiciones*** definidas para una ***Interacción*** se cumplen, la ***Interacción*** puede continuar; en caso contrario la ***Interacción*** no puede ser realizada (en este caso se presenta un texto con información relevante cuando no se cumplen las ***Condiciones***).

Cuando se cumplen las ***Condiciones***, la ***Interacción*** continúa de distintas formas dependiendo del tipo al que corresponde:

- ***Informar***: se muestra la información y se pasa a la ***Interacción Siguiente***.

- ***Describir***: se muestra la descripción y se pasa a la ***Interacción Siguiente***.

- ***Preguntar***: se presenta al ***Protagonista*** una ***Pregunta*** de ***Alternativas*** o bien una ***Pregunta Numérica*** según corresponda. Hay que señalar que está contemplado el hecho que estas ***Preguntas*** pueden ser formuladas en base a los pasos de la resolución de problemas; en consecuencia esta ***Pregunta*** puede ser un problema a resolver o bien una porción de un problema mas grande a resolver. Luego si la respuesta que ingresa el usuario está correcta, se presenta un texto que indica que este evento fue realizado satisfactoriamente y se pasa a la ***Interacción Siguiente***, en caso contrario se presenta un texto indicando que se respondió incorrectamente y se mantiene la ***Interacción Actual***.

- ***Recoger***: se presenta un texto que informa al ***Protagonista*** el ***Concepto*** que acaba de recoger, se añade a la colección del ***Protagonista*** el ***Concepto*** que acaba de recoger y se pasa a la ***Interacción Siguiente***. Hay que señalar que esta ***Interacción*** también soporta una formulación que puede estar dentro de los pasos de la resolución de problemas; esto se debe a que esta ***Interacción*** puede pedir la entrega de ***Conceptos*** en ciertas cantidades para ser realizada satisfactoriamente. Los ***Conceptos*** que puede ***Recoger*** el ***Protagonista*** pueden ser del tipo ***Ítem***, del tipo ***Personaje*** y del tipo ***Decorativo***; es importante destacar que cuando esta ***Interacción*** ocurre satisfactoriamente, la ***Gráfica*** presente sobre el ***Escenario*** desaparece, por lo que el ***Protagonista*** no colisiona con ella nuevamente y en consecuencia no ocurren más interacciones con esta ***Gráfica***.

- ***Transar***: en forma general esta ***Interacción*** pide ***Conceptos*** de la colección del ***Protagonista*** (los que ha conseguido durante el juego) y a cambio entrega otros conceptos en algunas cantidades determinadas. Hay que señalar que esta ***Interacción*** también soporta una formulación que puede estar dentro de los pasos de la resolución de problemas. Un ***Personaje*** puede presentar un problema que debe ser resuelto a través de la entrega por parte del ***Protagonista*** de distintos ***Conceptos*** en alguna cantidad de cada uno. Si se resuelve correctamente el problema, el ***Personaje*** entrega como premio otros ***Conceptos*** en alguna cantidad, se muestra un texto relacionado a este evento y se pasa a la ***Interacción Siguiente***; en caso contrario, se muestra un texto indicando que no se resolvió el problema y se mantiene la ***Interacción Actual***.

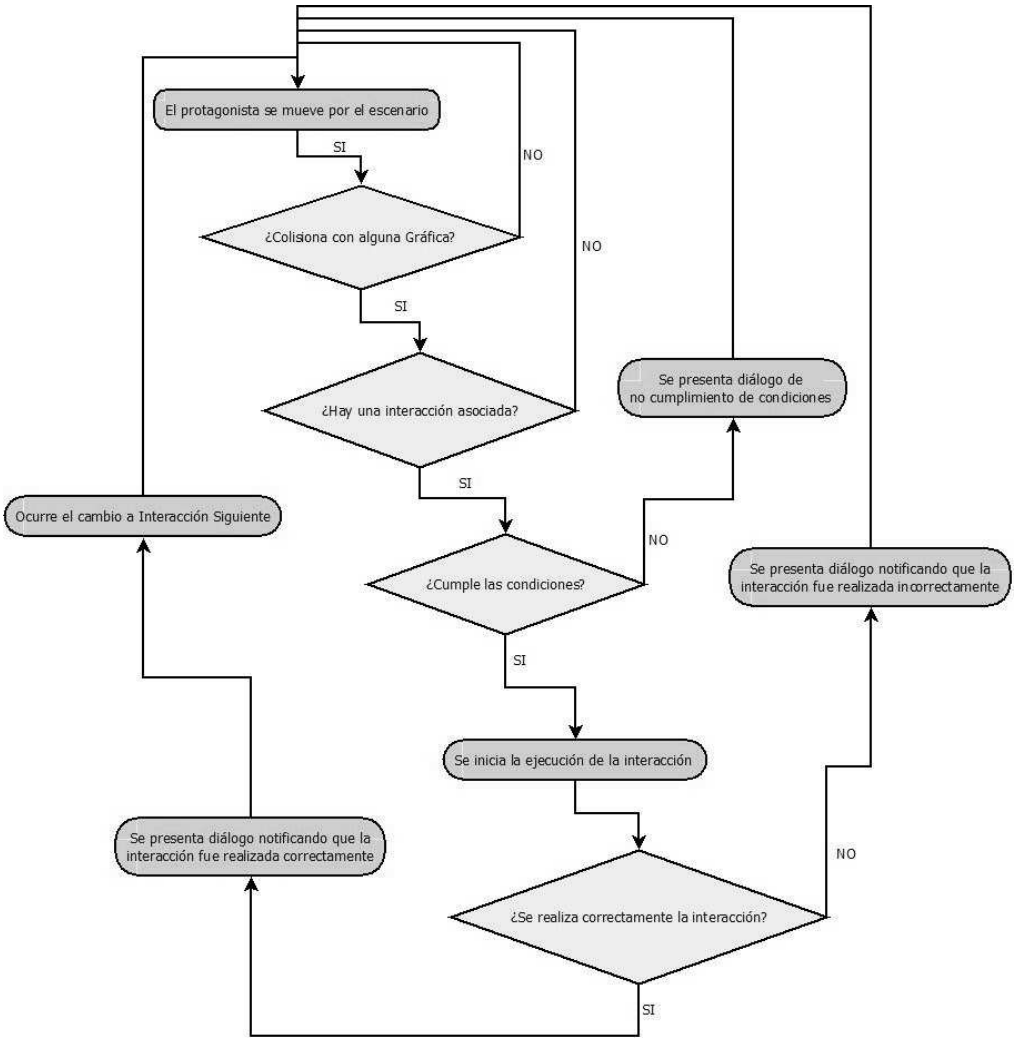


Figura 10: diagrama de flujo de una Interacción.

4.7. Integración de la resolución de problemas

Como ya pudo verse en el modelo de interacción, la resolución de problemas se hace visible directamente en las interacciones del tipo *Preguntar, Recoger* y *Transar*. Sin embargo, los otros 2 tipos interacciones (*Informar* y Describir) también participan activamente en la metodología de resolución de problemas.

Recordando lo señalado en el marco teórico en relación con los pasos de la resolución de problemas (ver capítulo 2.1.1) tenemos:

- **Comprensión e Identificación del problema.**
- **Concepción de un plan para la resolución del problema.**
- **Realización de un plan para la resolución del problema.**
- **Evaluación de la solución.**

Las ***Interacción Informar*** y ***Describir*** idealmente cumplen la función de ayudar al jugador con información relevante, pero también puede ofrecer información irrelevante aumentado la complejidad, ya que el jugador debería ser capaz de distinguir la información relevante y descartar lo irrelevante. Estas ***Interacciones*** se puede decir que son transversales en el apoyo y contribución a cada uno los pasos de la resolución de problemas.

Las ***Interacciones Preguntar***, ***Recoger*** y ***Transar*** no solamente pueden servir para plantear un problema sino también para ayudar secuencialmente a comprender un problema principal a través de la resolución de problemas más simples lo cual equivale a guiar una concepción del plan a través de la resolución de problemas más simples. Con esto también se puede ayudar a descartar información no relevante, a distinguir los datos relevantes y finalmente a idear y realizar el plan que resuelve un problema más grande. Por esta razón también se puede decir que estas ***Interacciones*** son transversales en el apoyo y contribución a cada uno de los pasos de la resolución de problemas.

Resolver problemas a través de prueba y error constituye la forma básica de resolución; ya sea en una ***Pregunta*** de ***Alternativas*** probando muchas veces hasta resolverla por descarte, o en una ***Pregunta Numérica*** ingresando valores distintos hasta calzar la solución, o bien en un problema planteado a través de las ***Interacciones Recoger*** y ***Transar*** donde el alumno puede probar con cantidades de los ***Conceptos*** recolectados en el juego hasta resolver por descarte. Si bien esto está soportado por las ***Interacciones*** en el juego, el desafío principal es lograr que el alumno resuelva el problema a través del razonamiento de la solución. Debido a esto es vital complementar con buena información la construcción de este razonamiento; en este sentido las ***Interacciones*** fueron pensadas con la intención de proveer las herramientas necesarias para ayudar al alumno en la ejecución de los pasos de la resolución de problemas, delegando responsabilidad en la forma y calidad de la información que se entrega en estas interacciones.

En resumen las ***Interacciones*** soportan muchos aspectos que apoyan y contribuyen a cada uno de los pasos de la resolución de problemas. Sin embargo, la efectividad de esta característica depende también de otros aspectos tales como la construcción de la hiperhistoria en la que se desenvuelve el ***Protagonista*** y la jugabilidad, incluyendo también la forma y calidad de la información que se incorpora en el modelo de ***Interacciones***.

4.8. Jugabilidad

Anteriormente se definieron los componentes que forman parte del juego y sus interrelaciones; además se mostró la forma en que se desarrolla el modelo de Interacciones y su relación con la resolución de problemas. Con esto claramente especificado, la construcción de una hisperhistoria utilizando los componentes definidos para el juego, sujeto al conjunto de reglas que impone el modelo de ***Interacciones*** junto

a la incorporación de la resolución de problemas, determina la jugabilidad de los juegos que pueden ser creados utilizando el motor de videojuegos educativos RPG en este trabajo.

A continuación se muestra como se estructura el juego desde el punto de vista de los objetivos y desafíos que este presenta a los usuarios:

- ***Interacciones***: son los elementos descritos anteriormente en el modelo de ***Interacciones*** y corresponden a los nodos de la hiperhistoria. Corresponde a todas las acciones que se pueden ejecutar en un momento dado cuando el ***Protagonista*** colisiona con alguna ***Grafica*** del juego.

- ***Tareas***: corresponden a objetivos específicos dentro del juego, donde el jugador a través de la resolución de problemas cortos, tiene la posibilidad de desarrollar el cumplimiento de un objetivo de mayor envergadura. Una ***Tarea*** se resuelve "idealmente" a través de la ejecución de una secuencia mínima de ***Interacciones***; se dice "idealmente" debido a que el Protagonista puede ejecutar libremente otras ***Interacciones*** que tienen relación con otras ***Tareas*** o reiterar pasos de la misma ***Tarea*** en una cantidad indeterminada de ocasiones.

- ***Misión***: corresponde al objetivo principal del juego, el cual se presenta al jugador como un problema principal que debe ser resuelto. Idealmente una ***Misión*** se cumple resolviendo una cantidad mínima de ***Tareas*** disponibles, ya que existen ***Tareas*** que pueden influir en menor grado, o bien que pueden no influir de ninguna forma, con la resolución del problema principal; esto se traduce en que no es estrictamente necesario pasar por todas las ***Tareas*** para resolver el problema principal del juego.

En la figura 11 se puede observar un esquema de la estructura del juego con estos elementos.

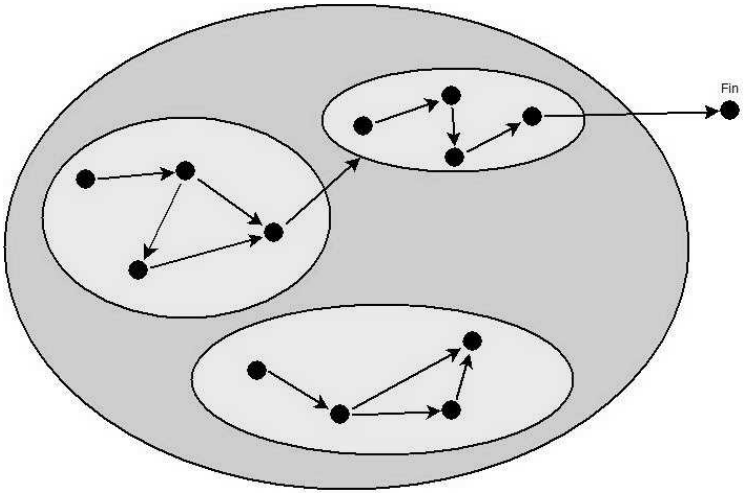


Figura 11: Esquema estructural del juego.

De esta forma la jugabilidad se construye a partir de la combinación de todos los componentes definidos para el juego, sus interrelaciones y las reglas elaboradas en el modelo de ***Interacciones***, todas dentro del marco de la estructura del juego antes descrita.

5. Implementación

A continuación se muestra como se realizó la implementación del diseño general del motor de videojuegos RPG, para llegar finalmente a los prototipos de videojuegos RPG de prueba que fueron construidos utilizando el motor, cuyas funcionalidades fueron evaluadas.

5.1. Software de desarrollo

Durante la implementación se emplearon los siguientes programas:

- JDK 6 (Java Development kit): es un software que provee herramientas de desarrollo para la creación de programas en java.

- NetBeans 6.5 [4]: NetBeans es un entorno de desarrollo para J2ME, el cual integra herramientas que facilitan el desarrollo de aplicaciones para teléfonos móviles.

- Netbeans Mobility Pack [4]: NetBeans Mobility Pack es un kit que corre en conjunto con NetBeans (como una herramienta integrada en Netbeans 6.5) y sirve para escribir, probar y hacer debug de aplicaciones java para dispositivos móviles.

- Sun Java Wireless Toolkit 2.2 [4][6]: es un conjunto de herramientas que permite crear aplicaciones que corren en dispositivos que cuentan con J2ME y MIDP. Esta se encuentra disponible como una herramienta integrada en Netbeans 6.5 una vez instalada.

- Nokia PC Suite: es un software de sincronización para teléfonos móviles de la marca Nokia. Este programa sirve para poder instalar las aplicaciones desarrolladas con Netbeans en un teléfono móvil de la marca Nokia.

- GIMP (GNU Image Manipulation Program): es un software que permite crear y manipular gráficos, incluyendo retoque de imagen y composición. Este programa fue utilizado para crear y editar muchas de las imágenes utilizadas en los prototipos funcionales.

5.2. Hardware objetivo

El universo de teléfonos móviles contempla diversos modelos, con un rango de características muy variable. Uno de los objetivos de este trabajo apuntó a la portabilidad de la solución a la mayor cantidad de dispositivos posibles, sin embargo esto debe estar sujeto a la restricción de que los dispositivos sean capaces de manejar ciertas características gráficas que permitan ejecutar un juego de este tipo. A esto se

sumó una antigüedad base de los dispositivos objetivo para este trabajo, considerando la tasa de recambio tecnológico en telefonía móvil.

Por las razones antes expuestas los teléfonos móviles que contempla este trabajo se acotan a aquellos que posean soporte para J2ME y trabajen con el perfil MIDP en su versión 2.0; esto se apoya en evidencias del porcentaje de teléfonos móviles con estas características en los usuarios de la compañía Movistar en Chile (ver gráfico de la figura 12), los cuales alcanzan el 53%.

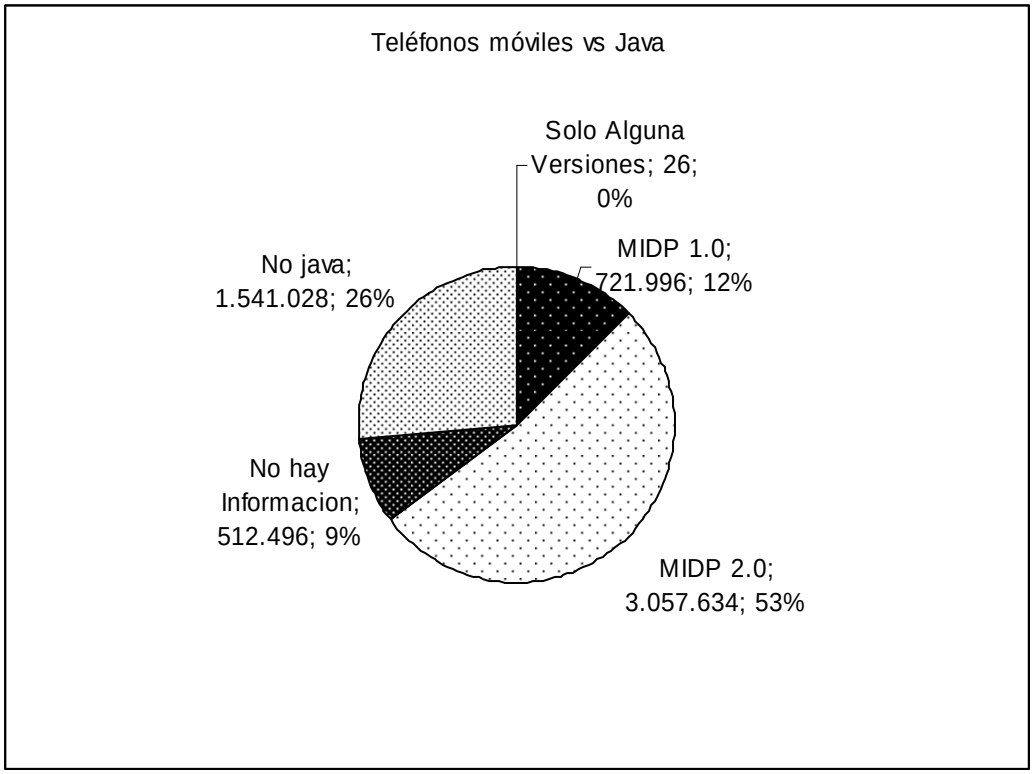


Figura 12: gráfico de la cantidad de teléfonos móviles de los usuarios de la compañía Movistar en Chile en relación a la tecnología Java que poseen a través de sus especificaciones de MIDP asociadas, fuente: Movistar, septiembre 2008, sobre un total de 5.833.180 teléfonos móviles de usuarios.

