Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Aplicación Android para la práctica de verbos compuestos del idioma inglés

Reporte Final de Proyecto Terminal

Alumno: Julio César Castillo García 208303917

Trimestre 2013 Invierno

Asesora: Dra. María Lizbeth Gallardo López
Profesor-Investigador
Departamento de Sistemas

Índice

1. Introducción

En la actualidad el aprendizaje del idioma inglés se ha convertido en una necesidad dado que es empleado para la comunicación de varias áreas del conocimiento y el desarrollo humano. Prácticamente se podría afirmar que debido a la globalización, es el idioma hablado en todo el mundo hoy día. A pesar de ello, en muchas ocasiones el aprendizaje de un idioma puede tornarse complicado, ya sea por la forma en que éste es enseñado, la falta de práctica o incluso la apatía de algunas personas hacia la compresión de nuevos idiomas. Es sabido que la memoria humana puede ser asociativa; por ejemplo, asociar una imagen a las frases de un idioma puede ayudar a las personas a recordar la frase, además de ayudarle a emplear la frase en el contexto adecuado.

Una de las características que diferencian el inglés de otros idiomas y que además le añade riqueza, son los verbos compuestos o *phrasal verbs*. Se trata de verbos que cambian de significado cuando se les agrega una partícula (como una preposiciones); por ejemplo, la expresión *I'm looking at you* se interpreta al castellano como "mirar a alguien", mientras que la expresión *I'm looking for you* tiene una connotación de "estar buscando a alguien". El hecho de que el significado de un verbo compuesto cambie con solo ubicarse antes de una u otra partícula es uno de los obstáculos con los que un estudiante del idioma inglés puede encontrarse; por lo anterior surge la necesidad de aprender el empleo de los *phrasal verbs* en el contexto adecuado.

En años recientes los teléfonos móviles así como los teléfonos inteligentes[1] están más al alcance de las personas, lo cual se ha reflejado en un incremento en el desarrollo y uso de aplicaciones para este tipo de dispositivos. Así mismo, los teléfonos inteligentes que incluyen un sistema operativo como Android[2] están aumentando en su número de usuarios, en gran parte debido a la robustez del sistema y a la creciente aparición de nuevas aplicaciones para esta plataforma.

2. Objetivos

2.1. Objetivo General

Diseñar e implementar una aplicación tipo juego para dispositivo móvil con sistema operativo Android para la práctica de verbos compuestos del idioma inglés.

[1] Teléfono móvil construido sobre una plataforma informática móvil, que puede usarse como una computadora de bolsillo y cuenta con más conectividad que uno convencional.

[2] Android es un sistema operativo móvil basado en Linux, enfocado para ser utilizado en dispositivos móviles. Liberado bajo licencia de uso de tipo GNU GPL.

2.2. Objetivos Específicos

- Diseñar e implementar el mini juego Throwing verbs.
- Diseñar e implementar el mini juego Phrases Soup.
- Diseñar e implementar el mini juego Special.
- Realizar pruebas de funcionamiento de cada uno de los mini juegos.
- Integrar los mini juegos dentro de la aplicación móvil para establecer la secuencia del juego.
- Implementar un módulo de escritorio que permita agregar más verbos e imágenes a la aplicación móvil.
- Realizar pruebas para verificar que los archivos generados por la aplicación de escritorio sean reconocidos automáticamente por la aplicación móvil al abrirla.

3. Antecedentes

Existe gran variedad de aplicaciones interactivas para la puesta en práctica de idiomas, por ejemplo The Oxford Picture Dictionary[3], mediante la asociación de imágenes busca que el usuario aprenda el vocabulario de diversas situaciones de la vida cotidiana y estudiantil, esta aplicación cuenta con un módulo de evaluación en el cual se puede poner a prueba lo aprendido lo cual se corresponde con el proyecto propuesto. La aplicación mencionada no cuenta explícitamente con algún apartado dedicado a los verbos compuestos del inglés.

La aplicación Learn and Play English[4] se acerca más a los usuarios de dispositivos móviles que buscan aprender y poner en práctica los idiomas, pero hasta un nivel básico, y no incluye un módulo específico para los verbos compuestos del inglés. El desarrollo propuesto incluye además una aplicación de escritorio que permitirá hacer crecer y enriquecer el vocabulario con el que la aplicación cuenta, lo cual no es posible en esta aplicación.

La aplicación Phrasal Verbs[5], que está dedicada completamente a los verbos compuestos del inglés dando su significado y proporcionando pequeños ejercicios para evaluar el aprendizaje de quien lo usa; en este aspecto existe concordancia con el proyecto que proponemos; sin embargo, los ejercicios están conformados por asociaciones de los verbos compuestos a sus significados en inglés o sinónimos de las palabras en inglés y no incluye ningún tipo de imágenes que permitan reforzar la interpretación y el uso de los verbos compuestos en el contexto adecuado.

[3] Es una aplicación de escritorio desarrollada por la Universidad de Oxford.
[4] Aplicación móvil desarrollada por la compañía DOMOsoft.
[5] Es una aplicación para plataforma Android desarrollado por Ángel Ruíz Alonso.

4. Justificación

En mi experiencia como estudiante del idioma inglés y como se mencionó anteriormente, el aprendizaje de los verbos compuestos característicos de este idioma puede convertirse en una tarea muy complicada: 1) por la gran cantidad que existe de éstos, 2) por los significados tan variados que pueden tener según la combinación entre las palabras y principalmente 3) por la falta de práctica de los mismos.

Algunas herramientas utilizan imágenes para asociarlas con palabras del inglés o cuentan con pequeños ejercicios para poner en práctica los verbos compuestos, pero no se enfocan totalmente en la puesta en práctica de los verbos empleando imágenes lo suficientemente descriptivas; tampoco se observa que las acciones sean asociadas con las palabras que conforman un verbo compuesto.

El presente proyecto considera además que en un ambiente de juego se hará mucho más atractivo practicar el uso de los verbos compuestos del inglés, a través de la asociación de los verbos con las imágenes que describen las acciones dentro de un contexto determinado. La aplicación estará formada por una serie de mini juegos que operarán por niveles de dificultad; además, en el largo plazo, se podría integrar una funcionalidad que permita escuchar la pronunciación de cada uno de los verbos incluidos en la aplicación.

Cabe señalar que la herramienta que proponemos podrá ser usada no solo por estudiantes del Centro de Lenguas Extranjeras de la UAM-A, sino por cualquier persona interesada en reforzar su conocimiento de los verbos compuestos del idioma inglés por medio de la práctica.