Como ya señalamos anteriormente en el capítulo 2.5, **MIDP** es un perfil asociado a **J2ME** para teléfonos móviles, y en particular la versión **MIDP 2.0** se encuentra integrada en el hardware de teléfonos móviles relativamente modernos.

Si bien existe una gran cantidad de modelos y marcas de teléfonos móviles con estas características, persisten algunas dificultades de portabilidad del software a los distintos modelos, producto de que el manejo de aspectos como el sonido, la resolución de pantalla y la interfaz del teclado, ocurren de forma diferente en algunas ocasiones para los diferentes dispositivos.

En este trabajo de memoria se trabajó con el modelo de teléfono celular Nokia 6230, el cual ofrece MIDP 2.0, una resolución de pantalla de 128x128 pixeles.

Figura 13: teléfono móvil Nokia modelo 6230.

5.3. Arquitectura de la solución

En este desarrollo se utilizó una adaptación del patrón de diseño ***Modelo Vista Controlador*** (***MVC***), ya que se ajustó perfectamente a la solución desarrollada; esto además fue impulsado por las ventajas que ofrece la estructura que define, que facilita la realización de rediseños y cambios en la implementación de la aplicación, que disminuyó el costo en tiempo para las modificaciones que fueron realizadas durante el desarrollo y que permitieron ajustar la solución de una forma optima. En el código fuente de la aplicación se crearon los paquetes ***Modelo***, ***Vista*** y ***Controlador***, y a través de las clases implementadas dentro de cada uno de estos paquetes se refleja la utilización de ***MVC***.

Otra característica implícita dentro de la implementación en este desarrollo, fue aplicar la programación orientada a objetos, capturando el diseño de la solución especificado en el capítulo 4 e implementándolo dentro del marco de la arquitectura ***MVC***.

A continuación introducimos ***MVC***, mostramos cómo fue utilizado en este desarrollo y se explica cómo fue implementada cada una de sus partes.

5.3.1. Utilización de patrón de diseño MVC

Modelo Vista Controlador (***MVC***) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, facilitando de esta forma la modificación o personalización de cada parte:

- ***Modelo***: se encarga de establecer una representación específica de la información con la cual el sistema opera. Para el desarrollo de este trabajo se definieron todos los componentes del juego que fueron introducidos en el capítulo 4.2.

- ***Vista***: se encarga de representar la especificación del modelo en un formato más adecuado para la interacción del usuario, usualmente corresponde a la interfaz de usuario. Aquí se maneja el comportamiento de los elementos gráficos de la interfaz del juego definidos en el capítulo 4.4 y cuya relación con las componentes del ***Modelo*** fue explicada en el capítulo 4.5.

- ***Controlador***: se encarga manejar los eventos generalmente asociados a las acciones del usuario, lo que genera consultas en el ***Modelo***, lo que se traduce probablemente en cambios en la ***Vista***. Estos cambios probables que se reflejan en la ***Vista*** son producto de las acciones que ejecuta usuario en el juego relacionadas con lo visto en el modelo de Interacciones explicado en el capítulo 4.6.

Aunque se pueden encontrar diferentes implementaciones de ***MVC***, en este trabajo se adaptó la variante en la que no hay comunicación directa entre el ***Modelo*** y la ***Vista***; de esta forma la ***Vista*** recibe los datos consultados al ***Modelo*** a través del ***Controlador***, sin embargo como se mostrará en el capítulo 5.3.3, se extendió la funcionalidad del ***Controlador*** a través de la incorporación de consultas a elementos que se definen en la ***Vista*** y que tienen relación con elementos del ***Modelo*** pero no pertenecen éste (estos corresponden a los elementos gráficos de la interfaz del juego definidos en el capítulo 4.4). Otra característica que fue incorporada proviene del hecho que el software trabaja en forma autónoma y por lo tanto el acceso a los datos ocurre solo de forma local; es por esta razón que en el ***Controlador*** se implementó una especie de base de datos con tablas de hash, que contiene todos los datos con los que trabaja el videojuego, los cuales se encuentran, según corresponda, como instancias de los componentes del ***Modelo*** o bien como instancias de los elementos definidos en la ***Vista***, indexados por sus identificadores (***llaves primarias***) y por las llaves que los relacionan con los otros elementos (***llaves foráneas***) en caso de tener este tipo de relación especificada.

Cabe agregar que los elementos del ***Modelo*** manejados por el ***Controlador*** son consultados y/o modificados por éste según corresponda; en cambio, cuando se consultan a través del ***Controlador*** los elementos definidos en la ***Vista***, este delega el control a la ***Vista*** para que ella realice las modificaciones necesarias. En el ***Modelo*** se encuentran los datos con los que trabaja el videojuego; cuando ocurren eventos en la ***Vista***, se consultan los datos del ***Modelo*** relevantes a dichos eventos, encontrando respuestas definidas en estos datos consultados para todas las acciones posibles que se realizan en la ***Vista***, lo que permite manejar correctamente el comportamiento del videojuego.

El flujo del patrón ***MVC*** en este trabajo es el siguiente:

- El usuario interactúa con una interfaz manejada por la ***Vista*** de la aplicación, de alguna forma.

- El ***Controlador*** recibe por parte de la interfaz de la ***Vista*** la notificación de la acción solicitada por el usuario que gatilla consultas al ***Modelo***, o bien a los elementos definidos en la ***Vista***, según sea el caso.

- El ***Controlador*** accede al ***Modelo***, consultándolo de forma adecuada a la acción gatillada por el usuario, o bien a los elementos definidos en la ***Vista***, según sea el caso.

- La ***Vista*** obtiene los datos desde el ***Controlador*** para generar la interfaz apropiada para el usuario (como se señaló anteriormente, en el caso de los elementos definidos en la ***Vista*** manejados por el ***Controlador***, este ultimo delega el control de estos elementos a la ***Vista*** para que pueda realizar modificaciones sobre estos elementos).

- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

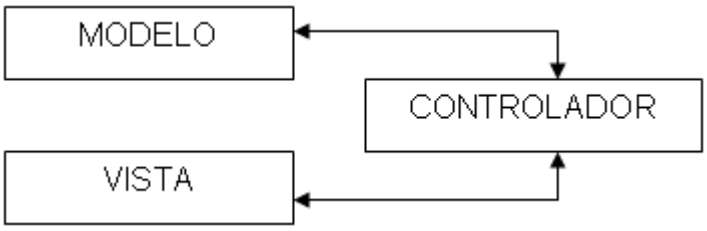


Figura 14: variante original de MVC

5.3.2. Modelo

En la aplicación se creó el paquete *Modelo* dentro del cual se agregaron todos los paquetes en los cuales se implementaron las clases correspondientes a cada una de las componentes del juego identificadas en el diseño (ver capitulo 4.2) junto con sus subcomponentes identificadas para cada una de estas componentes principales. A continuación se nombran los paquetes y sus componentes:

- ***Protagonista***: dentro de este paquete se implementa la clase ***Protagonista*** y la interfaz ***iProtagonista***.

- ***Escenario***: dentro de este paquete se implementan la clase ***Escenario*** y la interfaz ***iEscenario***.

- ***Gráfica***: dentro de este paquete se implementan la clase abstracta ***Gráfica***, la interfaz ***iGráfica*** y las clases ***Animación*** y ***Capa*** que heredan de la clase ***Gráfica*** e implementan la interfaz ***iGráfica***.

- ***Concepto***: dentro de este paquete se implementan la clase abstracta ***Concepto***, la interfaz ***iConcepto*** y las clases ***Personaje***, ***Ítem*** y ***Decorativo*** que heredan de la clase ***Concepto*** e implementan la interfaz ***iConcepto***.

- ***Interacción***: dentro de este paquete se implementan la clase abstracta ***Interacción***, la interfaz ***iInteracción*** y las clases ***Informar***, ***Describir***, ***Responder***, ***Recoger*** y ***Transar*** que heredan de la clase ***Interacción*** e implementan la interfaz ***Interacción***.

Para el uso que se le asignó a los componentes del ***Modelo***, resultó conveniente que varias clases que heredan de una clase abstracta implementen la misma interfaz ya que se simula que estos objetos son del tipo de la interfaz, centralizando el hecho de que varios objetos sean consultados para un mismo objetivo, pero permitiendo también saber a qué subtipo en específico pertenecen, para poder realizar modificaciones adecuadas en la ***Vista***.

En las componentes del ***Modelo*** se agregaron atributos identificadores que permiten implementar lo que se especificó en el diseño como relación entre las componentes del juego (ver capitulo 4.3). Cada componente tiene un identificador o ***llave primaria*** y si corresponde según las relaciones asociadas que tenga, posee ***llaves foráneas*** que son referencias a los identificadores de otras componentes.

La aplicación trabaja con un ***Protagonista*** y un ***Escenario*** activo, y si bien la relación entre ellos es 1 a 1, no es necesario incluir llaves que especifiquen su relación puesto que se accede directamente a ellos a través de sus identificadores. Algo parecido ocurre con el ***Concepto*** ya que también es accedido directamente a través de su identificador.

En la ***Grafica*** es distinto, porque en este componente se consulta frecuentemente por su ***Interacción Actual*** y se encuentra relacionado a un ***Escenario*** y a un ***Concepto***. Como se puede observar en el modelo relacional entre componentes (ver capítulo 4.3), un ***Escenario*** se compone de varias ***Graficas*** y un ***Concepto*** puede estar asociado a distintas ***Graficas***. Por estas razones, se agregaron una llave que referencia al identificador del ***Escenario*** al que pertenece, una segunda llave que referencia al identificador del ***Concepto*** que representa y además, una tercera llave que referencia al identificador de la ***Interacción Actual*** que tiene disponible para ejecutar, introducido en el modelo de interacciones explicado en el capítulo 4.6.

En la ***Interacción*** se agregó una llave que referencia al identificador de la ***Grafica***, ya que una ***Grafica*** tiene varias interacciones. De esta forma se puede acceder al conjunto de ***Interacciones*** posibles para cada ***Grafica*** y para consultar por la

Interacción Actual de una ***Grafica*** específica, se busca con la llave que referencia al identificador de la ***Interacción Actual*** sobre este conjunto de ***Interacciones*** posibles. Con esto se implementó lo necesario para ejecutar la especificación del modelo Interacciones visto en el capítulo 4.6.

Con estas relaciones especificadas se puede ver lo fácil que resulta hacer consultas como la *Interacción Actual* o el *Concepto*, a partir de una *Gráfica* en un *Escenario* específico.

Este conjunto de especificaciones implementadas tiene la misma estructura relacional de una base de datos.

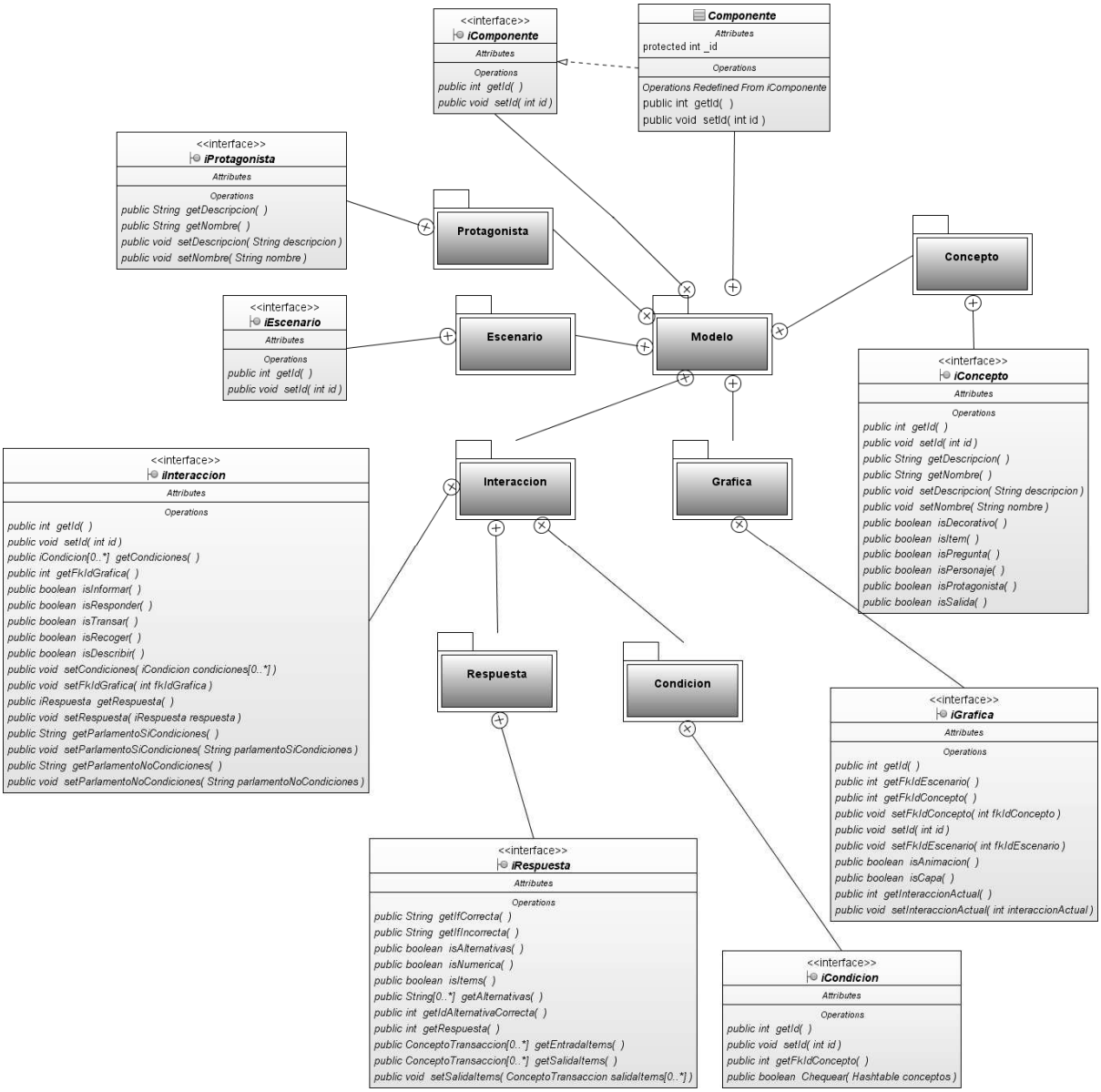


Figura 15: diagrama con los paquetes e interfaces dentro del paquete Modelo

5.3.3. Controlador

En el paquete ***Controlador*** se implementaron todas las consultas sobre los datos, que serán utilizadas por la vista, agrupándolas por clases que manejan los conjuntos de componentes con las mismas características en el juego. De esta forma se crearon ***clases controladoras*** que son relativas a los componentes ***Escenario***, ***Grafica***, ***Concepto*** e ***Interacción*** del ***Modelo***, la que se llamaron respectivamente ***Escenarios***, ***Graficas***, ***Conceptos*** e ***Interacciones***. Aquí se agregaron también **clases controladoras** que manejan los conjuntos de componentes que modelan elementos definidos en la ***Vista*** los que fueron llamados ***SpritesProtagonistas***, ***SpritesPersonajes***, ***TiledLayersBases*** y ***TiledLayersElementos***.

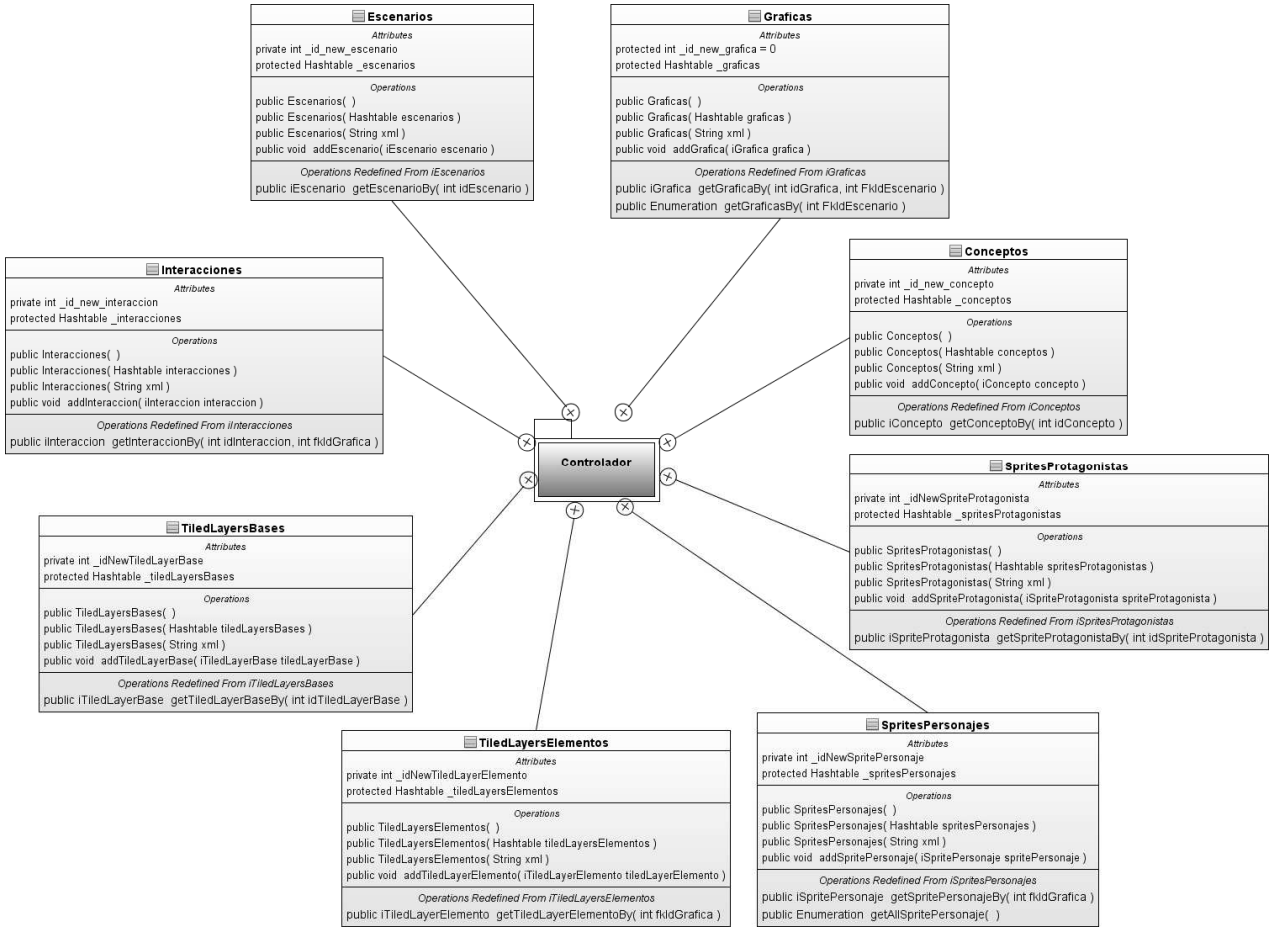


Figura 16: diagrama con las clases del paquete Controlador.

Como el software no accede a ninguna base de datos para crear, consultar y editar los componentes especificados en el ***Modelo*** del juego y en los elementos definidos en la ***Vista***, se construyó una especie de base de datos con tablas de hash en cada una de las ***clases controladoras***.

Cada tabla de hash dentro de una ***clase controladora*** puede ser vista como una tabla de la base de datos, donde se mantiene una colección de instancias del tipo de

componente manejada por la ***clase controladora***. Estas tablas de hash están indexadas con las ***llaves primarias*** de cada tipo de componente y si corresponde, con las ***llaves foráneas*** en caso de que las tenga.

5.3.4. Vista

En el paquete ***Vista*** tenemos la aplicación principal que se ejecuta en el teléfono móvil, acompañado de los paquetes ***Diseño***, ***Elemento***, ***Canvas*** y ***Lógica***.

En el paquete ***Diseño*** se juntan todos los diseños generados con la herramienta de edición de elementos gráficos del juego, integrada en Netbeans, donde cada archivo corresponde al diseño gráfico de un juego. En cada diseño se definen los ***Sprites***, ***Tiledlayers*** y su distribución en los distintos ***Escenarios***.

En el paquete ***Elemento*** se definieron clases que modelan distintos elementos gráficos los que se encapsulan con identificadores para poder ser indexados al igual que los componentes modelados en el paquete ***Modelo***. Estas clases corresponden a:

- ***SpriteProtagonista***: encapsula la construcción de los ***Sprites*** de un ***Protagonista*** y tiene un identificador que permite indexación en la tabla de hash manejada por la ***clase controladora SpritesProtagonistas***.

- ***SpritePersonaje***: encapsula la construcción de los ***Sprites*** de un ***Personaje*** y tiene como llaves dos identificadores, uno equivalente al identificador de la ***Gráfica*** y otro equivalente al identificador del ***Escenario*** sobre el cual esta ubicada la grafica, lo que permite la indexación en la tabla de hash manejada por la ***clase controladora SpritesPersonajes***.

- ***TiledLayerBase***: encapsula la construcción de un ***TiledLayer*** correspondiente a la superficie caminable o no colisionable de un ***Escenario*** específico y tiene un identificador equivalente al identificador de este mismo ***Escenario***, lo que permite la indexación en la tabla de hash manejada por la ***clase controladora TiledLayersBases***.

- ***TiledLayerElemento***: encapsula la construcción de un ***TiledLayer*** de un elemento en la superficie de un ***Escenario*** y tiene como llaves dos identificadores, uno equivalente al identificador de la ***Gráfica*** y otro equivalente al identificador del ***Escenario*** sobre el cual esta ubicada la ***Gráfica***, lo que permite la indexación en la tabla de hash manejada por la ***clase controladora TiledLayersElementos***.

Además dentro del paquete ***Elemento*** se definieron otras clases que son utilizadas por las clases que componen este paquete, y que se encargan del manejo de la animación y el movimiento de estos elementos definidos cuando se requiere.

En el paquete ***Lógica*** se definen controladores de mayor nivel para el juego, donde a partir de una clase base que contiene un set de funciones básicas de control de juego se pueden generar nuevas clases que hereden de esta clase base y que reescriban las funciones o que agreguen nuevas funciones (en el anexo 9.5 se pueden ver las funciones disponibles en la interfaz *iLogica*). En las clases que heredan de la clase Lógica se crea una función que se encarga de poblar las tablas de hash manejadas en las ***clases controladoras*** con datos relacionalmente correctos para las tablas de los componentes del ***Modelo*** y los elementos propios de la ***Vista***, según corresponda. La idea fue crear para cada prototipo una clase que hereda de la clase ***Lógica***, donde para cada prototipo se agruparon la creación de sus propios datos para el ***Modelo*** y elementos de la ***Vista***, permitiendo además modificar las funciones base implementadas en la clase ***Lógica*** o bien añadiendo nuevas funciones, que pueden ser utilizadas en la ***Vista***. En los anexos 9.6 y 9.7 se pueden ver las funciones "setDatos()" correspondientes a los prototipo 1 y 2 implementadas en las clases ***LogicaPrototipo1*** y ***LogicaPrototipo2*** respectivamente las cuales heredan de la clase ***Lógica***.

Finalmente en el paquete ***Canvas*** se crearon clases que manejan el comportamiento del dibujo a bajo nivel sobre la pantalla (elemento que en ***J2ME*** se denomina canvas). Aquí se crearon clases que heredan del canvas base y donde progresivamente se fueron agregando nuevas características. Finalmente se obtuvo el siguiente set de funcionalidades:

- Manejar la captura de los botones presionados en el teléfono móvil. Ya sea para controlar el movimiento del ***Protagonista*** o bien otras funcionalidades que se relacionan con los botones de control establecidos para el dispositivo.

- Control sobre las Interacciones que se gatillan a partir de las colisiones del ***Protagonista*** con las ***Gráficas*** sobre el ***Escenario***.

- Mostrar texto en pantalla, con scroll arriba y abajo cuando corresponde.

- Mostrar ***Preguntas con Alternativas*** en pantalla, donde se muestra un texto con el enunciado y el usuario puede seleccionar la alternativa correcta desde un conjunto de opciones, donde además estas alternativas texto con scroll cuando corresponde.

- Mostrar ***Preguntas Numéricas***, donde se muestra un texto con el enunciado y el usuario puede ingresar una respuesta numérica.

- Mostrar las ***Interacciones Recoger*** y ***Transar*** de una forma análoga a la ***Pregunta con Alternativas***, cuando estas ***Interacciones*** están sujetas a entregar un ***Concepto*** que está en la colección de ***Conceptos*** obtenidos en el juego por el ***Protagonista***.

- Utilización en general de funciones implementadas en las clases del paquete ***Lógica***, las cuales fueron implementadas para facilitar el manejo en el canvas del videojuego.

En el paquete ***Canvas*** la idea también fue reutilizar las implementaciones para un nuevo juego específico, donde podemos crear una nueva clase que herede por ejemplo de la clase ***SceneCanvasPrototipo2*** (esta clase corresponde al **canvas** utilizado por el Prototipo 2), permitiendo de esta forma reutilizar los elementos controlados por ese **canvas** y agregar nuevas características o bien modificar las existentes.

5.4. Interfaces del usuario

5.4.1. Visualización de los componentes en pantalla

El tamaño (en pixeles con dimensiones x e y) de los objetos gráficos en la pantalla es fijo y no se ejecutan ajustes proporcionales cuando los tamaños de la pantalla son más pequeños o más grandes.

En la pantalla se muestra una porción del ***Escenario*** completo, la que denominaremos **clip**. El tamaño de este **clip** depende del dispositivo donde se está ejecutando la aplicación, ya que el software cuando está en ejecución es capaz de rescatar los valores correspondientes a las dimensiones X e Y del display de la pantalla y con estos valores puede asignar las mismas dimensiones al **clip** y en consecuencia poder moverlo junto con el ***Protagonista*** sin salir de los márgenes del ***Escenario***.

La diferencia entre la visibilidad del ***Escenario*** de juego de un dispositivo con una pantalla más grande con respecto a uno con una pantalla más pequeña, ocurre solo en el **clip** que el dispositivo es capaz de mostrar y la ventaja que tiene un dispositivo con una pantalla más grande es la mejor panorámica del ***Escenario*** del juego, sin embargo los objetos los ve del mismo tamaño que un dispositivo con la pantalla mas pequeña. En la figura 17 se ilustran distintas panorámicas para un teléfono móvil con una resolución de pantalla de 128x128 pixeles el usuario solo ve el área 1, para un teléfono con una resolución de pantalla de 128x160 el usuario ve las áreas 1 y 2 y finalmente para un teléfono con una resolución de pantalla de 240x320 el usuario ve las áreas 1, 2 y 3.

Las ***Interacciones*** son los componentes del juego que se muestran a través de textos, los cuales tienen un tamaño fijo de letra y también están restringidos para no sobrepasarse del tamaño del display de la pantalla. Como la idea no es perder la noción del posicionamiento del ***Protagonista*** en el ***Escenario*** cuando ocurre una ***Interacción***, se optó por dejar a lo más 3 líneas de texto con un fondo blanco que cubre la parte inferior de la pantalla, la que llamaremos ***área de diálogos***. En cada línea se muestran secuencias de palabras completas que calzan dentro del tamaño horizontal del ***área de diálogos***, continuando en la siguiente línea cuando la palabra queda fuera de la línea anterior y así sucesivamente hasta completar las 3 líneas o acabar antes las palabras disponibles para mostrar. Cuando el texto a mostrar sobrepasa el ***área de diálogos***, se

ofrece un **scroll** de texto, el que desplaza las líneas mostradas hacia arriba o hacia abajo de 1 en 1, permitiendo mostrar más líneas sin perder el hilo de la lectura.

De esta forma los dispositivos con pantallas más grandes son capaces de mostrar más palabras por línea y en consecuencia más texto en el ***área de diálogos***, sin necesidad de requerir tantas veces el ***scroll*** cuando se trata de textos muy largos.

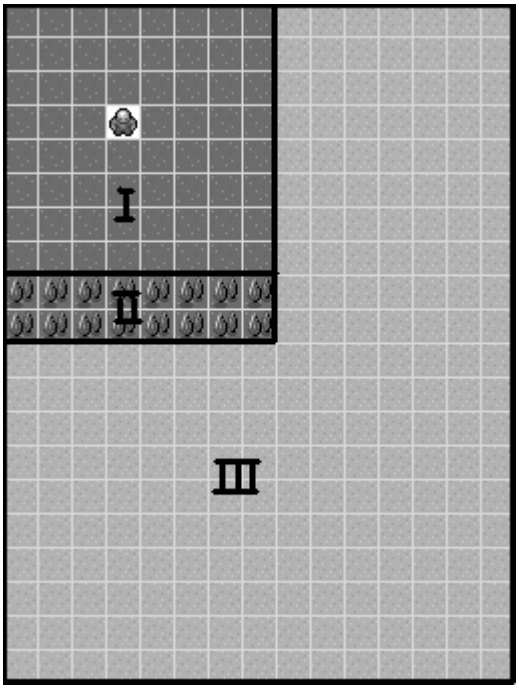


Figura 17: clip de un escenario para distintas resoluciones.

5.4.2. Botones de entrada

Siempre es preferible tratar de mantener lo más simple posible el control de las acciones de un juego. En este sentido se optó por usar solo los botones del joystick para las distintas funciones disponibles como entrada para el juego.

Los botones del joystick del teléfono móvil los identificamos como: botón **arriba**, botón **abajo**, botón **izquierda**, botón **derecha** y botón **central** (o **fire**) (ver figura 18).

Distintas funciones del juego utilizan distintos botones del joystick:

- Para el movimiento del ***Protagonista***: botones **arriba**, **abajo**, **izquierda** y **derecha**.
- Para continuar luego de ***Interacciones Informar*** y ***Describir***: botón **fire**.
- Para seleccionar una de las alternativas disponibles en las ***Interacciones Recoger***, ***Transar*** y ***Responder***: botones **izquierda** y **derecha**.

- Para confirmar una de las alternativas disponibles en las ***Interacciones Recoger***, ***Transar*** y ***Responder***: botón **fire**.

- Para realizar ***scroll*** de texto en cualquiera de las interacciones: botones **arriba** y **abajo**.

Figura 18: botones del joystick de un teléfono móvil.

5.4.3. Comportamiento de las componentes

En la pantalla de juego hay una superficie caminable del ***Escenario***, donde el ***Protagonista*** se puede desplazar libremente, restringida por los ***Personajes***, ***Ítems*** y ***Decorativos***, con los cuales el ***Protagonista*** colisiona. Si estos elementos colisionables tienen una ***Interacción Actual*** disponible y el ***Protagonista*** colisiona con ellos, se producen apariciones de texto en un ***área de diálogo***s de la pantalla relacionadas con la ***Interacción***. Aquí se suspende el control del usuario sobre los movimientos del ***Protagonista*** y el control pasa a estar en el ***área de dialogo*** con las características nombradas anteriormente en el capítulo 5.4.2.

Cuando se sale de una ***Interacción*** (ya sea tras haberla cumplido satisfactoriamente o no, o bien cuando esta es interrumpida por el no cumplimiento de las condiciones) se presiona el botón **fire** retomando el control de los movimientos del ***Protagonista***.

5.5. Construcción del Prototipo 1

Para la construcción del primer prototipo en un principio se definieron el ***Protagonista***, los ***Personajes***, los ***Ítems*** y los **Decorativos** de la hiperhistoria, además de la base del ***Escenario*** (o superficie no colisionable). Para cada uno de ellos se asoció una representación gráfica:

- ***Protagonista***:
  - **Niño**: tiene ***Sprites*** de un niño con movimiento en las 4 direcciones posibles.

- ***Personajes***:
  - **Cucaracha**: tiene ***Sprites*** de una cucaracha con movimientos en las direcciones izquierda y derecha
  - **Tortuga**: tiene ***Sprites*** de una tortuga con movimientos en las direcciones izquierda y derecha

- ***Ítem***:
  - **Antorcha**: tiene un ***Sprite*** de una antorcha sin desplazamiento con la animación de la llama.

- ***Decorativos***:
  - **Árboles** y **Casa**, **Arbustos**, **Palmeras**, etc.: ***Tiledlayer*** compuesto por frames con todos esos elementos.

- Base ***Escenario***:
  - **Pasto** y **flores**: ***Tiledlayer*** compuesto por frames con estos elementos.

En este prototipo se utilizaron dos imágenes que venían incluidas en un código de un juego de ejemplo en Netbeans. Estas imágenes están compuestas por un conjunto de frames con elementos para crear un ***Protagonista***, ***Personajes***, ***Ítems*** y ***Decorativos***, además de otros frames para construir la base del ***Escenario*** sobre el cual se ubican los elementos antes nombrados. En este mismo código de ejemplo hay una clase con funciones que acceden a estos elementos gráficos creados con sus características particulares a partir de la herramienta de edición de elementos gráficos de Netbeans, que es la encargada de autogenerar esta clase. Dicha clase autogenerada se agregó al paquete de ***Diseño*** que está en la ***Vista*** de la aplicación y se reutilizó tal cual como base para agregar nuevos elementos que no estaban creados en dicho diseño y reordenando a gusto los elementos en el ***Escenario***. En la figura 19 se puede ver las dos imágenes con las cuales se construyen las graficas del juego y en la figura 20 se puede ver la disposición del ***Escenario*** completo.

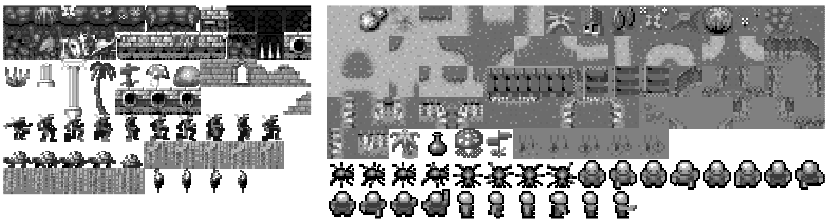


Figura 19: imágenes con las que se construyen los sprites y los tiledlayers.