Se ha optado por emplear una plataforma para dispositivos móviles con sistema operativo Android, dada la expansión de este mercado y el incremento de dispositivos con este tipo de plataforma. El uso de dispositivos móviles permitirá que un mayor número de personas pueda acceder a los mini juegos en cualquier momento y en cualquier lugar, sin necesidad de una computadora o un libro con el listado de verbos compuestos para comenzar con su práctica.

5. Desarrollo del Proyecto

5.1. Metodología

La solución toma como base el modelo MVC (Modelo-Vista-Controlador) [1], que define 3 tipos de objetos: El modelo, que es el objeto de la aplicación; la Vista que es la representación en pantalla, y el Controlador que define la forma en que la interfaz de usuario reacciona a las entradas del usuario. Esto es, el modelo MVC desacopla los elementos mencionados para incrementar la flexibilidad de una aplicación y la reutilización de los elementos en otras aplicaciones.

Mario Zechner en su libro Beggining Android Games [2] propone un conjunto de clases que conforman un *framework*[6] para la implementación de juegos de cualquier tipo, el cual se empleó como base, utilizando aquellos elementos requeridos en la implementación de nuestra aplicación. En efecto se incluyen clases que permiten administrar la vista y el control de entradas del usuario, y hacia la lógica de la aplicación, el modelo. Una razón para emplear esta *framework*, es su facilidad de uso a nivel de la implementación y de la forma en que se adecuó como solución en la implementación de nuestra aplicación. De esta forma, en primera instancia se presentan las clases de la lógica del juego. Posteriormente detallamos las clases utilizadas del *framework*.

5.2. Arquitectura del Software

De acuerdo al modelo vista controlador, considerando el hecho de que estamos desacoplando la lógica del negocio y la interfaz o la cara hacia el usuario, se tiene el diagrama en la Figura 1.

[6] Un framework es un conjunto de bibliotecass y módulos de software, entre otras herramientas, que provee funcionalidades definidas para facilitar el proceso desarrollo de software.

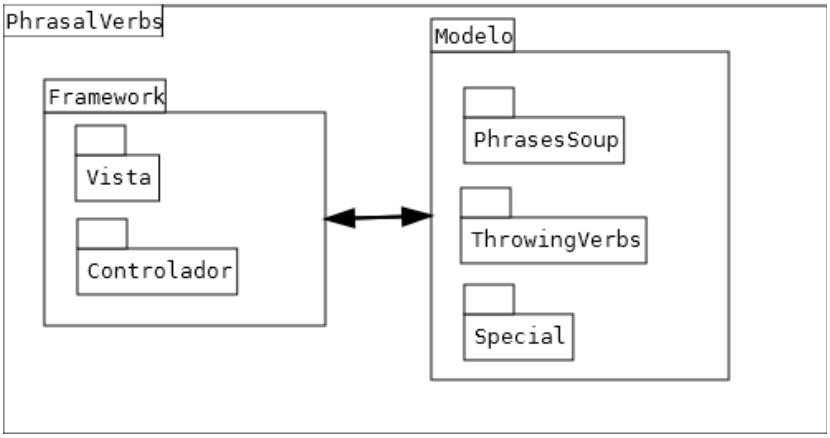

Figura 1. Arquitectura del Sistema.

Tomando en cuenta que se trata de una arquitectura stand-alone ya que se da servicio a un solo usuario sin requerir de conexión a red alguna ni de algún otro componente externo de la aplicación móvil durante su ejecución [3], se observa que la vista y el controlador están contenidos dentro del *framework*. Así, el *framework* contiene las clases gráficas diseñadas para encargarse de la representación en pantalla de todos los elementos gráficos del juego. El controlador, incluido dentro del *framework*, se comunica con el modelo, en este caso `PhrasalVerbs`, el cual contiene las clases de cada uno de los mini-juegos. De esta forma, el controlador de acuerdo a la información que recibe del usuario, se comunica con el modelo, el cual en respuesta envía el estado de dicho movimiento. Finalmente el *framework,* a través de la vista, presenta al usuario el estado visual de la aplicación.

5.3. Diagrama de Casos de Uso General

Considerando directamente los casos de uso relacionados con las funcionalidades que proporcionará la aplicación mostramos en la Figura 2 el diagrama de casos de uso general del sistema.

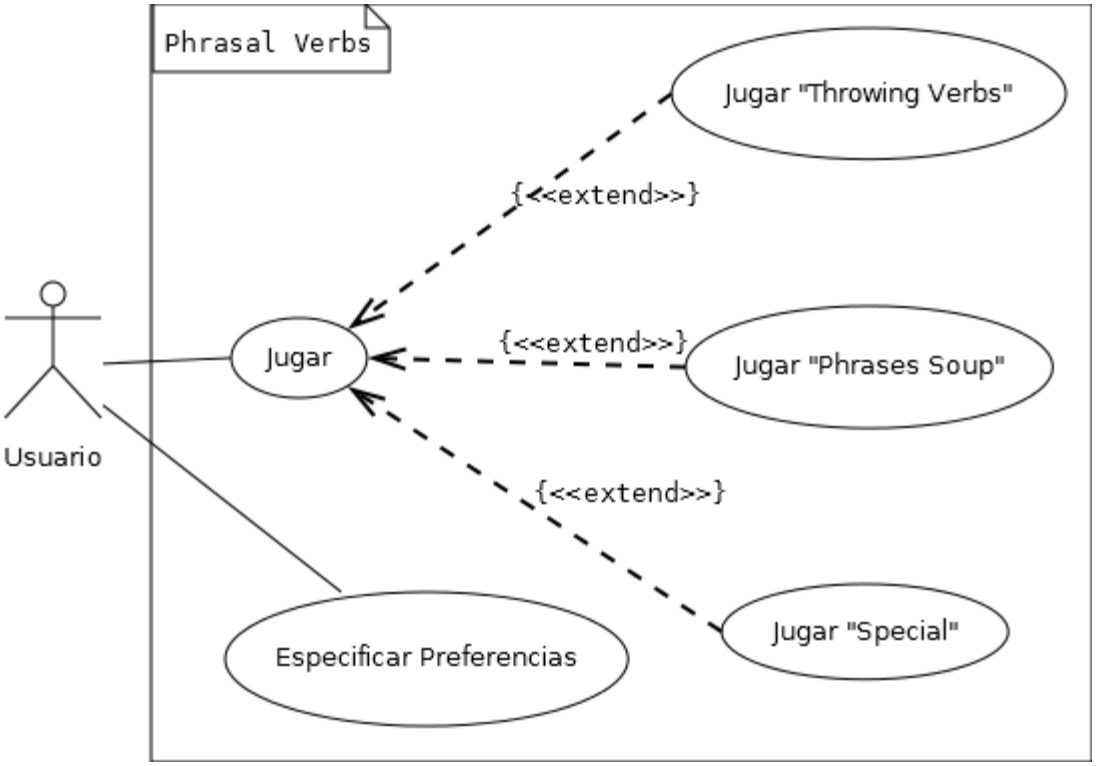

Figura 2. Diagrama de Casos de Uso General.

Cada módulo o mini-juego heredará directamente de Jugar, debido a que compartirán varias características, tales como: la aleatoriedad en que las imágenes serán presentadas al usuario, el manejo de puntaje, movimientos, etc.

En lo particular, cada mini-juego se encargará de implementar la lógica de cada uno de los mini-juegos de tal forma que, aunque todos compartan varias características, cada uno tiene características particulares, a saber: los elementos que se presentarán al usuario, la evaluación de si un movimiento es correcto; entre otros.

Especificar preferencias permitirá modificar la reproducción de sonidos, y aunque no lo haga directamente el usuario, de acuerdo a los mini-juegos desbloqueados previamente, el avance será guardado dentro de las preferencias, para que al reingresar a la aplicación se desbloqueen los botones que permitan acceder a los mini-juegos ya desbloqueados con anterioridad.

5.4. Caso de Uso de Texto

Para explicar el proceso de análisis y diseño de los mini-juegos, se tomará como ejemplo al mini-juego Throwing Verbs con dificultad principiante, para el cual se detallará su caso de uso de texto y su diagrama de secuencia; también se presenta el diagrama de clases de todo el modelo, y el diagrama de clases de la *framework*. Para mayor detalle sobre los casos de uso de texto y el resto de los artefactos de análisis y diseño remitirse al documento de diseño de la aplicación.

MP1: Mini-juego Throwing Verbs - Principiante

Descripción: Caso de uso que muestra el funcionamiento del juego Throwing Verbs en nivel Principiante.

Actores
Primario: Jugador.
Secundario: N/A.

Disparador: El actor primario solicita comenzar el mini-juego Throwing Verbs.
Precondición: El actor primario está comenzando el juego.
Pos condición: Se indica la puntuación del actor primario obtenida al término del juego.

Escenario Principal:

1. El sistema carga el mini-juego Throwing Verbs.
2. El sistema selecciona 7 de los 10 verbos compuestos disponibles para el nivel principiante.
3. El sistema elige aleatoriamente el orden en que los 7 verbos compuestos serán presentados al actor primario.
4. De acuerdo al orden elegido para los verbos compuestos, se toma uno de estos para presentarla al actor primario.
5. El sistema elige 5 imágenes aleatoriamente de todas con las que se cuenta y considera como sexta imagen la que corresponde al verbo compuesto seleccionado en el punto 3.
6. El sistema selecciona aleatoriamente el orden en que las 6 imágenes serán presentadas junto con el verbo compuesto al actor principal.
7. El sistema comienza un contador de segundos empezando desde 0.
8. El actor principal elige una de las 6 imágenes para asignarla al verbo compuesto que presenta el sistema, y la lanza hacia donde se encuentra posicionado el verbo compuesto. No han transcurrido más de 15 segundos desde el punto 7.
9. La imagen corresponde con el identificador del verbo compuesto, el sistema aumenta el puntaje de respuestas correctas en 1 y continúa el flujo en el punto 4 hasta haber presentado al actor principal todos los verbos compuestos elegidos en el punto 3.
10. El sistema presenta al actor principal el puntaje de aciertos y movimientos incorrectos obtenidos.
11. El actor principal obtiene más de 80% de movimientos correctos, se desbloquea el mini-juego Phrases Soup en nivel Intermedio, caso de caso de uso de texto MI1.
12. Se termina la ejecución del mini-juego Throwing Verbs.

Flujo Alternativo:

8. Transcurrieron más de 15 segundos desde el punto 7, el sistema incrementa el puntaje de respuestas incorrectas.
   8.1. El flujo continúa en el punto 4 hasta haber presentado al actor principal todos los verbos compuestos elegidos en el punto 3. En caso contrario el flujo continúa en el punto 10.

11. El actor principal no obtiene más de 80% de movimientos correctos.
    11.1. El flujo continúa en el punto 12.

Requerimientos no funcionales:

El proceso de carga del mini-juego no debe tomar más de 2 minutos.

Tabla 1 Caso de Uso de Texto Throwing Verbs - Principiante.

5.5. Diagramas de Clases

Los casos de uso de texto de la aplicación *Phrasal Verbs* permitieron identificar las clases que representan la lógica de cada uno de los mini-juegos; es decir, nuestro modelo. A continuación se describen las clases.

Se emplea la clase abstracta `JuegoPV` (juego *Phrasal Verbs*), para que de forma generalizada los métodos que define puedan ser llamados por las clases controladoras; entonces no importará el mini-juego del que se trate sino que a través de esta clase definiremos como se comunicará el *framework* con nuestro modelo, incluso en caso de que en trabajos futuros se desee agregar más mini-juegos, bastará con heredar de `JuegoPV` para conocer la forma de comunicación con el modelo. Ya que existen funcionalidades compartidas en los mini-juegos, como el ordenamiento aleatorio de imágenes, verbos compuestos o enunciados, la clase abstracta nos permite definir comportamientos para las clases que hereden de `JuegoPV`. De esta forma, las clases `PhrasesSoup`, `Special` y `ThrowingVerbs` que heredan de `JuegoPV`, definen finalmente el comportamiento particular de cada mini-juego, y de acuerdo a las reglas de cada juego, se indicará a través de la clase abstracta si las jugadas[7] del usuario son correctas.

Las relaciones entre las clases que forman el modelo se muestran en la Figura 3.

[7] En la aplicación Phrasal Verbs, se define una jugada como un movimiento realizado por el usuario, con el fin de asignar una imagen a un verbo compuesto, una imagen a un enunciado, etc. la cual tendrá como respuesta si la jugada es correcta o incorrecta.