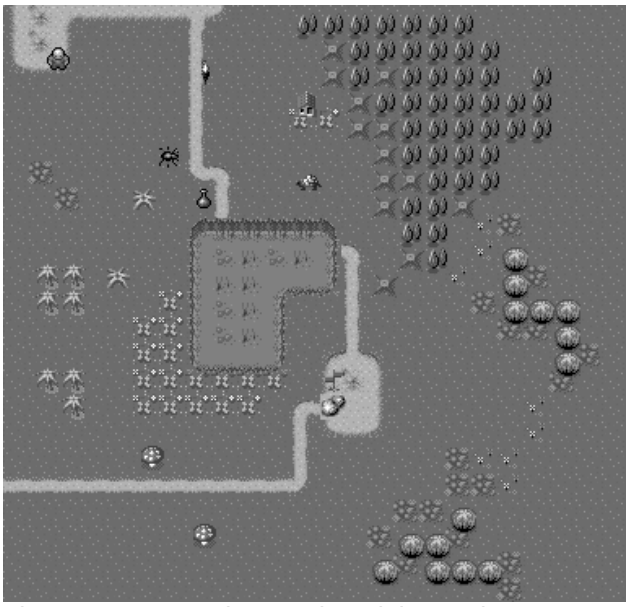


Figura 20: escenario completo del prototipo 1.

Para este primer prototipo se dispuso un escenario donde se construyó una historia simple la que consistió en resolver la tarea de capturar la **Cucaracha**. Para capturar a la **Cucaracha** había que obtener previamente la **Antorcha**. La **Antorcha** podía ser obtenida sin requisitos previos. Y por otra parte la **Tortuga** hace una pregunta cuando el ***Protagonista*** tiene en su poder a la **Cucaracha**.

Cuando las condiciones de las ***Interacciones*** no se cumplen, se notificó con un texto en el ***área de diálogos*** un mensaje definido para esto. También se observó como se desarrollan las ***Interacciones*** cuando cumplen las condiciones y se generaron salidas en el ***área de diálogos*** para notificar la realización satisfactoria o insatisfactoria de dichas ***Interacciones***.

En la figura 21 se muestra la secuencia ideal de resolución del problema. Como la **Cucaracha** tiene como restricción para ser recogida (o capturada) el hecho que el ***protagonista*** debe tener la **Antorcha**, se puede observar que cuando el ***Protagonista*** no tiene la **Antorcha** e interactúa con la **Cucaracha** esta no permite ser capturada y se despliega un texto en el ***área de diálogos***: "*Cucaracha: Lero Lero no me atrapas*". También se puede observar que el ***Protagonista*** al interactuar con la **Antorcha**, la recoge directamente y se presenta un texto en el ***área de diálogos***: "*Antorcha: Has conseguido una antorcha de fuego*"; la **Antorcha** se puede recoger directamente debido

a que no tiene ninguna restricción asociada. Finalmente, el ***Protagonista*** cuando vuelve a interactuar con la **Cucaracha**, ya con la **Antorcha** en su poder, la logra capturar y se presenta el texto rn el ***área de diálogos***: "*Cucaracha: Ay, me quemaste y me has capturado*".

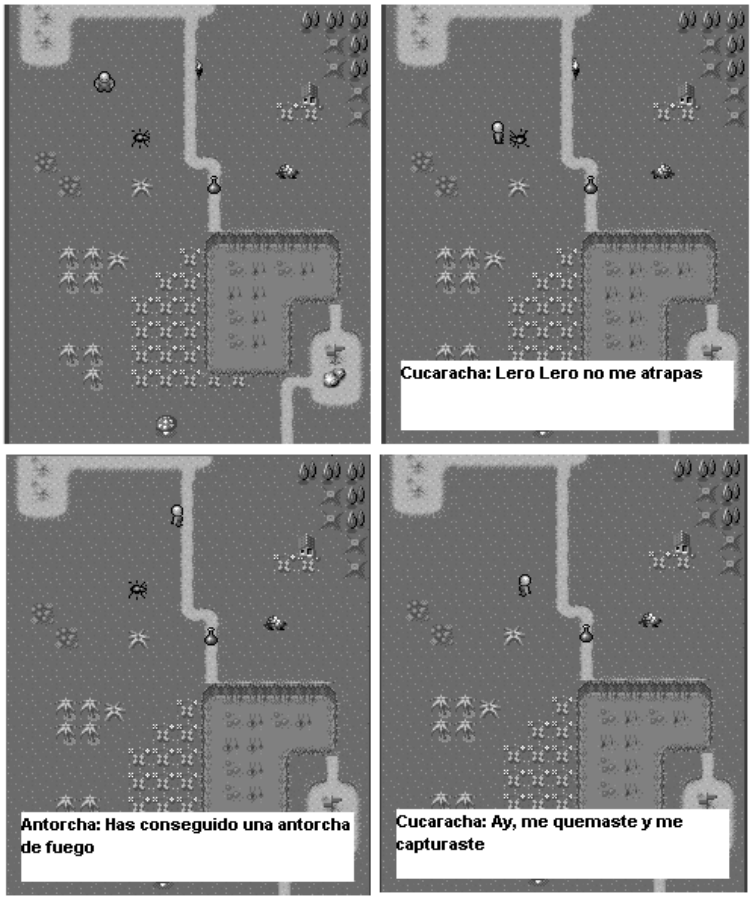


Figura 21: solución del problema propuesto en el prototipo 1.

5.6. Construcción del Prototipo 2

Para la construcción del segundo prototipo en un principio se definieron el ***Protagonista***, los ***Personajes***, los ***Ítems*** y los ***Decorativos*** de la hiperhistoria, además de la base del ***Escenario*** (o superficie no colisionable). Para cada uno de ellos se asoció una representación gráfica:

- ***Protagonista***
  - Niño: tiene ***Sprites*** de un niño en las 4 direcciones posibles de movimiento.

- ***Personajes***
  - Panadero: tiene un ***Sprite*** de un panadero detenido moviendo el cuerpo.

- Carnicero: tiene un ***Sprite*** de un carnicero detenido moviendo el cuerpo.
- Piloto: tiene ***Sprites*** de un niño que se mueve en las direcciones izquierda y derecha
- Pescador: tiene un ***Sprite*** de un pescador detenido moviendo su caña de pescar.
- Apicultor: tiene un ***Sprite*** de un hombre detenido moviendo el cuerpo.
- Niña del maíz: tiene ***Sprites*** de una niña que se mueve en las direcciones izquierda y derecha.
- Niña de la lechuga: tiene ***Sprites*** de una niña que se mueve en las direcciones izquierda y derecha.
- Niña del negocio: tiene ***Sprites*** de una niña que se mueve en las direcciones izquierda y derecha.
- Niño de las manzanas: tiene ***Sprites*** de un niño que se mueve en las direcciones izquierda y derecha.
- Niño del molino: tiene ***Sprites*** de un niño que se mueve en las direcciones izquierda y derecha.
- Agricultora: tiene un ***Sprite*** de una agricultora detenido moviendo su herramienta de trabajo.
- Perro: tiene ***Sprites*** de un perro que se mueve en las direcciones izquierda y derecha.
- Gallina: tiene ***Sprites*** de una gallina que se mueve en las direcciones izquierda y derecha.

- ***Ítems*** (cada uno esta formado por un ***TiledLayer*** compuesto por un ***Frame*** que representa gráficamente el significado de los ***Ítems***)
  - Carne
  - Maíz
  - Harina
  - Huevo
  - Lechuga
  - Trigo
  - Manzana
  - Miel
  - Pan
  - Pescado

- ***Decorativos*** (cada uno esta formado por un ***TiledLayer*** compuesto por varios ***Frames***)
  - Agua
  - Casa
  - Panadería
  - Carnicería
  - Avioneta
  - Bote de pesca
  - Molino

- Árboles con colmenas de abejas
- Árboles con manzanas
- Árboles
- Campo de trigo
- Campo de lechuga
- Campo de maíz

- Base ***Escenario***
  - Pasto, caminos de tierra y flores: esta formado por un ***Tiledlayer*** que se compone de ***Frames*** que representan estos elementos.

Con la herramienta-editor de diseño de juegos integrada en Netbeans se crearon estos elementos gráficos a partir del mismo set de imágenes del primer prototipo más 3 nuevas imágenes compuestas por un set de ***Personajes***, un set de ***Ítems*** y un set de ***Decorativos*** (ver figura 22) confeccionados para este prototipo. Una vez creados los elementos, se ordenaron en el ***Escenario*** tal como se ve en la vista panorámica del ***Escenario*** completo en la figura 23. La herramienta-editor generó una clase de diseño con funciones que obtienen los elementos diseñados que fue agregada al paquete ***Diseño*** en la ***Vista***.

Figura 22: nuevas imágenes incorporadas con las que se construyen otros Sprites y TiledLayers.

El juego de este prototipo tiene como misión salir de la isla y para esto debe lograr que el piloto pueda subir a la avioneta.

El piloto no puede subir a la avioneta porque hay un perro que no lo deja acercarse. Por lo tanto la ***Interacción*** del piloto tiene como condición tener capturado al perro, y con esto el juego se finaliza. Mientras el perro no sea capturado por el ***Protagonista***, el piloto presenta el diálogo que no puede acercarse a su avioneta.

Por otra parte el perro quiere que lo alimenten con un alimento rico en proteínas. En el juego se pueden obtener 3 tipos de alimentos ricos en proteínas: el huevo, el pescado y la carne; pero es la carne lo que satisface al perro y con esto es capturado.

En resumen los Personajes entregan un Ítem a cambio de otro Ítem disfrazando el intercambio en una necesidad planteada como problema, entregando pistas de cual es la respuesta pero nunca pidiendo el Ítem por su nombre. Análogamente a la forma en que el perro pide su alimento (algo rico en proteínas), los demás Personajes piden Ítems planteando una problemática consistente en la necesidad del Personaje por obtener algo con ciertas características nutricionales.

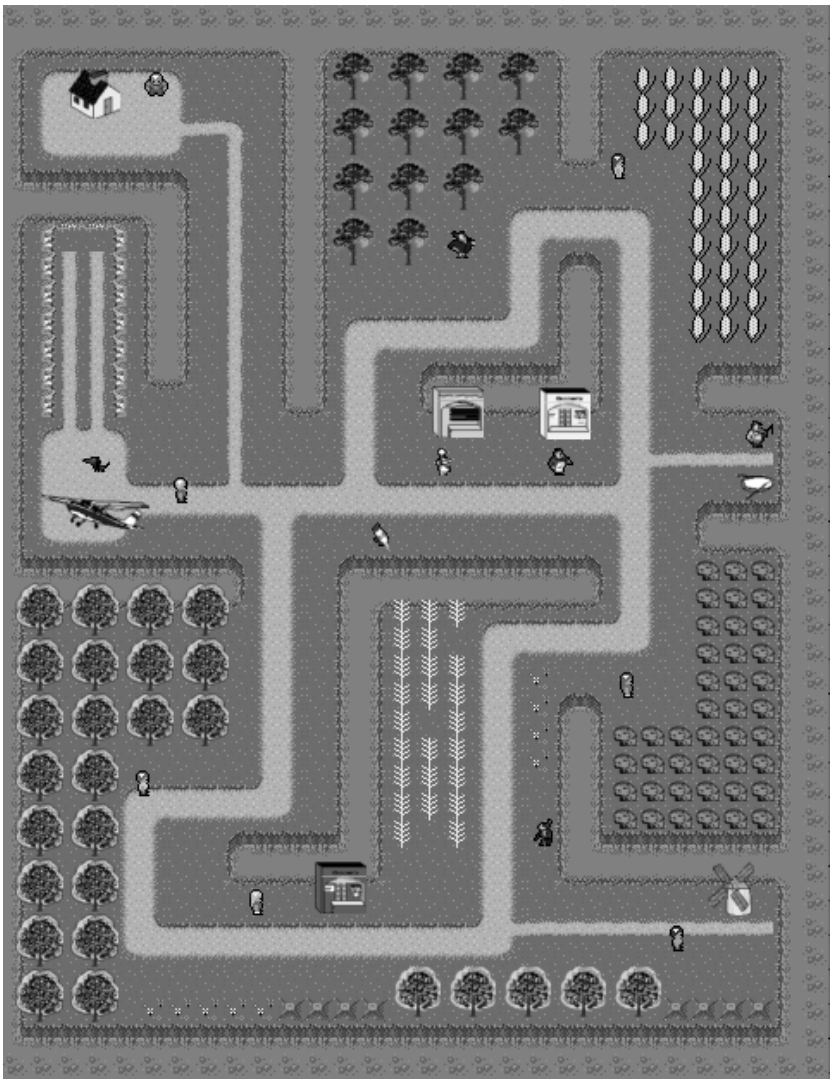


Figura 23: escenario completo del prototipo 2.

En la siguiente tabla se muestra la forma en que los personajes plantean los problemas:

	Plantea problema pidiendo
Panadero	Ingrediente base para hacer pan
Carnicero	Alimento a base de cereales y rico en carbohidratos
Piloto	Liberar el paso de la avioneta bloqueado por el perro
Pescador	Almuerzo vegetariano rico en vitaminas
Apicultor	-
Niña del maíz	-
Niña de la lechuga	Alimento concentradamente dulce
Niña del negocio	Su gallina perdida
Niño de las manzanas	-
Niño del molino	Materia prima para fabricar harina
Agricultora	Alimento dulce y rico en vitaminas
Perro	Alimento rico en proteínas
Gallina	Alimento del tipo cereal

Tabla 2: personajes y problemas planteados

En la siguiente tabla se muestran las secuencias de interacciones disponibles por cada usuario:

Personaje	Interacción 1			Interacción 2			Interacción 3 ->		
	Tipo	Recibe	Entrega	Tipo	Recibe	Entrega	Tipo	Recibe	Entrega
Panadero	Describir	-	-	Transar	Harina	Pan	Informar	-	-
Carnicero	Describir	-	-	Transar	Pan	Carne	Informar	-	-
Piloto	Describir	-	-	Transar	Perro	FIN	Informar	-	-
Pescador	Describir	-	-	Transar	Lechuga	Pescado	Informar	-	-
Apicultor	Describir	-	-	Transar	-	Miel	Informar	-	-
Niña del maíz	Describir	-	-	Transar	-	Maíz	Informar	-	-
Niña de la lechuga	Describir	-	-	Transar	Miel	Lechuga	Informar	-	-
Niña del negocio	Describir	-	-	Transar	Gallina	Huevo	Informar	-	-
Niño de las manzanas	Describir	-	-	Transar	-	Manzana	Informar	-	-
Niño del molino	Describir	-	-	Transar	Trigo	Harina	Informar	-	-
Agricultora	Describir	-	-	Transar	Manzana	Trigo	Informar	-	-
Perro	Describir	-	-	Recoger	Carne	Perro	Informar	-	-
Gallina	Describir	-	-	Recoger	Maíz	Gallina	Informar	-	-

Tabla 3: Resumen de las interacciones disponibles por personajes, lo que quieren recibir los personajes y lo que entregan los personajes.

Cada ***Personaje*** parte con la ***Interacción 1*** como su ***interacción Actual***. A medida que se cumplen las condiciones y se realiza satisfactoriamente una ***Interacción*** ocurre el cambio a la ***Interacción Siguiente***. De esta forma se pasa desde la ***Interacción 1*** a la ***Interacción 2***, de la ***Interacción 2*** a la ***Interacción 3*** y finalmente en la ***Interacción 3*** el cambio va a apuntar siempre a la misma ***Interacción 3***.

En la figura 24 se muestran los posibles flujos a seguir y el flujo que soluciona el problema. Hay que indicar que el Protagonista tiene libertad para ejecutar las acciones en el escenario y estos flujos se pueden dar de forma entrelazada.

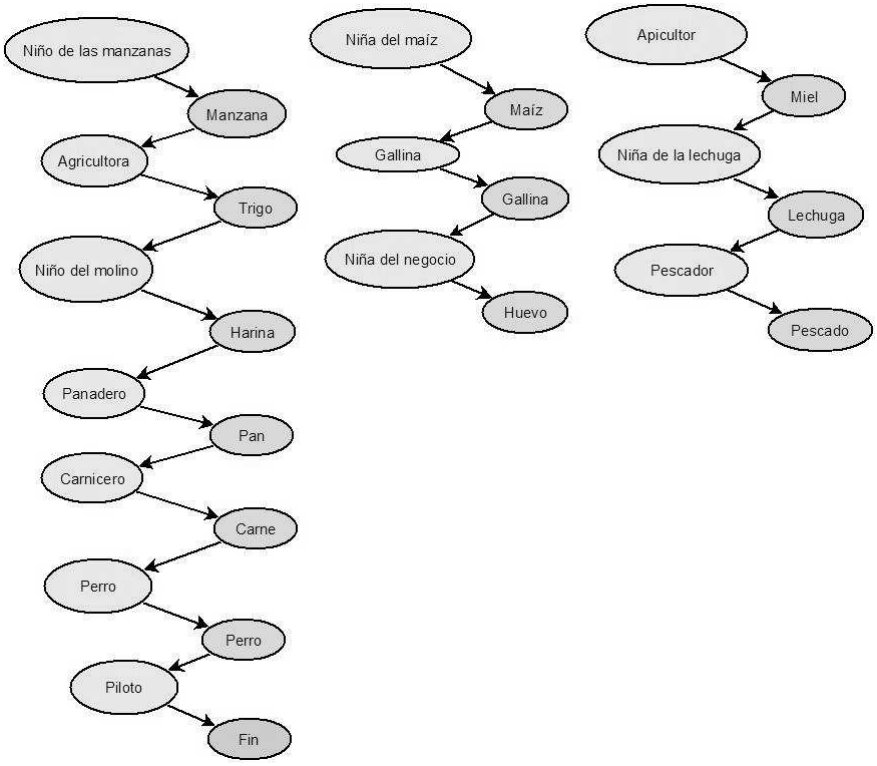


Figura 24: Diagrama que muestra los posibles flujos a seguir; el color verde representa a los personajes, el naranjo a los ítems y el celeste al final del juego.

Finalmente en la figura 25 se muestran algunas imágenes de la ejecución del prototipo 2 en el teléfono móvil Nokia 6230.

Figura 25: Imágenes del prototipo 2 en el teléfono móvil Nokia 6230.

6. Evaluación

6.1. Fundamentación Teórica

Si bien existen diversos métodos de evaluación de usabilidad, en este trabajo utilizamos sólo los métodos de observación no participante, cuestionario, feedback del usuario y test de usuario final. Para estos métodos se indican a continuación sus características para ser aplicados apropiadamente.

Antes entrar en detalles se debe señalar que la utilización correcta y provechosa de los métodos de usabilidad depende principalmente de dos factores o restricciones:

- **La etapa de desarrollo del sistema**: si bien todos los métodos pueden aplicarse antes, durante y después del desarrollo de un sistema, cada uno de ellos tiene características distintas y conviene aplicarlos en algunas etapas de desarrollo y no en otras [3].

- **La cantidad de usuarios disponibles**: existe una restricción de la cantidad de usuarios mínima con la que debemos contar para obtener resultados representativos al aplicar los distintos métodos [3].

A continuación describimos los distintos métodos que se usaron en este trabajo y los factores anteriormente nombrados que las caracterizan en su aplicación:

- **Observación no participante**: es conveniente realizarla durante la etapa de desarrollo del software. Este tipo observación se realiza guiando a un evaluador a través de una pauta, cuyo objetivo es registrar las acciones específicas realizadas por usuario final en la interacción con un sistema de acuerdo a lo que restringe la pauta; en el caso de la observación no participante el evaluador no interactúa con el usuario final. Para aplicar satisfactoriamente este método se necesitan al menos 3 usuarios. [3]

- **Cuestionarios**: pueden ser utilizados antes, durante y después de la etapa de desarrollo. Además se necesitan al menos 30 usuarios para utilizar este método con resultados relevantes. [3]

- **Test de usuario final**: es conveniente realizarlo después de la etapa de desarrollo. Este test corresponde a una serie de características de aceptación del sistema, a las cuales el usuario le asigna una evaluación numérica. Para aplicar satisfactoriamente este método se necesitan más de 100 de usuarios. [3]

- **Feedback del usuario**: es conveniente realizarlo durante la etapa de desarrollo. Se pregunta al alumno su opinión acerca del sistema; que cosas le gustan y desagradan, u otras observaciones importantes que quieran ser

consultadas. Para aplicar satisfactoriamente este método se necesitan cientos de usuarios. [3]

6.2. Evaluación previa

Previo al diseño y desarrollo de la aplicación fue importante conocer la opinión de los usuarios finales acerca de sus gustos y puntos de vista.

Para esto se consideró como referencia un estudio de usabilidad de un software educativo llamado Método Científico (descrito en el capítulo 2.9), el que fue llevado a cabo durante los meses de Octubre y Noviembre de 2008 y donde se aplicaron las metodologías de usabilidad antes nombradas. Sólo una parte específica de aquél trabajo de usabilidad es la que tiene relevancia para esta memoria y corresponde a los resultados obtenidos a través de las preguntas de feedback del usuario que se encuentran adjuntas al test de usuario final (ver anexo 9.1). Por esta razón en esta evaluación previa sólo se consideran los aspectos de la evaluación de usabilidad del software Método Científico que son relevantes para este trabajo y descartando las restantes partes de dicho estudio.

6.2.1. Objetivos

- Evaluar puntos de vista de los usuarios finales antes de desarrollar una aplicación educativa.

6.2.2. Muestra

Se contó con 6 cursos mixtos de educación general básica de 2 establecimientos diferentes, conformados por 40 a 45 alumnos con edades entre los 10 y 13 años:

- Colegio Arnaldo Falabella (colegio público):
  - 1 curso de 5º Básico.
  - 1 curso de 6º Básico.
  - 1 curso de 7º Básico.

- Colegio Madre Vicencia (particular subvencionado):
  - 2 cursos de 6º Básico.
  - 1 curso de 7º Básico.

6.2.3. Instrumentos

Se utilizó un Cuestionario de Feedback del usuario, el cual forma parte de la pauta de aceptación de usuario final, y está compuesto por preguntas que solicitan

información al alumno de lo que le gustó y no le gustó de la aplicación, qué cambios haría en el software, qué utilidad y otros usos le ve al software; finalmente también deja un espacio para una opinión abierta (ver anexo 9.2).

6.2.4. Procedimiento

En todos los cursos que consideraba la muestra se dieron las instrucciones previas al uso del software en la última sesión de trabajo; se llevaron copias impresas del test de aceptación de usuario final (ver anexo 9.2) y al final de esta última sesión de trabajo con el software, se solicitó a los alumnos que completaran este test.

6.2.5. Resultados

A continuación se resumen algunos aspectos relevantes:

- Gustaron mucho las etapas interactivas del software donde el usuario ejecutaba acciones y se desplegaban elementos gráficos.

- No gustaron las etapas donde el usuario desconoce los conceptos y etapas donde debe escribir.

- Los niños demandan elementos gráficos más llamativos y elementos entretenidos que motiven más a aprender. Este software no era un juego, por lo que proponían la inclusión de mayor diversión a través de juegos auxiliares.

En resumen, se puede señalar que los niños valoran mucho los elementos interactivos del software y la buena calidad de los elementos gráficos, agregando a esto más elementos de entretención y juegos. Con lo cual se cautivaría el elemento motivacional en los niños para que aprendan utilizando una aplicación educativa.

6.3. Evaluación del prototipo 1

En esta sesión de evaluación se probó el primer prototipo funcional de la aplicación (ver capitulo 5.5). Este prototipo muestra en ejecución el diseño de la aplicación, a través de un juego que fue construido a partir de una hiperhistoria básica. Este prototipo utilizó características base de la aplicación tales como el comportamiento de los elementos interactivos de prueba y los controles de movimiento del protagonista (ambos fueron finalmente los definitivos). También se dio una pincelada al uso de controles más complejos (control de texto más amplio que el área disponible y control de la presentación de las alternativas de una pregunta), donde no está muy clara la forma más adecuada de disponerlos al usuario. Por esta razón resultó imprescindible evaluar y detectar problemas en esta etapa de desarrollo para poder rediseñar y mejorar varios aspectos de la aplicación.

6.3.1. Objetivos

Evaluar y detectar problemas en:

- El diseño base del juego.
- Jugabilidad básica del juego.
- El uso de controles del juego.
- El uso de material interactivo.

6.3.2. Muestra

Se contó con 9 alumnos de cursos mixtos de 3º, 4º y 5º año de educación general básica del colegio José Joaquín Prieto Vial de la comuna de Maipú. Las edades de los alumnos fluctuaban entre los 9 y los 11 años.

Figura 26: entrada del colegio José Joaquín Prieto Vial.

6.3.3. Instrumentos

Se utilizó una pauta de observación y cuestionario de feedback de usuario (ver anexo 9.3). Con estos instrumentos se persigue cumplir con los objetivos propuestos. Hay que recalcar que la pauta no requiere entrar en mayor detalle de explicaciones, puesto que el evaluador y el desarrollador son la misma persona.

En la pauta de observación están por defecto incluidos los elementos principales de la tarea que se llevara a cabo, pero se contemplan espacios en blanco para otros elementos no esperados. Las acciones que se registraron fueron: si el alumno preguntó qué hacer con cada elemento, si al interactuar con cada elemento recorrió el texto en el ***área de diálogos*** con el ***scroll*** arriba y abajo (ya que intencionalmente se sobrepasa las dimensiones del ***área de diálogos*** en algunas ocasiones), si presionó el botón *Fire* para cerrar el dialogo con el elemento, si presionó el botón *Fire* para otra acción no implementada y si realizó la acción esperada con cada elemento.

Las preguntas de cuestionario de feedback del usuario fueron focalizadas en el ámbito me gusta/no me gusta listando los elementos utilizados en este prototipo. Estas preguntas fueron adaptadas de las preguntas de feedback del usuario de la pauta de usuario final del profesor Jaime Sánchez I. (ver anexo 9.1).

6.3.4. Procedimiento

Se contó con un teléfono móvil Nokia 6230 con la aplicación del prototipo 1 instalada. Además se llevaron copias impresas de la pauta de observación y cuestionario de feedback de usuario (ver anexo 9.3).

Individualmente se hicieron las pruebas de usabilidad donde se siguieron los siguientes pasos:

- Se entregaron las instrucciones verbales a cada niño(a) acerca de lo que había que hacer en el juego: *“Este es un prototipo de juego RPG para teléfonos celulares, el que está actualmente en desarrollo. El objetivo del juego es que captures a la cucaracha y la idea es que tú descubras como hacerlo interactuando con los elementos del juego. Los controles del juego corresponden solamente a los botones del joystick del teléfono: arriba, abajo, izquierda, derecha y botón central (Fire)”.*

- La secuencia resumida a seguir que resolvía el problema propuesto fue: intentar capturar a la cucaracha, no poder capturarla (porque necesitaba la antorcha), tomar la antorcha y capturar la cucaracha. Pero esto es un resumen ideal de la secuencia que resuelve el problema, ya que el niño(a) tuvo la libertad para recorrer el mapa por donde él quisiera, registrando las interacciones con otros elementos que no eran los principales.

- Si el niño(a) tuvo dudas de lo que había que hacer o como debía hacer algo se le ayudó y se registró en la pauta de observación la ayuda que requirió.

- En cada interacción de la secuencia resumida había un texto de diálogo que sobrepasaba el área de escritura, por lo que se probó si el niño(a) recorría el texto con los botones arriba y abajo. Esta acción también se registró en la pauta de observación.

- La acción de presionar el botón central (o *Fire*) para cerrar los diálogos también fue registrada, así como también el empleo de este botón para acciones no disponibles.

- En la pauta de observación se registro si realizó correctamente la sub-tarea relativa al ítem o personaje de la secuencia de pasos de resolución.

- Cuando el niño capturó a la cucaracha, se le pidió que fuera a interactuar con la tortuga y la cual le mostró una pregunta con 2 alternativas. Aquí se registró la forma de utilizar el control de manejo de alternativas.

- Finalmente se aplicó el cuestionario de feedback del usuario verbalmente, se leyeron las preguntas y se registraron las respuestas en la misma hoja.

Figura 27: fotografías de la evaluación del prototipo 1 en el colegio José Joaquín Prieto.

6.3.5. Resultados

De los datos recogidos de la aplicación de la pauta de observación se detectó lo siguiente:

- Los alumnos preguntaron muy poco acerca de lo que debían hacer, y cuando preguntaron, lo hicieron luego de interactuar con el primer elemento que colisionaron.

- Ningún alumno recorrió el texto de los diálogos con los botones arriba y abajo.

- Los alumnos presionaron el botón *Fire* como se esperaba y no lo presionaron para otras acciones.

- Los alumnos realizaron las sub-tareas esperadas para cada ítem y completaron la tarea completa.

- En la interacción final con la tortuga no fue intuitivo el recorrer las alternativas con los botones izquierda derecha, pero una vez que se les explicó cómo se recorrían, les pareció bien.

De los datos tomados del cuestionario de feedback del usuario se obtuvo:

Pregunta	No	Mas o menos	Si	Opiniones
¿Te gustaron las imágenes?	0	1 (11%)	8 (89%)	(-)Se ven borrosas, pueden ser más bonitas
¿Te gustaron las animaciones?	0	1 (11%)	8 (89%)	
¿Te gustaron los textos?	1 (11%)	1 (11%)	7 (77%)	(+)Están bien escritos, (+)texto gracioso de la cucaracha
¿Te gustaron los botones de control del teléfono?	0	0	9 (100%)	
¿Te gustó el control del personaje?	0	0	9 (100%)	
¿Te gustó la Modalidad de juego?	1 (11%)	0	8 (89%)	(-)No entendí el juego

Tabla 4: resumen de datos obtenidos con la pregunta acerca de los gustos por los elementos del juego.

Ante las preguntas:

- ¿Entendiste el objetivo del juego? los alumnos coincidieron en que se trataba de “capturar a la cucaracha con la antorcha”

- ¿Qué agregarías al juego? los alumnos indicaron los siguientes elementos: más árboles, plantas, arbustos, palmeras y vegetación. Agregar más monitos, y más monitos para confundir, cambiar personajes y agregar mayor dificultad.

- ¿Para qué podría servir el juego? los alumnos coincidieron en que podría servir para entretención.

6.3.6. Análisis

Los elementos interactivos de prueba (imágenes, animaciones y textos) fueron bien recibidos por los alumnos.

Se observó que los alumnos exploraron e interactuaron en forma natural con todo el entorno disponible, por lo tanto la interacción con el entorno del juego es altamente intuitiva.

Los alumnos piden en general una mayor cantidad de elementos gráficos de distintas características para agrandar más el juego. Esta es una característica que se intentó mejorar en el prototipo 2.

El ***scroll*** de texto resultó no ser intuitivo. Por esta razón, se consideraron las siguientes opciones para mejorar esto en el prototipo 2: restringirse a utilizar textos más cortos, ampliar el área de escritura de texto, incluir algún elemento que indique que hay mas texto hacia abajo o hacia arriba, o bien una combinación entre algunas o todas estas opciones.

Si bien los alumnos no recorren intuitivamente las alternativas, el hecho de explicar cómo se pueden escoger soluciona en parte el problema. De todas formas en el prototipo 2 para mejorar este aspecto se agregaron elementos guía que indicaron que el alumno puede presionar el botón izquierda y derecha.

Por otra parte los botones elegidos para controlar el juego y el control de los movimientos del personaje fueron unánimemente aceptados, lo que apoya la elección adoptada.

Finalmente, los alumnos entendieron que el objetivo del juego es la acción de capturar a la cucaracha con la antorcha.

6.4. Evaluación del prototipo 2

En esta sesión de evaluación se probó el segundo prototipo funcional de la aplicación (ver capitulo 5.6). Este prototipo muestra en ejecución el diseño de la aplicación, a través de un juego que fue construido a partir de una hiperhistoria más compleja en comparación a la del prototipo 1, con una mayor cantidad de ***Personajes***, ***Ítems*** y ***Decorativos***. También se puso en ejecución las características más complejas que no habían sido probadas anteriormente tales como el modelo de interacción y sus elementos asociados, y que son las que determinan la jugabilidad del juego. Por esta razón fue conveniente evaluar como perciben el juego los niños y detectar problemas de diseño que sean importantes.

6.4.1. Objetivos

Los objetivos consisten en evaluar y detectar problemas en:

- el diseño del juego.
- la jugabilidad.
- el uso de controles para:
    - el movimiento del protagonista.
    - la lectura de texto en los diálogos.
    - las interacciones con alternativas.
- el uso de material interactivo en general.

6.4.2. Muestra

Se contó con una muestra de 10 alumnos de cursos mixtos de 3º, 4º y 5º año de educación general básica del colegio José Joaquín Prieto Vial de la comuna de Maipú. Las edades de los alumnos fluctúan entre los 9 y los 11 años. Estos alumnos no son los mismos de la evaluación del prototipo 1.

6.4.3. Instrumentos

Se utilizó una adaptación (ver anexo 9.2) de la pauta de evaluación resumida de software para niños ciegos elaborada por el profesor Jaime Sánchez I. (ver anexo 9.1), en la cual se reemplazaron específicamente las preguntas que buscaban medir la interpretación de los sonidos por preguntas que buscan medir en cambio la interpretación de las imágenes:

- La pregunta ¿Me gustan los sonidos del software? se reemplazó por la pregunta ¿Me gustan las imágenes del juego?
- La pregunta ¿Los sonidos del software son claramente identificables? se reemplazó por la pregunta ¿Las imágenes del juego son claramente identificables?
- La pregunta ¿Los sonidos del software me transmiten información? se reemplazó por la pregunta ¿Las imágenes del juego me transmiten información?

Además en esta pauta se cambió la palabra “software” por “juego” en todos los lugares donde aparecía.

Otro elemento que se utilizó fue una pauta de observación (ver anexo 9.4) cuya intención es medir el desempeño de los usuarios con respecto al desempeño ideal de lo que el desarrollador espera en distintos aspectos del manejo de la aplicación (el uso de controles, la jugabilidad de la aplicación y la interacción con los elementos del juego). En

este caso la pauta no requiere entrar en mayor detalle de explicaciones puesto que el evaluador y el desarrollador son la misma persona.

6.4.4. Procedimiento

Se llevó un teléfono móvil Nokia 6230 con la aplicación del prototipo 2 instalada. Además se llevaron copias impresas de las pautas adaptadas de usuario final (ver anexo 9.2) y copias impresas de una pauta de observación (ver anexo 9.4).

Individualmente se hicieron las pruebas de usabilidad donde se siguieron los siguientes pasos:

- Se entregaron las instrucciones verbales a cada niño(a) acerca de lo que había que hacer en el juego: *"Este es un juego RPG para teléfonos celulares. En este juego hay una isla con personajes y otros elementos. El objetivo del juego es salir de la isla, para lograrlo debes lograr que el piloto pueda pilotear la avioneta. La idea es que tu recorras el mapa e interactúes con los personajes y resuelvas el inconveniente que tiene el piloto para manejar la avioneta. Los controles del juego corresponden solamente a los botones del joystick del teléfono: arriba, abajo, izquierda, derecha y botón central (Fire), donde además los botones arriba y abajo te sirven para mostrar mas texto cuando no se alcanza a mostrar completamente en el espacio asignado y los botones izquierda y derecha te sirven para recorrer las alternativas cuando los personajes te piden algo o te hacen preguntas".*

- Se entregó el teléfono móvil al alumno por alrededor de 5 minutos para que pudiera jugar.