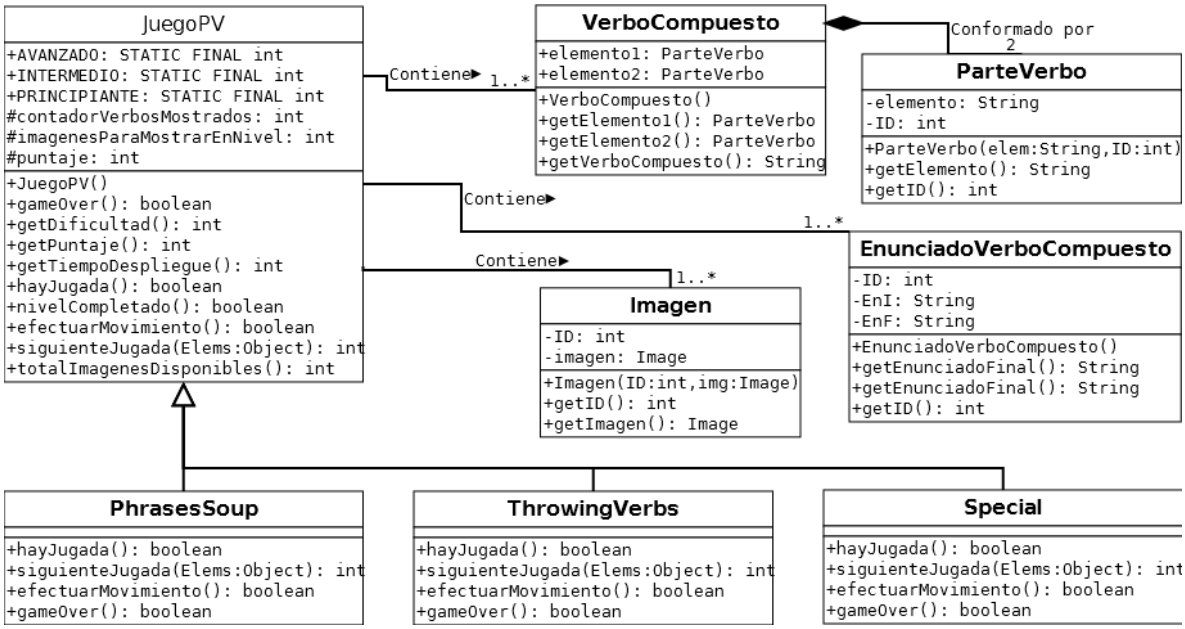

Figura 3 Diagrama de Clases del Modelo.

En primera instancia, se aprecia la implementación de la lógica de los mini-juegos al heredar de `JuegoPV`.

La clase `Imagen`, almacena el identificador de la imagen así como la referencia a la imagen. La clase `JuegoPV`, al definir a un mini-juego, siempre de acuerdo a las reglas del modelo, contiene una ó varias clases de tipo `Imagen` para poder asociarlas con verbos compuestos.

La clase `ParteVerbo` almacenará cada una de las palabras de los verbos compuestos. Para efectos de *Phrasal Verbs* se considerarán verbos compuestos por 2 palabras o partículas; por lo cual, `ParteVerbo` contendrá la palabra e identificador de la palabra para su uso en los mini-juegos.

La clase `VerboVompuesto` está conformada por 2 clases de tipo `ParteVerbo`, con lo cual se genera un verbo compuesto, una de las bases de todo *Phrasal Verbs*. `JuegoPV` al igual que con la clase `Imagen`, contiene de una a varias clases de este tipo, lo cual corresponderá con las características de cada mini-juego.

La clase `EnunciadoVerboCompuesto` se emplea en el mini-juego Special almacenando los enunciados que deben formarse con los verbos compuestos, haciendo referencia al identificador del verbo e imagen a los que pertenecen. La clase `JuegoPV`, contiene de una a varias clases de este tipo, aunque por ahora sea empleado únicamente por el mini-juego Special.

5.6. Diagrama de Clases del Framework para Phrasal Verbs

Como se mencionó al inicio de la sección 5, se empleó como base el *framework* para la implementación de juegos; ahora, explicaremos las clases elegidas de acuerdo a los requerimientos de *Phrasal Verbs*. El *framework* incluye clases controladoras y clases para la vista; con ellas, se soporta el funcionamiento de de nuestra aplicación, ver Figura 4.