- En este prototipo hubo solo una secuencia de pasos que resolvía el problema, sin embargo de acuerdo a como se hizo el planteamiento era valido probar con otras 2 soluciones alternativas que aparentemente podían ser soluciones validas para el alumno, pero que no resolvían el problema.

- Cuando el alumno tenía dudas acerca de qué debía hacer se le entregaron instrucciones que lo ayudaron a continuar.

- Al final de los 5 minutos se invitó a los alumnos a completar la pauta de evaluación de usuario final adaptada junto con el cuestionario de feedback del usuario. El evaluador por su parte completó la pauta de observación evaluando el desempeño del alumno en los distintos puntos.

Figura 28: fotografías de la evaluación del prototipo 2 en el colegio José Joaquín Prieto.

6.4.5. Resultados

A continuación se muestra el resumen de los resultados obtenidos con los distintos métodos de usabilidad.

De los datos capturados por la pauta de observación se obtuvo:

De acuerdo al manejo idealmente esperado en general…	No	Sí
¿Maneja correctamente los controles de movimiento?	0	10 (100%)
¿Recorre el mapa?	0	10 (100%)
¿Interactúa con los personajes?	0	10 (100%)
¿Interactúa con los decorativos?	0	10 (100%)
¿Ocupa alguna vez el scroll de texto arriba/abajo?	8 (80%)	2 (20%)
¿Recorre las alternativas?	0	10 (100%)
¿Obtiene los ítems que busca?	0	10 (100%)
¿Entiende la relación entre el ítem buscado y el personaje que lo necesita?	0	10 (100%)

Tabla 5: resumen de los datos obtenidos de la pauta de observación.

De los datos obtenidos de las pautas de usuario final se obtuvieron los siguientes resultados:

Alumno #	1	2	3	4	5	6	7	8	9	10	
Sexo (F: femenino, M: masculino	F	F	F	F	F	F	F	F	M	F	
Edad	11	11	11	9	9	10	10	8	10	10	PROMEDIO
¿Me gusta el juego?	10	9	10	10	10	9	10	7	10	10	**9,5**
¿El juego es entretenido?	9	8	10	1	9	10	10	9	10	10	**8,6**
¿El juego es desafiante?	10	7	9	9	1	7	7	10	5	10	**7,5**
¿El juego me hace estar activo?	10	9	8	6	6	10	7	10	8	10	**8,4**
¿Volvería a trabajar con el juego?	9	8	3	10	10	10	9	7	10	10	**8,6**
¿Recomendaría este juego a otros niños?	7	10	10	10	10	7	10	1	10	10	**8,5**
¿Aprendí con este juego?	1	8	6	10	10	7	10	10	10	10	**8,2**
¿Me sentí controlando las situaciones del juego?	10	10	6	10	1	3	10	5	7	10	**7,2**
¿Supe que hacer en cada paso del juego?	10	10	7	10	10	10	9	5	10	7	**8,8**
¿El juego es interactivo?	5	9	6	10	10	7	7	9	10	10	**8,3**
¿El juego es fácil de utilizar?	10	10	4	10	10	5	6	5	10	10	**8**
¿El juego es motivador?	10	7	4	9	10	10	8	9	10	10	**8,7**
¿El juego se adapta a mi ritmo?	10	9	5	10	9	10	9	5	1	10	**7,8**
¿El juego me permitió entender nuevas cosas?	6	7	7	9	10	10	10	9	10	10	**8,8**
¿Me gustan las imágenes del juego?	5	8	8	10	10	10	10	10	10	10	**9,1**
¿Las imágenes del juego son claramente identificables?	2	10	7	10	10	10	10	10	10	10	**8,9**
¿Las imágenes del juego me transmiten información?	1	9	10	5	10	10	7	9	10	10	**8,1**

Tabla 6: resumen de los resultados de las pautas de evaluación de usuario final (puntaje máximo 10 puntos).

En cada una de las preguntas de feedback de usuario final se recopilaron las siguientes ideas:

- ¿Qué te gustó del juego?
  - Buscar las cosas que la gente pide.
  - Buscar las cosas que la gente necesita.
  - Cuando tenía que darle la comida al perro.
  - Cuando tenía que entregarle las cosas a los demás.
  - La forma del juego, me gustan los juegos de aventura.
  - Los personajes, los campos.
  - Que anda ayudando a las personas.
  - Es entretenido.
  - Me gustó todo.

- ¿Qué no te gustó del juego?
  - Cuando me costaba encontrar las cosas que pedía.
  - Que no tenía instrucciones.
  - Que no encontrara un amigo.

- Había que pedir muchas cosas.

- ¿Qué le agregarías al juego?
  - Que los personajes hablen más.
  - Que sea más grande.
  - Instrucciones.
  - Más personajes.
  - Un personaje que le ayude.
  - Más personajes con problemas.
  - Una niña linda.
  - Mas cosas para comer y más monos que puedan hacer otras cosas.

- ¿Para qué crees que te puede servir el juego?
  - Para tener paciencia en buscar.
  - Para saber lo que son las cosechas.
  - Para saber más de los animales.
  - Para aprender, aprender a buscar, aprender a recorrer las partes del juego.
  - Para enseñar.
  - Para ayudar.
  - Para descubrir nuevas cosas.

- Observaciones:
  - El juego me gustó.
  - Es muy bueno.
  - El juego estuvo increíble.
  - Estuvo muy bueno, es recomendable.

6.4.6. Análisis

En los resultados de la observación se obtiene que en general las funcionalidades que el software implementa son bien recibidas por el alumno, sin embargo un tema pendiente es el ***scroll*** de texto con los botones arriba y abajo, los cuales fueron agregados en forma intencional en algunas interacciones donde el texto sobrepasaba el área de diálogo, los cuales siguen mostrando una tendencia negativa en su uso.

En cuanto a la aceptación del juego en general los resultados fueron satisfactorios en cada uno de los indicadores.

Los niños se mostraron muy a gusto con el juego, su jugabilidad y los elementos que tenían a su disposición.

Sin embargo, los alumnos siempre quieren aumentar en general el tamaño del juego con más ***Personajes***, más diálogos, más ***Ítems*** y otras funciones. También

destaca el hecho de mejorar las instrucciones, las cuales deben ser manejadas dentro de cada Interacción que constituye un nodo de la hiperhistoria.

Los alumnos coinciden plenamente en percibir que el juego sirve para aprender cosas y a diferencia de la evaluación del prototipo 1, en la pregunta que pide la opinión a los alumnos sobre para qué sirve el juego, en esta evaluación ningún alumno nombró la palabra “entretención” como respuesta, lo cual refleja la afirmación inicialmente realizada. Sin embargo, el tema de medir el impacto que tiene la aplicación en el aprendizaje de algún tipo de contenido debe ser medido más adelante, ya que no fue contemplado en este trabajo.

7. Conclusiones y trabajo futuro

En este trabajo se cumplieron satisfactoriamente los objetivos de diseñar e implementar un motor de videojuegos educativos RPG. Este motor ofrece un conjunto de funcionalidades que permite desarrollar sobre él un videojuego educativo RPG. Durante este trabajo se desarrolló el motor y se utilizó para la creación de los prototipos 1 y 2, los cuales fueron testeados con usuarios finales.

Otro de los objetivos planteados fue apoyar el desarrollo de la aplicación a través de la evaluación de usabilidad de los prototipos de videojuegos educativos RPG durante las etapas de desarrollo. Con esto se logró mejorar el diseño de la aplicación a medida que se fue desarrollando y a la vez justificar decisiones de diseño e implementación apoyados en los resultados de las evaluaciones.

Los resultados de las evaluaciones de usabilidad evidenciaron que los alumnos adoptaron en forma natural la interacción con los elementos del juego. La jugabilidad de juego también fue bien recibida por los alumnos, los cuales comprendieron la dinámica del juego, consistente en entregar cosas a los ***Personajes*** y recibir otras cosas a cambio, logrando efectivamente estimular su capacidad de resolución de problemas. A esto agregamos que los alumnos entienden que el juego sirve para aprender. En resumen con esto podemos avalar en parte que se obtuvo una herramienta que apoya el aprendizaje escolar a través de la estimulación de la capacidad de resolución de problemas del usuario.

Los alumnos mostraron en general una buena aceptación de la aplicación, lo cual incrementa su motivación para usar esta herramienta, apoyándose en los resultados de las pautas de evaluación de usuario final. Es importante señalar que la muestra no es representativa, ya que se necesitan más de 100 usuarios que respondan la pauta de evaluación de usuario final para que el resultado sea sobre una muestra representativa. En este sentido queda propuesto evaluar con más usuarios la aplicación como trabajo futuro, para asegurar la aceptación de la aplicación.

Un testeo cognitivo de la aplicación queda propuesto como trabajo futuro. Con esto se podría medir el impacto en el aprendizaje que genera el uso de esta aplicación y con ello avalar completamente el objetivo que tiene relación con obtener una herramienta que apoye el aprendizaje escolar a través de la estimulación de la capacidad de resolución de problemas del usuario.

La herramienta consiste en una aplicación que cumple con la expectativa de llegar a una cantidad considerable de usuarios. Esto se debe principalmente a que la aplicación fue diseñada y desarrollada para ser ejecutada en teléfonos móviles que cuenten con MIDP 2.0, adoptando intencionalmente como teléfono de prueba para el desarrollo, un modelo relativamente antiguo, como es el Nokia 6230 que fue lanzado en Octubre de 2003 [46], el que cuenta con características de display y memoria para ejecutar programas más reducidas si lo comparamos con un teléfono más moderno. Si bien el desempeño del teléfono móvil Nokia 6230 al ejecutar la aplicación desarrollada

fue cualitativamente aceptable, en el teléfono móvil Sony Ericsson W300i (un modelo del año 2006), se observó una ejecución de la aplicación mucho más fluida y en un display más amplio; en general, si se utilizan teléfonos más modernos, la experiencia con la aplicación mejora considerablemente producto de una mejor capacidad de procesamiento, una mayor memoria de ejecución para las aplicaciones y una mayor resolución del display. Si sumamos a esto, que los teléfonos móviles ofrecen una relación alumno/dispositivo cercana al 1:1 y que los teléfonos móviles que cuentan con los requisitos establecidos para la implementación del software, conforman un considerablemente alto porcentaje del total de los teléfonos móviles, se puede inferir que se asegura la portabilidad de la aplicación a una gran cantidad de dispositivos.

La arquitectura de la solución facilita la edición y creación de diferentes videojuegos tanto en contenido, como en objetivos educativos, ya que se crearon funciones de alto nivel que permiten llevar a cabo esta labor minimizando las intervenciones en el código, esto último sujeto a conservar la estructura base (elementos gráficos y control de las acciones de la interfaz de usuario) que ofrece el motor de juegos. Sin embargo, la arquitectura de la solución también provee la capacidad de extender nuevas funcionalidades del motor de videojuegos a partir de la estructura base implementada; esto se logra creando nuevas clases que redefinen las funcionalidades existentes o bien agregando nuevas funcionalidades, tal como se vio en el capítulo 5.3 de arquitectura de la solución.

En resumen se cumplieron satisfactoriamente todos los objetivos propuestos para este trabajo.

8. Referencias

[1] Gros, B., & Aguayos, J. (2004). Pantallas, juegos y educación: La alfabetización digital en la escuela. Bilbao: Desclée de Brouwer.

[2] Gros, B. (Coord.) (2008). Videojuegos y Aprendizaje.

[3] Nielsen, J. (1994) Usability Engineering. New York: Morgan Kauffmann Publishers; 1st edition, 362 p.

[4] http://java.sun.com/, último acceso Julio 2009.

[5] http://java.ciberaula.com/articulo/introduccion_j2me/, último acceso Julio 2009.

[6] http://java.sun.com/products/sjwtoolkit/, último acceso Julio 2009.

[7] Salinas, A., Sánchez, J. (2005). Uso de PDAs en el Entorno Escolar. En Sánchez, J. (editor). Nuevas Ideas en Informática Educativa, pp. 32-40. Santiago de Chile: Lom Ediciones S.A.

[8] Sánchez, J., Salinas, A. (2006). PDAs and Ubiquitous Computing in the School. Human Centered Technology Workshop 2006. Pori, Finland, Junio 11-13, 2006, pp. 249–258.

[9] Sánchez, J., Elías, M. (2006). Aprendizaje de ciencias a través de audio en niños ciegos. En Sánchez, J. (editor). Nuevas Ideas en Informática Educativa, pp. 11-21. Santiago de Chile: Lom Ediciones S.A.

[10] Sánchez, J., Salinas, A., Saenz, M. (2007) Mobile Game-Based Methodology for Cience Learning, In Jacko (Ed.). Human-Computer Interaction, Part IV, HCII 2007, LNCS 4553, pp. 322-331, 2007.

[11] Zurita, G., & Nussbaum, M. (2007). A conceptual framework based on activity theory for mobile CSCL. British Journal of Educational Technology, 38(2), pp. 211-235.

[12] www.c5.cl, http://www.c5.cl, último acceso Julio 2009.

[13] www.darkshire.net, http://www.darkshire.net/~jhkim/rpg/whatis/computer.html, último acceso Julio 2009.

[14] Shaffer, D. 2005. Epistemic games. Innovate 1 (6). http://www.innovateonline.info/index.php?view=article&id=79, último acceso Agosto 2008.

[15] Shaffer, D. W., Squire, K. R., Halverson, R., & Gee, J. P. (2005). Video Games and the Future of Learning. Phi Delta Kappan, 87 (2), pp. 104-111.

[16] Shaffer, D. W. (2006). Epistemic frames for epistemic games. Computers and Education. 46 (3), pp. 223-234.

[17] Prensky, M. (2001). Digital Natives, Digital Immigrants Part 1. On The Horizon - The Strategic Planning Resource for Education Professionals, 9 (5), 1-6.

[18] Prensky, M. (2001). Digital Natives, Digital Immigrants Part 2: Do They Really Think Differently? On the Horizon, 9 (6), 1-6.

[19] Prensky, M. (2006). Listen to the Natives. Educational Leadership. 63 (4), 8-13.

[20] Prensky, M. (2005). "Engage Me or Enrage Me": What Today's Learners Demand. Educause Review, 40 (5), 60,62,64.

[21] www.alegsa.com.ar, http://www.alegsa.com.ar/Dic/dispositivo%20movil.php, último acceso Julio 2009.

[22] leo.ugr.es, http://leo.ugr.es/J2ME/INTRO/intro_4.htm, último acceso Julio 2009.

[23] leo.ugr.es, http://leo.ugr.es/J2ME/INTRO/intro_5.htm, último acceso Julio 2009.

[24] www.java.com, http://www.java.com/es/download/faq/whatis_j2me.xml, último acceso Julio 2009.
[25] leo.ugr.es, http://leo.ugr.es/J2ME/INTRO/index2.htm, último acceso Julio 2009.
[26] leo.ugr.es, http://leo.ugr.es/J2ME/INTRO/intro_1.htm, último acceso Julio 2009.
[27] www.xtec.cat, http://www.xtec.cat/~jcorder1/problema.htm, último acceso Julio 2009.
[28] www.winmates.net, http://www.winmates.net/includes/polya.php, último acceso Julio 2009.
[29] ww2.educarchile.cl, http://ww2.educarchile.cl/eduteca/7mm/sitio/respuesta3.htm, último acceso Julio 2009.
[30] www.dcc.uchile.cl/~jsanchez/Pages/papers Hiper-Historias para Hiper-Aprender.
[31] www.dcc.uchile.cl/~jsanchez/Pages/papers Análisis de una Metodología para Construir Hiperhistorias.
[32] www.ldc.usb.ve, http://www.ldc.usb.ve/~abianc/hipertexto.html#Definiciones, último acceso Julio 2009.
[33] leo.ugr.es, http://leo.ugr.es/J2ME/INTRO/intro_2.htm, último acceso Julio 2009.
[34] grasia.fdi.ucm.es, http://grasia.fdi.ucm.es/j2me/_J2METech/index.html, último acceso Julio 2009.
[35] www.mailxmail.com, http://www.mailxmail.com/curso-desarrollo-aplicaciones-dispositivos-inalambricos-j2me/introduccion, último acceso Julio 2009.
[36] leo.ugr.es, http://leo.ugr.es/J2ME/INTRO/intro_2b.htm, último acceso Julio 2009.
[37] Nancy Hitschfeld, apuntes del curso “Programación Orientada al Objeto”.
[38] Java design patterns 101, David Gallardo, 08 Jan 2002
[39] A Pattern Language: Towns, Buildings, Construction, by Christopher Alexander (Oxford University Press, 1977)
[40] Design Patterns: Elements of Reusable Object-Oriented Software, by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (Addison-Wesley, 1995).
[41] Jaime Sánchez I., apuntes del curso “Taller de Usabilidad de Interfaces de Software”.
[42] http://www.gimp.org/, último acceso Julio 2009.
[43] Sánchez, J., Mendoza, C., Salinas, A. (2009). Mobile serious games for collaborative problem solving. In Brenda K. Wiederhold and Giuseppe Riva (Editors) the Annual Review of Cybertherapy and Telemedicine 2009. Amsterdam: Studies in Health Technology and Informatics (SHTI) series, IOS Press, Volume 144, pp. 193-197.
[44] Sánchez, J., Sáenz, M. (2009) Video Gaming for Blind Learners School Integration in Science Classes. T. Gross et al. (Eds.): INTERACT 2009, Part I, LNCS 5726, pp. 36-49, 2009.
[45] Olivares, R. (2009). Tesis para optar al grado de magíster en educación mención en informática educativa, Universidad de Chile, Facultad de Ciencias Sociales.

[46] www.forum.nokia.com, http://www.forum.nokia.com/devices/6230, último acceso Octubre 2009.
[47] Prensky, M. (2001). Digital Game-Based Learning. New York: McGraw-Hill.
[48] Guerrero, L., Ochoa, S., Pino, J. (2006). Selecting Computing Devices to Support Mobile Collaboration. GROUP DECISION AND NEGOTIATION Volume 15, Issue 3, pp. 243-271, May 2006.
[49] www.actiontrip.com, http://www.actiontrip.com/features/tenmostinfluentialrpgsofourtime.phtml, último acceso Octubre 2009.
[50] www.bestrpggames.net, http://www.bestrpggames.net/2009/08/15/best-rpg-pc-games-top-10-pc-rpg-games/, último acceso Octubre 2009.
[51] edugamesblog.wordpress.com, http://edugamesblog.wordpress.com/2007/12/15/the-top-10-free-educational-video-games/, último acceso Octubre 2009.
[52] www.gamemobile.co.uk, http://www.gamemobile.co.uk/mobile-games-reviews.php?limit=20&genre=16&Go=Go, último acceso Octubre 2009.

9. Anexos

9.1. Pauta de evaluación de usuario final

Pauta resumida usuario final.
"Evaluación de Usabilidad de Software para niños ciegos"
Dr. Jaime Sánchez I. Universidad de Chile

La presente pauta tiene por objetivo evaluar la usabilidad de un software para niños ciegos.

Nombre del Software	

Nombre del niño	Edad	Sexo

	Poco									Mucho
	1	2	3	4	5	6	7	8	9	10
¿Me gusta el software?										
¿El software es entretenido?										
¿El software es desafiante?										
¿El software me hace estar activo?										
¿Volvería a trabajar con el software?										
¿Recomendaría este software a otros niños?										
¿Aprendí con este software?										
¿Me sentí controlando las situaciones del software?										
¿Supe que hacer en cada paso del software?										
¿El software es interactivo?										
¿El software es fácil de utilizar?										
¿El software es motivador?										
¿El software se adapta a mi ritmo?										
¿El software me permitió entender nuevas cosas?										
¿Me gustan los sonidos del software?										
¿Los sonidos del software son claramente identificables?										
¿Los sonidos del software me transmiten información?										

Cuestionario

1.- ¿Qué te gustó del software?
2.- ¿Qué no te gustó del software?
3.- ¿Qué agregarías al software?
4.- ¿Para qué crees que te puede servir el software? ¿Qué otros usos le darías al software?
Observaciones o comentarios

9.2. Pauta adaptada de evaluación de usuario final

Pauta resumida usuario final.
"Evaluación de Usabilidad de Software"
Dr. Jaime Sánchez I. Universidad de Chile

La presente pauta tiene por objetivo evaluar la usabilidad de un juego para niños.

Nombre del Juego	

Nombre del niño	Edad	Sexo

	Poco									Mucho
	1	2	3	4	5	6	7	8	9	10
¿Me gusta el juego?										
¿El juego es entretenido?										
¿El juego es desafiante?										
¿El juego me hace estar activo?										
¿Volvería a trabajar con el juego?										
¿Recomendaría este juego a otros niños?										
¿Aprendí con este juego?										
¿Me sentí controlando las situaciones del juego?										
¿Supe que hacer en cada paso del juego?										
¿El juego es interactivo?										
¿El juego es fácil de utilizar?										
¿El juego es motivador?										
¿El juego se adapta a mi ritmo?										
¿El juego me permitió entender nuevas cosas?										
¿Me gustan las imágenes del juego?										
¿Las imágenes del juego son claramente identificables?										
¿Las imágenes del juego me transmiten información?										

Cuestionario

1.- ¿Qué te gustó del juego?
2.- ¿Qué no te gustó del juego?
3.- ¿Qué agregarías al juego?
4.- ¿Para qué crees que te puede servir el juego? ¿Qué otros usos le darías al juego?
Observaciones o comentarios

9.3. Pauta de observación 1 y encuesta feedback de usuario

Nombre:
Edad:
Sexo:
Nivel:
Establecimiento:

Al interactuar con:	¿Pregunta que hacer?	¿Recorre texto arriba/abajo?	¿Presiona Fire para cerrar dialogo?	¿Presiona Fire para que acción inesperada?	¿Realiza la acción esperada?
Cucaracha					
Antorcha					
Tortuga					
Poción					

Encuesta de feedback de usuario

	¿Te gusta? Escala: No - Más o Menos – Sí	Opinión
Imágenes		
Animaciones		
Textos		
Botones de control		
Control personaje		
Modalidad de Juego		

¿Entendiste Jugabilidad / sucesos del juego?

¿Qué agregarías al juego?

¿Para que podría servir el juego?

9.4. Pauta de observación 2

Nombre:
Edad:
Sexo:
Nivel:
Establecimiento:

De acuerdo al manejo idealmente esperado…	Escala: Sí - No	Observaciones
¿Maneja correctamente los controles de movimiento?		
¿Recorre el mapa?		
¿Interactúa con los personajes?		
¿Interactúa con los decorativos?		
¿Ocupa el scroll de texto?		
¿Recorre las alternativas?		
¿Obtiene los ítems que busca?		
¿Entiende la relación entre el ítem buscado y el personaje que lo necesita?		

9.5. Interfaz iLogica

```
public interface iLogica {

    /**
     *
     */
    void IniciarBolso();

    /**
     * @return
     */
    int[] getIdItemsBolso();

    /**
     *
     */
    void NextItemBolso();

    /**
     *
     */
    void OcultarBolso();

    /**
     *
     */
    void PreviousItemBolso();

    /**
     * @param fkIdConcepto
     * @param fkIdGrafica
     */
    void addItemDeProtagonista(int fkIdConcepto, int fkIdGrafica);

    /**
     * @param fkIdConcepto
     * @param fkIdGrafica
     * @param k
     */
    void addItemDeProtagonista(int fkIdConcepto, int fkIdGrafica, int k);

    /**
     * @param condiciones
     * @return
     */
    boolean checkCondiciones(iCondicion[] condiciones);

    /**
     * @param idConcepto
     * @return
     */
    iConcepto getConcepto(int idConcepto);

    /**
     * @param idGrafica
     * @return
     */
    iConcepto getConceptoBy(int idGrafica);

    /**
     * @param fkIdConcepto
     * @return
     */
    int getCountItemDeProtagonistaBy(int fkIdConcepto);


    /**
     * @param fkIdConcepto
     * @return
     */
    iGrafica getGraficaItemDeProtagonistaBy(int fkIdConcepto);

    /**
     * @return
     */
    int getIdConceptoActualBolso();

    /**
     * @param fkIdGrafica
     * @return
     */
    iInteraccion getInteraccionActualBy(int fkIdGrafica);

    /**
     * @param fkIdGrafica
     */
    void setInteraccionSiguienteFor(int fkIdGrafica);

    /**
     * @param fkIdGrafica
     * @return
```

```
     */
    Sprite getSprite(int fkIdGrafica);

    /**
     * @return
     */
    iSpriteProtagonista getSpriteProtagonista();

    /**
     * @param fkIdGrafica
     * @return
     */
    TiledLayer getTiledLayer(int fkIdGrafica);

    /**
     * @return
     */
    iTiledLayerBase getTiledLayerBaseForEscenario();

    /**
     *
     */
    void iniciarAnimaciones();

    /**
     * @param idGrafica
     */
    void quitarGrafica(int idGrafica);

    /**
     * @param fkIdConcepto
     */
    void removeItemDeProtagonista(int fkIdConcepto);

    /**
     * @param fkIdConcepto
     * @param k
     */
    void removeItemDeProtagonista(int fkIdConcepto, int k);

    /**
     * @param sprite
     * @return
     */
    boolean spritePersonajeCollides(Sprite sprite);

    /**
     * @return
     */
    MensajeColision spriteProtagonistaCollides();

    /**
     * @param lm
     * @throws java.io.IOException
     */
    void updateEscenario(LayerManager lm) throws IOException;

    /**
     * @return
     */
    int getXBolso();

    /**
     * @param xBolso
     */
    void setXBolso(int xBolso);

    /**
     * @return
     */
    int getYBolso();

    /**
     * @param yBolso
     */
    void setYBolso(int yBolso);

    /**
     *
     */
    void setEstadoInicial();
}
```

9.6. Función setDatos() del prototipo 1

```
    private void setDatos() throws IOException
    {
        Escenarios escenarios = new Escenarios();
        escenarios.addEscenario(new Escenario(0));
        this._escenarios = escenarios;

        this._tiledLayerBolso = new TiledLayerElemento(1,this._gameDesign.getBolso());
        this._tiledLayerBolso.getTiledLayer().setVisible(false);
        //tabla uso estado actual
        SpritesProtagonistas spritesProtagonistas = new SpritesProtagonistas();
        spritesProtagonistas.addSpriteProtagonista(new SpriteProtagonista(0, this._gameDesign.getKarel(), this._gameDesign.KarelSeqWalkDownDelay, this._timer,
this._gameDesign.KarelSeqWalkUp,Sprite.TRANS_NONE, this._gameDesign.KarelSeqWalkDown, Sprite.TRANS_NONE, this._gameDesign.KarelSeqWalkSide,
Sprite.TRANS_MIRROR, this._gameDesign.KarelSeqWalkSide, Sprite.TRANS_NONE));
        this._spritesProtagonistas = spritesProtagonistas;

        //tabla uso estado actual
        TiledLayersBases tiledLayersBases = new TiledLayersBases();
        tiledLayersBases.addTiledLayerBase(new TiledLayerBase(0,this._gameDesign.getBase()));
        this._tiledLayersBases = tiledLayersBases;

        //id referenciado en graficas
        SpritesPersonajes spritesPersonajes = new SpritesPersonajes();
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(0, _gameDesign.getThomas(), _gameDesign.ThomasSeqWalkHorizDelay, this._timer, this,
_gameDesign.ThomasSeqWalkHoriz, Sprite.TRANS_ROT180, Sprite.TRANS_NONE, GameCanvas.RIGHT, 10));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(1, _gameDesign.getTortuga(), _gameDesign.tortugaseq001Delay, this._timer, this,
_gameDesign.tortugaseq001, Sprite.TRANS_NONE, Sprite.TRANS_MIRROR, GameCanvas.LEFT, 7));
        this._spritesPersonajes = spritesPersonajes;

        //id referenciado en graficas
        TiledLayersElementos tiledLayersElementos = new TiledLayersElementos();
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(2,this._gameDesign.getThings()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(3,this._gameDesign.getTrees()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(4,this._gameDesign.getWater(), this._gameDesign.AnimWaterWater,
this._gameDesign.AnimWaterSeq001, this._gameDesign.AnimWaterSeq001Delay,this._timer));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(5,this._gameDesign.getLlama2(), this._gameDesign.animLlamaLlama2,
this._gameDesign.animLlamaseq001, this._gameDesign.animLlamaseq001Delay, this._timer));
        this._tiledLayersElementos = tiledLayersElementos;

        //id referenciado en interacciones
        Graficas graficas = new Graficas();
        graficas.addGrafica(new Animacion(0,0,0));
        graficas.addGrafica(new Animacion(1,0,1));
        graficas.addGrafica(new Capa(2,0,2));
        graficas.addGrafica(new Capa(3,0,3));
        graficas.addGrafica(new Capa(4,0,4));
        graficas.addGrafica(new Capa(5,0,5));

        this._graficas = graficas;

        //id referenciado en grafica y condiciones
        Conceptos conceptos = new Conceptos();
        conceptos.addConcepto(new Personaje(0,"Cucaracha","Es muy sensible al calor "));
        conceptos.addConcepto(new Personaje(1,"Tortuga","Soy una tortuga muy preguntona "));
        conceptos.addConcepto(new Item(2, "Items", ""));
        conceptos.addConcepto(new Decorativo(3, "Arboles", ""));
        conceptos.addConcepto(new Decorativo(4, "Agua", ""));
        conceptos.addConcepto(new Item(5, "Antorcha", ""));

        this._conceptos = conceptos;

        Interacciones interacciones = new Interacciones();
        iCondicion[] condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(5);
        ConceptoTransaccion[] salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(0,1);
        iRespuesta r1= new Items("Ay, me quemaste y me capturaste", "Lero Lero", null, salidas);
        interacciones.addInteraccion(new Recoger(0,0,condiciones," ", "Lero Lero no me atrapas ",r1));

        ConceptoTransaccion[] salidas2 = new ConceptoTransaccion[1];
        salidas2[0] = new ConceptoTransaccion(5,1);
        iRespuesta r2 = new Items("Has conseguido una antorcha de fuego","",null,salidas2);
        interacciones.addInteraccion(new Recoger(0, 5, null, "", "", r2));

        iCondicion[] condiciones3 = new iCondicion[1];
        condiciones3[0]=new TenerConcepto(0);
        String[] alternativas = {"a) No tengo la menor idea ","b) Con el fuego de una antorcha "};
        iRespuesta r3 = new Alternativas("correcto ","incorrecto ",alternativas, 1, null);
        interacciones.addInteraccion(new Responder(0, 1, condiciones3,"¿Como capturaste a la cucaracha? ", "Hola en cualquier momento te hare una pregunta ", r3));

        this._interacciones = interacciones;

    }
```