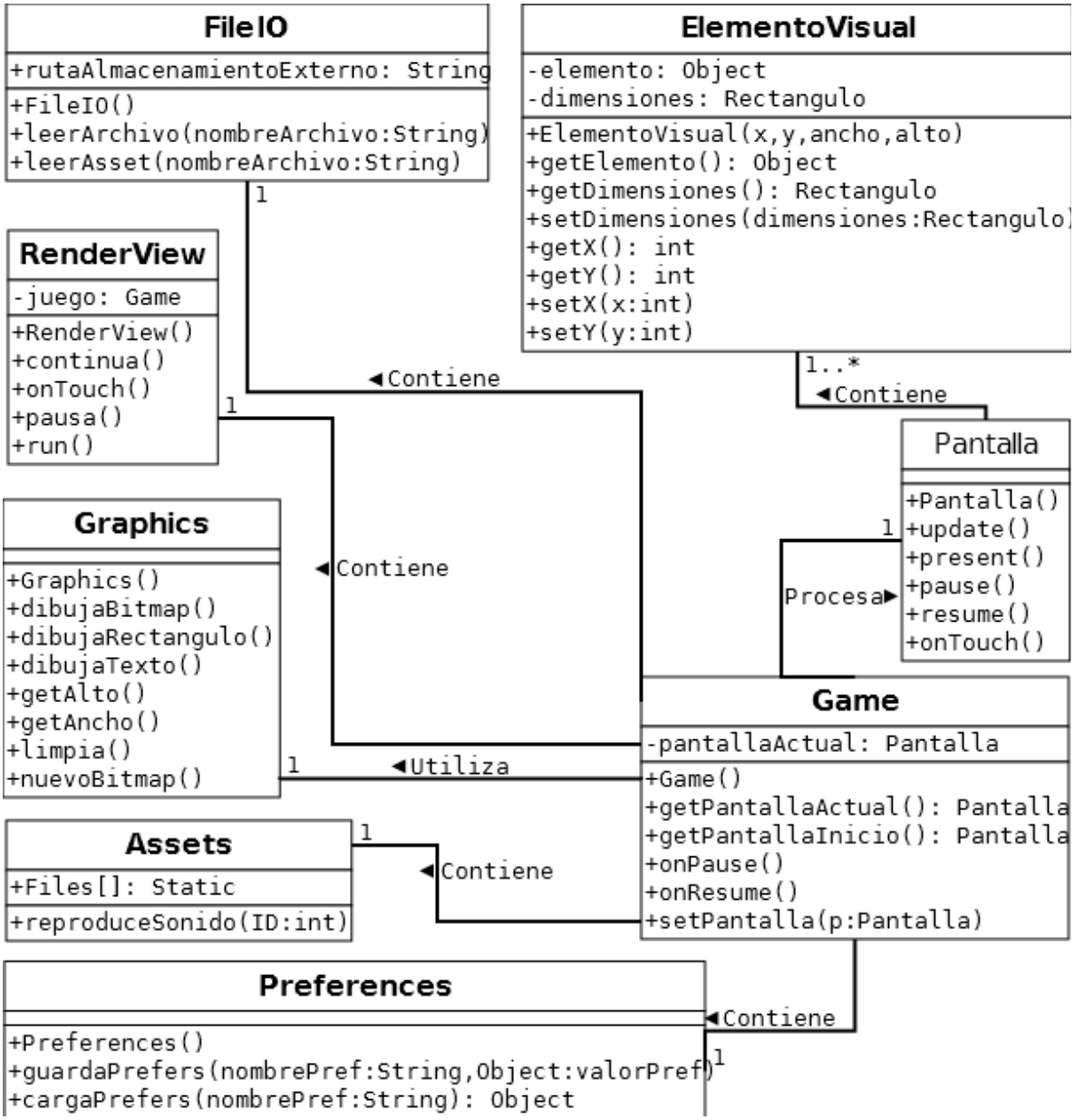

Figura 4 Diagrama de Clases del Framework para el juego.

La clase `Game`, engloba de forma general los componentes de cualquiera de los mini-juegos; esto es, la gestión de archivos de sonido y archivos de imágenes, la

gestión incluye: el acceso a archivos en medios de almacenamiento externo, el manejo de las pantallas del juego y la presentación del estado del juego en pantalla. Así, se puede notar que la clase `Game` contiene y utiliza cada una de las clases que conforman el *framework*: `RenderView`, `FileIO`, `Graphics`, `Assets` y `Preferences`.

La clase `Graphics` contiene todos los métodos que manejan todo lo que se dibuja en la pantalla así como el contenido de la misma. A través de esta clase se genera el movimiento de las imágenes y los verbos. La clase `Game` sólo contiene una clase de este tipo.

La clase `FileIO` nos es útil para manejar o administrar el uso de archivos dentro de la aplicación, ya sea de archivos dentro del ejecutable o directamente en la tarjeta de memoria del dispositivo móvil. De la misma forma que la clase `Graphics`, la clase `Game` sólo posee una clase de este tipo.

La clase `RenderView` controla el cambio de pantallas sobre el dispositivo móvil, así como el manejo de las pausas en el juego; además está encargada de indicar cuándo se ha reanudado una pantalla después de una pausa.

La clase `Assets` almacena los archivos: de imágenes, de texto y de sonidos que utilizará la aplicación, estos elementos son estáticos; es decir se crean una sola vez y perduran mientras la aplicación está en ejecución; de tal forma que no es necesario crear ejemplares de esta clase cada vez que se requiere tener acceso a estos elementos. Esta misma clase contiene también un método para la reproducción de clips de sonido[8], y se escogió un par de ellos para la aplicación. Sin embargo, el *framework* incluye una clase controladora de sonido para reproducir música de fondo, que para efectos de *Phrasal Verbs* no fue necesario incluir porque no se implementará con música de fondo.

La clase `Preferences` permite la lectura y escritura de preferencias, sólo basta con definir el nombre de la preferencia, por ejemplo: `sonidoActivado` y el valor que será asignado; durante la lectura se indica el nombre de la preferencia y es devuelto el valor que se tiene actualmente asignado dentro de las preferencias. Para detectar si el sonido se encuentra activado, se manejan valores de tipo `boolean`, de forma que se puede establecer a `true` si el sonido se encuentra activado y `false` si no se requiere la reproducción de sonido. La clase `Game` contiene una clase de este tipo para permitir la modificación y lectura de las preferencias de la aplicación.

[8] Se trata de archivos con sonido grabado con duración menor a 30 segundos y que por su tamaño pueden permanecer cargados en memoria durante toda la aplicación. Este tipo de clips, son utilizados para incluir efectos de sonido en cualquier tipo de juego.

`ElementoVisual` es la única clase que no forma parte del *framework*, y fue creada para facilitar la presentación de elementos visuales en pantalla; por lo tanto la clase Pantalla es la única que contiene elementos de tipo `ElementoVisual`, la cual se emplea para encapsular los elementos gráficos de la aplicación, como son: imágenes, botones, cuadros de texto, entre otros, de tal forma que el manejo de los elementos gráficos sea mucho más fácil.

Finalmente la clase `Pantalla`, que es una de las clases abstractas más importantes del *framework*, permite que la clase `Game` manipule cada una de las pantallas requeridas por el juego. Cuando la acción en el juego es realizar una pausa, o bien indicar que se deben dibujar en el dispositivo los elementos gráficos, la clase `Game` lo hace llamando a los métodos `pause()`, `present()` y `update()` definidos en la clase `Pantalla`. De esta forma, sin importar el tipo de pantalla que se trate, `Game` podrá manipular a través de `Pantalla`, el estado de las pantallas así como cambiar la pantalla actual. Las clases `PantallaCarga`, `PantallaPrincipal`, `PantallaPhrasesSoup`, `PantallaThrowingVerbs` y `PantallaSpecial` heredan de la clase `Pantalla`.

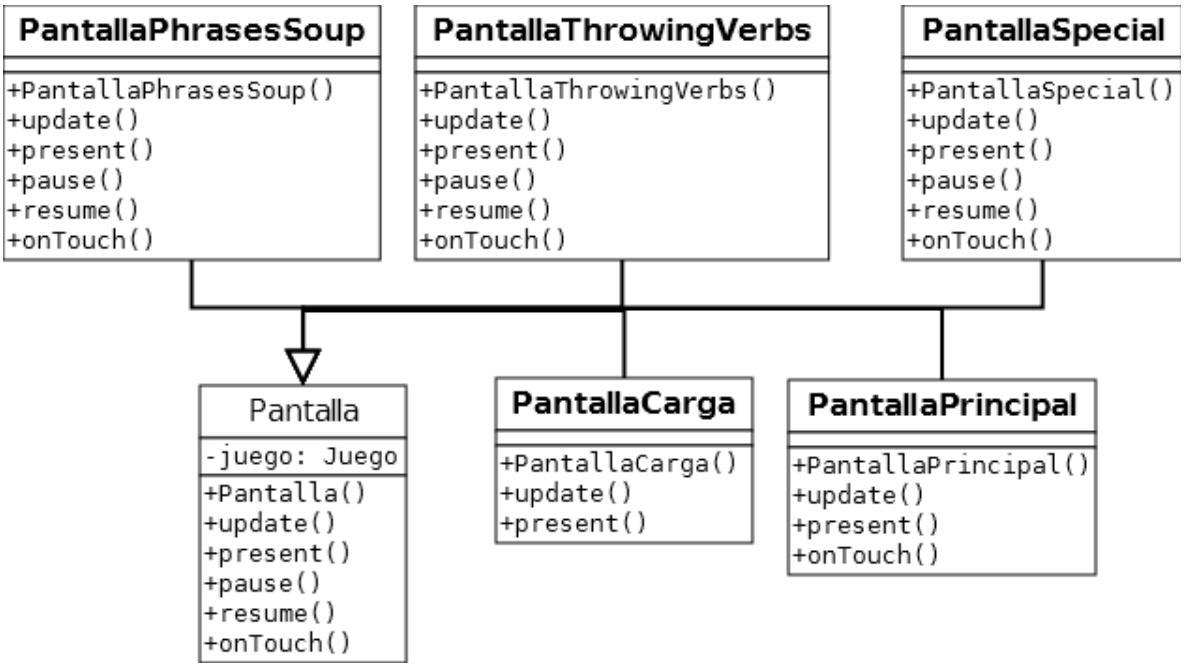

Figura 5 Implementación de la Clase Pantalla.

La clase `PantallaCarga` emplea una cantidad menor de métodos debido a que sólo se encarga de cargar en memoria todos los archivos de imagen, texto, sonido, etc. directamente a la clase `Assets` para su uso posterior.

Se detallarán más adelante los cambios de implementación y adecuaciones hechas al *framework* para el funcionamiento de la aplicación, y para mayor detalle de

la implementación del *framework* en general, referirse al libro de Mario Zechner "Beggining Android Games" [2].

5.7. Diagrama de Secuencia

Continuando con la explicación del mini-juego Throwing Verbs, presentamos el modelo de robustez en forma de diagrama de secuencia, donde se representa la interacción de los objetos frontera (clases de la vista), los objetos controlador (clases controladoras) y los objetos entidad (clases del modelo).

Se debe considerar que existe un ciclo que hace que la actividad se repita dependiendo de la dificultad del nivel con su consiguiente número de imágenes a desplegar pero por simplicidad no se incluye dentro del diagrama. Tampoco se consideran los casos en que el movimiento no es correcto, esto es, cuando no corresponda el verbo compuesto con la imagen a la que el usuario está asignando. Cuando esto suceda, se termina la jugada actual y si aún hay imágenes por mostrar se continúa con la siguiente.

El usuario inicia el mini-juego para lo cual se llaman los métodos de inicialización y el de `siguienteJugada()` para iniciar una jugada, la cual carga las imágenes y los verbos compuestos a utilizar, el usuario selecciona una imagen y se le indica si de acuerdo a las reglas del juego, éste es correcto, informando al usuario.