9.7. Función setDatos() del prototipo 2

```
    private void setDatos() throws IOException
    {
        iCondicion[] condiciones;
        ConceptoTransaccion[] entradas;
        ConceptoTransaccion[] salidas;
        iRespuesta respuesta;

        Escenarios escenarios = new Escenarios();
        escenarios.addEscenario(new Escenario(0));
        this._escenarios = escenarios;

        //tabla uso estado actual
        SpritesProtagonistas spritesProtagonistas = new SpritesProtagonistas();
        spritesProtagonistas.addSpriteProtagonista(new SpriteProtagonista(0, this._gameDesign.getHeroe(), this._gameDesign.heroeDownDelay, this._timer,
this._gameDesign.heroeUp,Sprite.TRANS_NONE, this._gameDesign.heroeDown, Sprite.TRANS_NONE, this._gameDesign.heroeRight, Sprite.TRANS_MIRROR,
this._gameDesign.heroeRight, Sprite.TRANS_NONE));
        spritesProtagonistas.addSpriteProtagonista(new SpriteProtagonista(1, this._gameDesign.getHeroina(), this._gameDesign.heroinaDownDelay, this._timer,
this._gameDesign.heroinaUp,Sprite.TRANS_NONE, this._gameDesign.heroinaDown, Sprite.TRANS_NONE, this._gameDesign.heroinaRight, Sprite.TRANS_MIRROR,
this._gameDesign.heroinaRight, Sprite.TRANS_NONE));
        this._spritesProtagonistas = spritesProtagonistas;

        //tabla uso estado actual
        TiledLayersBases tiledLayersBases = new TiledLayersBases();
        tiledLayersBases.addTiledLayerBase(new TiledLayerBase(0,this._gameDesign.getSuperficieBase()));
        this._tiledLayersBases = tiledLayersBases;

        //id referenciado en graficas
        SpritesPersonajes spritesPersonajes = new SpritesPersonajes();
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(0, _gameDesign.getNino1(), _gameDesign.nino1RightDelay, this._timer, this, _gameDesign.nino1Right,
Sprite.TRANS_MIRROR, Sprite.TRANS_NONE, GameCanvas.RIGHT, 5));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(1, _gameDesign.getNino2(), _gameDesign.nino2RightDelay, this._timer, this, _gameDesign.nino2Right,
Sprite.TRANS_MIRROR, Sprite.TRANS_NONE, GameCanvas.RIGHT, 5));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(2, _gameDesign.getNino3(), _gameDesign.nino3RightDelay, this._timer, this, _gameDesign.nino3Right,
Sprite.TRANS_MIRROR, Sprite.TRANS_NONE, GameCanvas.RIGHT, 5));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(3, _gameDesign.getNina1(), _gameDesign.nina1RightDelay, this._timer, this, _gameDesign.nina1Right,
Sprite.TRANS_MIRROR, Sprite.TRANS_NONE, GameCanvas.RIGHT, 5));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(4, _gameDesign.getNina2(), _gameDesign.nina2RightDelay, this._timer, this, _gameDesign.nina2Right,
Sprite.TRANS_MIRROR, Sprite.TRANS_NONE, GameCanvas.RIGHT, 5));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(5, _gameDesign.getNina3(), _gameDesign.nina3RightDelay, this._timer, this, _gameDesign.nina3Right,
Sprite.TRANS_MIRROR, Sprite.TRANS_NONE, GameCanvas.RIGHT, 5));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(6, _gameDesign.getAgricultora(), _gameDesign.agricultoraWorkDelay, this._timer));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(7, _gameDesign.getCarnicero(), _gameDesign.carniceroWorkDelay, this._timer));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(8, _gameDesign.getCocinero(), _gameDesign.cocineroWorkDelay, this._timer));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(9, _gameDesign.getLenador(), _gameDesign.lenadorWorkDelay, this._timer));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(10, _gameDesign.getPescador(), _gameDesign.pescadorWorkDelay, this._timer));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(11, _gameDesign.getPerro(), _gameDesign.perroseq001Delay, this._timer, this, _gameDesign.perroseq001,
Sprite.TRANS_NONE, Sprite.TRANS_MIRROR, GameCanvas.RIGHT, 5));
        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(12, _gameDesign.getGallina(), _gameDesign.gallinaRightDelay, this._timer, this, _gameDesign.gallinaRight,
Sprite.TRANS_MIRROR, Sprite.TRANS_NONE, GameCanvas.RIGHT, 5));

        spritesPersonajes.addSpritePersonaje(new SpritePersonaje(13, _gameDesign.getTortuga(), _gameDesign.tortugaLeftDelay, this._timer, this, _gameDesign.tortugaLeft,
Sprite.TRANS_NONE, Sprite.TRANS_MIRROR, GameCanvas.LEFT, 3));
        this._spritesPersonajes = spritesPersonajes;

        //id referenciado en graficas
        TiledLayersElementos tiledLayersElementos = new TiledLayersElementos();
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(14,this._gameDesign.getAgua()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(15,this._gameDesign.getSolido01()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(16,this._gameDesign.getSolido02()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(17,this._gameDesign.getSolido03()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(18,this._gameDesign.getSolidoAlmacen()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(19,this._gameDesign.getSolidoArboles()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(20,this._gameDesign.getSolidoAvioneta()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(21,this._gameDesign.getSolidoBotePescador()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(22,this._gameDesign.getSolidoCampoLechuga()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(23,this._gameDesign.getSolidoCampoMaiz()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(24,this._gameDesign.getSolidoCampoTrigo()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(25,this._gameDesign.getSolidoCarniceria()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(26,this._gameDesign.getSolidoCasa()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(27,this._gameDesign.getSolidoColmenas()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(28,this._gameDesign.getSolidoManzanos()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(29,this._gameDesign.getSolidoMolino()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(30,this._gameDesign.getSolidoPanaderia()));

        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(31,this._gameDesign.getItemCarne()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(32,this._gameDesign.getItemDinero()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(33,this._gameDesign.getItemHarina()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(34,this._gameDesign.getItemHuevo()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(35,this._gameDesign.getItemLeche()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(36,this._gameDesign.getItemLechuga()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(37,this._gameDesign.getItemLena()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(38,this._gameDesign.getItemMaiz()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(39,this._gameDesign.getItemManzana()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(40,this._gameDesign.getItemMiel()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(41,this._gameDesign.getItemPan()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(42,this._gameDesign.getItemPescado()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(43,this._gameDesign.getItemQueso()));
        tiledLayersElementos.addTiledLayerElemento(new TiledLayerElemento(44,this._gameDesign.getItemTrigo()));

        this._tiledLayersElementos = tiledLayersElementos;
```

```
//id referenciado en interacciones
Graficas graficas = new Graficas();
graficas.addGrafica(new Animacion(0,0,0));
graficas.addGrafica(new Animacion(1,0,1));
graficas.addGrafica(new Animacion(2,0,2));
graficas.addGrafica(new Animacion(3,0,3));
graficas.addGrafica(new Animacion(4,0,4));
graficas.addGrafica(new Animacion(5,0,5));
graficas.addGrafica(new Animacion(6,0,6));
graficas.addGrafica(new Animacion(7,0,7));
graficas.addGrafica(new Animacion(8,0,8));
graficas.addGrafica(new Animacion(9,0,9));
graficas.addGrafica(new Animacion(10,0,10));
graficas.addGrafica(new Animacion(11,0,11));
graficas.addGrafica(new Animacion(12,0,12));
graficas.addGrafica(new Animacion(13,0,13));
graficas.addGrafica(new Capa(14,0,14));
graficas.addGrafica(new Capa(15,0,15));
graficas.addGrafica(new Capa(16,0,16));
graficas.addGrafica(new Capa(17,0,17));
graficas.addGrafica(new Capa(18,0,18));
graficas.addGrafica(new Capa(19,0,19));
graficas.addGrafica(new Capa(20,0,20));
graficas.addGrafica(new Capa(21,0,21));
graficas.addGrafica(new Capa(22,0,22));
graficas.addGrafica(new Capa(23,0,23));
graficas.addGrafica(new Capa(24,0,24));
graficas.addGrafica(new Capa(25,0,25));
graficas.addGrafica(new Capa(26,0,26));
graficas.addGrafica(new Capa(27,0,27));
graficas.addGrafica(new Capa(28,0,28));
graficas.addGrafica(new Capa(29,0,29));
graficas.addGrafica(new Capa(30,0,30));
graficas.addGrafica(new Capa(31,0,31));
graficas.addGrafica(new Capa(32,0,32));
graficas.addGrafica(new Capa(33,0,33));
graficas.addGrafica(new Capa(34,0,34));
graficas.addGrafica(new Capa(35,0,35));
graficas.addGrafica(new Capa(36,0,36));
graficas.addGrafica(new Capa(37,0,37));
graficas.addGrafica(new Capa(38,0,38));
graficas.addGrafica(new Capa(39,0,39));
graficas.addGrafica(new Capa(40,0,40));
graficas.addGrafica(new Capa(41,0,41));
graficas.addGrafica(new Capa(42,0,42));
graficas.addGrafica(new Capa(43,0,43));
graficas.addGrafica(new Capa(44,0,44));

this._graficas = graficas;

//id referenciado en grafica y condiciones
Conceptos conceptos = new Conceptos();
conceptos.addConcepto(new Personaje(0,"Hugo ", "y trabajo en el molino "));
conceptos.addConcepto(new Personaje(1,"Paco ","y esos son mis manzanos "));
conceptos.addConcepto(new Personaje(2,"Luis ","el piloto de la avioneta "));
conceptos.addConcepto(new Personaje(3,"Kenita ","y ese es mi huerto de lechugas "));
conceptos.addConcepto(new Personaje(4,"Marlene ","y ese es mi campo de maíz "));
conceptos.addConcepto(new Personaje(5,"Michelle ","la dueña del almacén "));
conceptos.addConcepto(new Personaje(6,"Olga ","la agricultora "));
conceptos.addConcepto(new Personaje(7,"Don Pepe ","el carnicero "));
conceptos.addConcepto(new Personaje(8,"Walter ","el panadero "));
conceptos.addConcepto(new Personaje(9,"Marcelo ","el apicultor "));
conceptos.addConcepto(new Personaje(10,"Marco ","el pescador "));
conceptos.addConcepto(new Personaje(11,"Butch ","el perro "));
conceptos.addConcepto(new Personaje(12,"Lola ","la gallina "));
conceptos.addConcepto(new Personaje(13,"Tortuga "," "));
conceptos.addConcepto(new Decorativo(14, "Agua ", " "));
conceptos.addConcepto(new Decorativo(15, "Solido01 ", " "));
conceptos.addConcepto(new Decorativo(16, "Solido02 ", " "));
conceptos.addConcepto(new Decorativo(17, "Solido03 ", " "));
conceptos.addConcepto(new Decorativo(18, "almacén ", "Este es un almacén "));
conceptos.addConcepto(new Decorativo(19, "árboles ", "Estos son muchos árboles "));
conceptos.addConcepto(new Decorativo(20, "avioneta ", "Esta es una avioneta "));
conceptos.addConcepto(new Decorativo(21, "bote ", "Este es el bote de un pescador "));
conceptos.addConcepto(new Decorativo(22, "huerto de lechugas ", "Este es un huerto de lechugas "));
conceptos.addConcepto(new Decorativo(23, "campo de maiz ", "Este es un campo de maíz "));
conceptos.addConcepto(new Decorativo(24, "campo de trigo ", "Este es un campo de trigo "));
conceptos.addConcepto(new Decorativo(25, "carnicería ", "Esta es una carnicería "));
conceptos.addConcepto(new Decorativo(26, "casa ", "Esta es tu casa "));
conceptos.addConcepto(new Decorativo(27, "colmenas de abejas ", "Estas son muchas colmenas de abejas "));
conceptos.addConcepto(new Decorativo(28, "manzanos ", "Estos son muchos manzanos "));
conceptos.addConcepto(new Decorativo(29, "molino ", "Este es un molino "));
conceptos.addConcepto(new Decorativo(30, "panadería ", "Esta es una panadería "));
conceptos.addConcepto(new Item(31, "Carne ", " "));
conceptos.addConcepto(new Item(32, "Dinero ", " "));
conceptos.addConcepto(new Item(33, "Harina ", " "));
conceptos.addConcepto(new Item(34, "Huevo ", " "));
conceptos.addConcepto(new Item(35, "Leche ", " "));
conceptos.addConcepto(new Item(36, "Lechuga ", " "));
conceptos.addConcepto(new Item(37, "Leña ", " "));
conceptos.addConcepto(new Item(38, "Maíz ", " "));
conceptos.addConcepto(new Item(39, "Manzana ", " "));
```

```
        conceptos.addConcepto(new Item(40, "Miel ", " "));
        conceptos.addConcepto(new Item(41, "Pan ", " "));
        conceptos.addConcepto(new Item(42, "Pescado ", " "));
        conceptos.addConcepto(new Item(43, "Queso ", " "));
        conceptos.addConcepto(new Item(44, "Trigo ", " "));

        this._conceptos = conceptos;


        //id referenciado en interacciones
        Interacciones interacciones = new Interacciones();

        interacciones.addInteraccion(new Describir(0, 0, null));
        condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(44);
        entradas = new ConceptoTransaccion[1];
        entradas[0] = new ConceptoTransaccion(44,1);
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(33,1);
        respuesta = new Items("Muchas gracias, como premio te regalo un saco de harina ","Pense que me traias la materia prima que necesito ",entradas,salidas);
        interacciones.addInteraccion(new Transar(1, 0, condiciones, "Si tienes la materia prima que necesito seleccionala desde tus ítems: ", "Necesito trigo para poder hacer la
harina ", respuesta));
        interacciones.addInteraccion(new Informar(2, 0, null, "En el molino trabajamos el trigo y lo convertimos en harina ", ""));

        interacciones.addInteraccion(new Describir(0, 1, null));
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(39,1);
        respuesta = new Items("Tengo tantas manzanas, te regalaré una "," ",null,salidas);
        interacciones.addInteraccion(new Transar(1, 1, null, " ", " ", respuesta));
        interacciones.addInteraccion(new Informar(2, 1, null, "Las frutas son ricas en vitaminas y fibras ", ""));

        interacciones.addInteraccion(new Describir(0, 2, null));
        condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(11);
        respuesta = new Items("Muchas gracias, me has librado de ese perro, ahora podemos salir de esta isla "," ",null,null);
        interacciones.addInteraccion(new Transar(1, 2, condiciones, " ", "Ese perro no me deja acercarme a mi avioneta ", respuesta));
        interacciones.addInteraccion(new Informar(2, 2, null, "Ganaste, has finalizado el juego", ""));

        interacciones.addInteraccion(new Describir(0, 3, null));
        condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(40);
        entradas = new ConceptoTransaccion[1];
        entradas[0] = new ConceptoTransaccion(40,1);
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(36,1);
        respuesta = new Items("Muchas gracias, como premio te daré una lechuga ","Pense que me traias un alimento azucarado ",entradas,salidas);
        interacciones.addInteraccion(new Transar(1, 3, condiciones, "Si tienes algo dulce damelo por favor: ", "Hace mucho tiempo que no como algo concentradamente dulce ",
respuesta));
        interacciones.addInteraccion(new Informar(2, 3, null, "La lechuga es una verdura rica en vitaminas ", ""));

        interacciones.addInteraccion(new Describir(0, 4, null));
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(38,1);
        respuesta = new Items("Tengo mucho maiz, compartiré un poco contigo "," ",null,salidas);
        interacciones.addInteraccion(new Transar(1, 4, null, " ", " ", respuesta));
        interacciones.addInteraccion(new Informar(2, 4, null, "El maiz es un cereal rico en carbohidratos ", ""));

        interacciones.addInteraccion(new Describir(0, 5, null));
        condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(12);
        entradas = new ConceptoTransaccion[1];
        entradas[0] = new ConceptoTransaccion(12,1);
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(34,1);
        respuesta = new Items("Muchas gracias, como premio te regalo un huevo ","Pensé que tenías mi gallina ",entradas,salidas);
        interacciones.addInteraccion(new Transar(1, 5, condiciones, "Si tienes mi gallina seleccionala desde tus ítems: ", "Mi gallina se ha perdido, su alimento es rico en
carbohidratos ", respuesta));
        interacciones.addInteraccion(new Informar(2, 5, null, "El huevo es un alimento rico en proteínas", ""));

        interacciones.addInteraccion(new Describir(0, 6, null));
        condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(39);
        entradas = new ConceptoTransaccion[1];
        entradas[0] = new ConceptoTransaccion(39,1);
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(44,1);
        respuesta = new Items("Muchas gracias, a cambio te regalo un poco de trigo ","Pense que me traias lo que quiero comer ",entradas,salidas);
        interacciones.addInteraccion(new Transar(1, 6, condiciones, "Si tienes un alimento dulce y rico en vitaminas entregamelo: ", "Quiero comer algo dulce rico en vitaminas y
fibras ", respuesta));
        interacciones.addInteraccion(new Informar(2, 6, null, "El trigo es un cereal que es rico en carbohidratos", ""));

        interacciones.addInteraccion(new Describir(0, 7, null));
        condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(41);
        entradas = new ConceptoTransaccion[1];
        entradas[0] = new ConceptoTransaccion(41,1);
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(31,1);
        respuesta = new Items("Muchas gracias, a cambio te regalo un trozo de carne ","Pense que me traias mi alimento rico en carbohidratos ",entradas,salidas);
        interacciones.addInteraccion(new Transar(1, 7, condiciones, "Si rico en carbohidratos entregamelo: ", "Quiero comer algo a base de cereales y rico en carbohidratos ",
respuesta));
        interacciones.addInteraccion(new Informar(2, 7, null, "La carne es un alimento rico en proteinas ", ""));
```

```
        interacciones.addInteraccion(new Describir(0, 8, null));
        condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(33);
        entradas = new ConceptoTransaccion[1];
        entradas[0] = new ConceptoTransaccion(33,1);
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(41,1);
        respuesta = new Items("Muchas gracias, como premio te regalo un pan ","Pense que me traias el ingrediente que necesito ",entradas,salidas);
        interacciones.addInteraccion(new Transar(1, 8, condiciones, "Si tienes el ingrediente que necesito seleccionalo desde tus ítems: ", "No tengo harina para hacer pan ",
respuesta));
        interacciones.addInteraccion(new Informar(2, 8, null, "El pan se hace con harina y la harina se hace con trigo ", ""));

        interacciones.addInteraccion(new Describir(0, 9, null));
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(40,1);
        respuesta = new Items("Tengo muchas colmenas, te regalaré un poco de miel "," ",null,salidas);
        interacciones.addInteraccion(new Transar(1, 9, null, " ", " ", respuesta));
        interacciones.addInteraccion(new Informar(2, 9, null, "La miel es un alimento azucarado, debes consumirla moderadamente ", ""));

        interacciones.addInteraccion(new Describir(0, 10, null));
        condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(36);
        entradas = new ConceptoTransaccion[1];
        entradas[0] = new ConceptoTransaccion(36,1);
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(42,1);
        respuesta = new Items("Muchas gracias, como premio te daré un pescado que recien picó mi anzuelo ","Pense que me traias mi almuerzo vegetariano rico en vitaminas
",entradas,salidas);
        interacciones.addInteraccion(new Transar(1, 10, condiciones, "Si tienes el alimento que quiero seleccionalo desde tus ítems: ", "Decidi equilibrar mi dieta y consumir mas
vegetales ", respuesta));
        interacciones.addInteraccion(new Informar(2, 10, null, "El pescado es un alimento rico en proteinas ", ""));

        interacciones.addInteraccion(new Describir(0, 11, null));
        condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(31);
        entradas = new ConceptoTransaccion[1];
        entradas[0] = new ConceptoTransaccion(31,1);
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(11,1);
        respuesta = new Items("Me atrapaste ","Esa comida no me gusta, yo me alimento de proteinas ",entradas,salidas);
        interacciones.addInteraccion(new Recoger(1, 11, condiciones, "Alimentame con alguno de tus items: ", "Si no me alimentas no podrás atraparme ", respuesta));

        interacciones.addInteraccion(new Describir(0, 12, null));
        condiciones = new iCondicion[1];
        condiciones[0]=new TenerConcepto(38);
        entradas = new ConceptoTransaccion[1];
        entradas[0] = new ConceptoTransaccion(38,1);
        salidas = new ConceptoTransaccion[1];
        salidas[0] = new ConceptoTransaccion(12,1);
        respuesta = new Items("Me atrapaste ","Esa comida no me gusta yo me alimento con cereales ",entradas,salidas);
        interacciones.addInteraccion(new Recoger(1, 12, condiciones, "Alimentame con alguno desde tus ítems: ", "Si no me alimentas no podrás atraparme ", respuesta));

        interacciones.addInteraccion(new Describir(0, 18, null));
        interacciones.addInteraccion(new Describir(0, 19, null));
        interacciones.addInteraccion(new Describir(0, 20, null));
        interacciones.addInteraccion(new Describir(0, 21, null));
        interacciones.addInteraccion(new Describir(0, 22, null));
        interacciones.addInteraccion(new Describir(0, 23, null));
        interacciones.addInteraccion(new Describir(0, 24, null));
        interacciones.addInteraccion(new Describir(0, 25, null));
        interacciones.addInteraccion(new Describir(0, 26, null));
        interacciones.addInteraccion(new Describir(0, 27, null));
        interacciones.addInteraccion(new Describir(0, 28, null));
        interacciones.addInteraccion(new Describir(0, 29, null));
        interacciones.addInteraccion(new Describir(0, 30, null));
        this._interacciones = interacciones;

    }
```