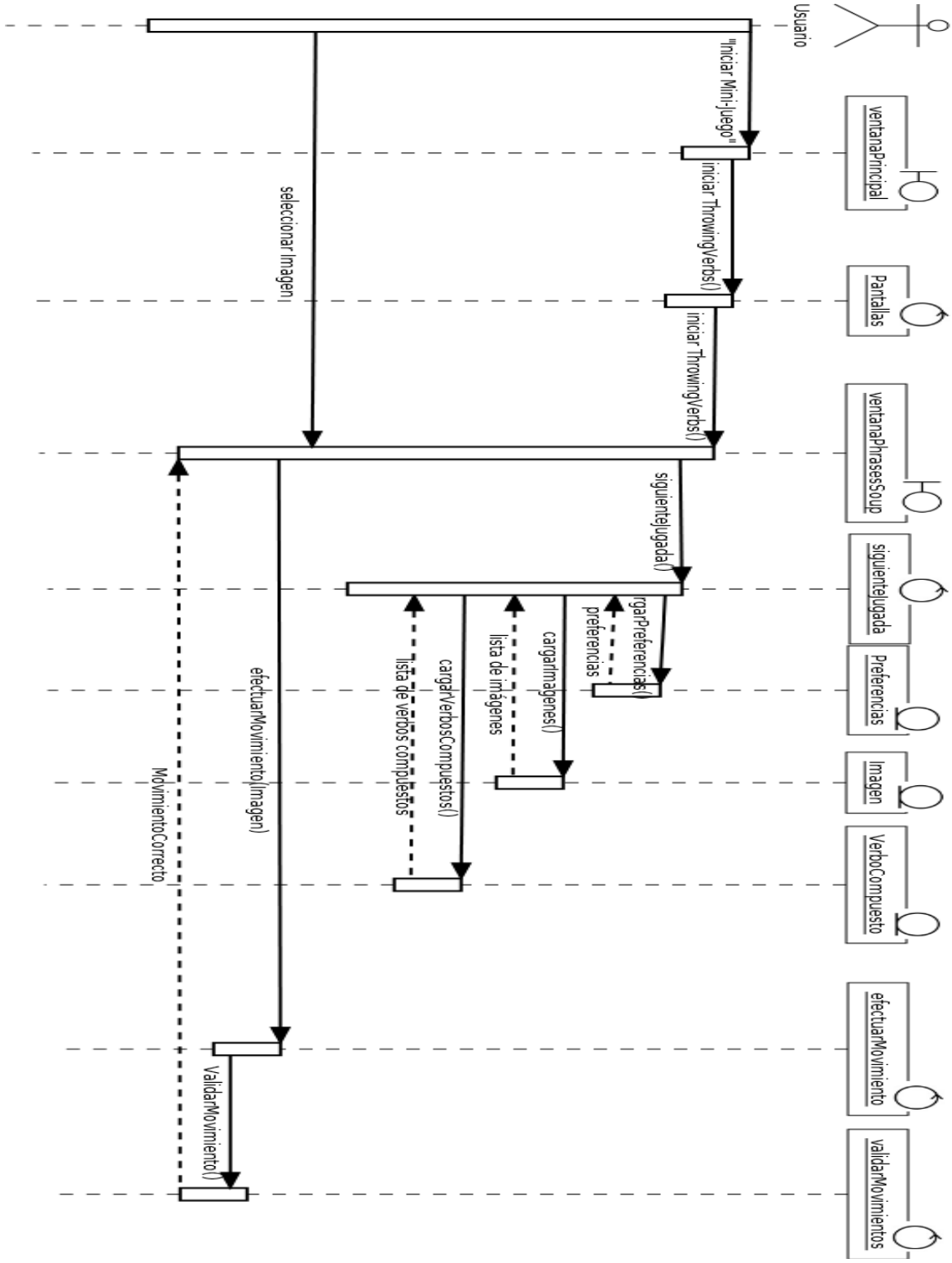

Figura 6. Diagrama de Secuencia de Throwing Verbs.

6. Implementación

6.1. Tecnología Empleada

El sistema está diseñado para funcionar en un Smartphone con sistema operativo Android con versión mínima 2.2 Froyo, aunque se recomienda versión 2.3 Gingerbread. El dispositivo deberá contar con pantalla táctil (no se requiere que la pantalla del dispositivo sea multitoque).

La resolución para la que se diseñó la aplicación es de 480 x 320 pixeles, aunque dispositivos de mayor tamaño funcionarán sin problemas debido a que la misma aplicación escala el tamaño de la imagen, con algo de pérdida de calidad en las imágenes por el estiramiento de la misma. Sin embargo, para dispositivos con una resolución inferior es posible que el escalamiento resulte en una pérdida mayor en la calidad de la imagen o que no sea visible para su adecuado funcionamiento.

La implementación de la aplicación móvil se realizó en el entorno de desarrollo Eclipse IDE 4.2.1 con el lenguaje de programación java, empleando el SDK[9] de Android. Además, se utilizó el plug-in Android Development Tools (ADT) disponible para su descarga desde el sitio web de Android [4], para integrar el desarrollo de aplicaciones para Android dentro de un proyecto de Eclipse.

6.2. Elementos de la Aplicación

La aplicación está conformada por una serie de elementos que son cargados al iniciar la aplicación y que se utilizan por cada uno de los mini-juegos.

Verbos Compuestos: Conformados de 2 palabras[10], se encuentran almacenados en un archivo de texto con un verbo compuesto por renglón con las palabras que lo conforman, separadas por un espacio. El archivo de texto se guarda dentro del ejecutable de la propia aplicación.

Enunciados de verbos compuestos: Enunciados para el uso del mini-juego Special que incluyen el verbo compuesto con el que está relacionado. También están almacenados dentro de un archivo de texto dentro del ejecutable de la aplicación. Ya que cada enunciado corresponde con un verbo compuesto, el orden en que se almacena el listado de enunciados es el mismo respecto al orden del listado de verbos compuestos al que debe ser asociado. Por este motivo, no se requiere incluir el identificador de los verbos compuesto o enunciados dentro de los archivos con listados.

[9] Kit de desarrollo de software por sus siglas en inglés.

[10] A pesar de que los verbos compuestos del idioma inglés pueden contener incluso 3 palabras, para efectos de este proyecto, sólo consideramos verbos compuestos formados por 2 palabras o partículas.

Imágenes: La colección de imágenes que están asociadas a cada uno de los verbos compuestos de la aplicación, tiene formato png, y dimensiones de 200 x 200 pixeles. Las imágenes empleadas en el proyecto fueron creadas por el Ilustrador Rubén de la Torre Hernández, mismas que están basadas en las ilustraciones del libro “Really Learn Phrasal Verbs” [5]. Las imágenes se almacenan dentro del ejecutable de la aplicación.

Sonidos: Considerando que los sonidos dentro de un videojuego pueden provocar un gran impacto en la experiencia del usuario [6], se decidió incluir sonido para indicar al usuario si la jugada efectuada es correcta o incorrecta. Es por ello que se hace uso de material en el sitio www.freesound.org, ya que tiene licencia Creative Commons [7] de tipo Attribution 3.0 Unported [8], la cual permite el uso, modificación del contenido e incluso su uso para fines comerciales.

Los archivos usados y sus respectivos créditos son:

- Sonido de respuesta correcta: “Game Sound Correct.wav” [9] del autor Bertrof [10].
- Sonido de respuesta incorrecta: “Error.wav” [11] del autor Autistic Lucario [12].
- Movimiento en mini-juego Throwing Verbs: “cartoon_siren_whistle_001.wav” [13] del autor soundscalpel.com [14].

Tipografías para los textos: Para la presentación de textos en pantalla, se empleó el tipo de fuente “vshandprinted” obtenida del sitio openfontlibrary.org y que en este caso proporciona licencia de uso “SIL Open Font License (OFL)” [15], la fuente fue creada por Thomas Von Strong [16]. La tipografía se almacena dentro del ejecutable de la aplicación y a través del *framework* se carga para su uso.

Android Activity: Dentro de una aplicación Android, siempre se debe contar con al menos una clase que herede de la clase `Activity` [17], ya que en general, esta clase permite la interacción con el usuario. De esta forma, en nuestra implementación definimos a la clase `Game`, la cual hereda de la clase `Activity`, la cual se carga automáticamente cuando el usuario inicia la aplicación *Phrasal Verbs* dentro del dispositivo móvil; posteriormente, `Game` administra las diferentes pantallas que se presentan al usuario.

6.3. Implementación del Modelo en Java

Siguiendo con nuestro ejemplo del mini-juego *Throwing Verbs* y de acuerdo a la Figura 3, tratamos a la clase `JuegoPV` como una clase abstracta dado que ya contiene datos y métodos compartidos por los demás mini-juegos. Contiene los datos que hacen referencia a los diversos tipos de dificultad, como son: PRINCIPIANTE, INTERMEDIO y AVANZADO. Además, la referencia a las imágenes que han sido mostradas durante una jugada, el total de imágenes que se muestran de acuerdo a la dificultad del nivel, y por supuesto la puntuación que ha acumulado el jugador durante la partida.

Durante la construcción del objeto de tipo `ThrowingVerbs` se lleva el control de los verbos que se muestran al usuario a través de arreglos de enteros, éstos se inicializan con los valores de los identificadores del juego, los cuales inician desde 0 hasta el número de imágenes que la aplicación tenga disponibles. Así mismo, dichos arreglos se reordenan aleatoriamente, de forma que no se tenga un orden fijo en la presentación de imágenes y verbos en pantalla.

La clase `JuegoPV` implementa los siguientes métodos públicos:

- getDificultad()
- getNivelNumerico()
- getTiempoDespliegue()
- getPuntaje()
- getImagenesParaMostrarEnNivel()
- totalImagenesDisponibles ()
- nivelCompletado()

getDificultad() Al ser creado un objeto de tipo `JuegoPV` se debe recibir un dato de tipo entero que está conformado por la variable DificultadesJuego y sus posibles valores son: PRINCIPIANTE (valor 1), INTERMEDIO (valor 2), AVANZADO (valor 3). Este método devuelve uno de estos valores, indicando cuál es la dificultad del mini-juego actual.

getNivelNumerico() se cuenta con una equivalencia numérica del mini-juego actual con la dificultad, esto, para que las clases puedan llevar el control de avance en la aplicación, el detalle se muestra en la Tabla 2. El mini-juego Special no tendrá una dificultad específica ya que podrá presentar al usuario todos los verbos compuestos e imágenes disponibles.

	Principiante	Intermedio	Avanzado
Phrases Soup	1	3	5
Throwing Verbs	2	4	6
Special	7		

Tabla 2 Nivel numérico de los mini-juegos.

getTiempoDespliegue() Indica el tiempo en segundos que se presenta al usuario una imagen o verbo, ya que se tiene un tiempo limitado para que el usuario realice una jugada. Inicialmente se habían propuesto tiempos distintos, pero durante las pruebas se ajustaron para proporcionar al usuario tiempo suficiente para responder. La Tabla 3 muestra los tiempos. Ya que el mini-juego no tendrá una dificultad definida, el tiempo de presentación de una pantalla, por la dificultad de éste mini-juego, será el mismo que de nivel Principiante.

Principiante	Intermedio	Avanzado
15 segundos	10 segundos	5 segundos

Tabla 3 Tiempo de duración en pantalla de una partida.

getPuntaje() Devuelve la puntuación acumulada en el mini-juego por cada jugada respondida correctamente. Internamente se tiene una variable tipo `protected` para que cada clase que implemente un mini-juego pueda aumentar la puntuación cuando sea necesario.

totalImagenesDisponibles() Devuelve el total de imágenes, verbos compuestos o partes de verbos compuestos que se presentan al usuario, o dicho de otra forma, el número de partidas que hay en el mini-juego. La cantidad de imágenes se muestran en la Tabla 4.

Principiante	Intermedio	Avanzado
10 imágenes	20 imágenes	30 imágenes

Tabla 4 Número de imágenes disponibles por dificultad de mini-juego .

Se hace notar, además, que para el mini-juego Special, todas las imágenes están disponibles, esto es, no tiene una dificultad asignada.

nivelCompletado() Indica mediante un valor boolean si una vez terminadas todas las partidas del mini-juego se ha logrado pasar al siguiente nivel, esto se hace siempre y cuando el número de respuestas correctas supere el 80% del total de jugadas.

Cada clase que herede de `JuegoPV` debe implementar los siguientes métodos:

- hayJugada()
- efectuarMovimiento(Object elemento1, Object elemento2, int ID)

- gameOver()
- siguienteJugada(Object[] listaElementos)

hayJugada() Debe calcular de acuerdo al número de imágenes, verbos o enunciados, si aún hay jugadas a proporcionar al usuario.

La clase `ThrowingVerbs` emplea una variable privada para llevar el contador de los verbos que se han mostrado al usuario, cada que se efectúa una jugada se incrementa, y aquí validamos si dicha variable no ha sobrepasado el límite de verbos que deben mostrarse por nivel.

efectuarMovimiento(Object elemento1, Object elemento2, int ID) De acuerdo a los objetos que reciba, los cuales pueden ser imágenes, objetos de tipo `ParteVerbo`, o `VerboCompuesto` y evalúa de acuerdo a las reglas de cada mini-juego si el movimiento es correcto, devolviendo un valor verdadero.

Al implementar nuestra clase `ThrowingVerbs`, comprobamos que elemento1 sea un objeto de tipo `Imagen` y que elemento2 sea de tipo `VerboCompuesto` y se procede a comparar los identificadores por medio de los métodos `getID()`, si son iguales procedemos a aumentar el puntaje y devolvemos un valor `true`, en otro caso simplemente se devuelve un valor `false`.

gameOver() Debe calcular si ya se han mostrado todas las partidas del mini-juego, con lo cual se habrá completado su ejecución.

Para implementar `ThrowingVerbs` nos valemos del método `hayJugada()` ya que de no haber más jugadas, el mini-juego habrá terminado.

siguienteJugada(Object[] listaElementos) Es uno de los métodos más importantes de la clase `JuegoPV`, pues se encarga de indicar el siguiente verbo compuesto a utilizar en la jugada (el cual es seleccionado aleatoriamente); además, precisa el orden de los objetos que se presentarán al usuario en la jugada, en este caso las imágenes. La Tabla 5 presenta el algoritmo para generar una nueva jugada en *Throwing Verbs*.

```
Si hayJugada(arregloParaAlmacenarOrdenImagenes[ ]) hacer
        identificador = arregloConOrdenDeVerbos[contadorDeVerbosMostrados]
        contadorDeVerbosMostrados = contadorDeVerbosMostrados + 1
        arregloDeOrdenamiento[ ]: Enteros
        arregloDeOrdenamiento[0] ← identificador
        Para i←1 hasta longitud(arregloDeOrdenamiento) hacer
                rellena arregloDeOrdenamiento[] con valores no repetidos de
arregloConOrdenDeVerbos[]
        reordenaAleatoriamente(arregloDeOrdenamiento[ ])
        Para cada elemento en arregloDeOrdenamiento[ ]
```

```
            arregloParaAlmacenarOrdenImagenes[ i ] ← Imagen con identificador
arregloDeOrdenamiento[ i ]
        devuelve identificador
```

Tabla 5 Pseudocódigo del algoritmo para generar nueva jugada en Throwing Verbs.

Se tiene un arreglo con los verbos que están disponibles en la aplicación `arregloConOrdenDeVerbos[]` y se hace uso de la variable `contadorDeVerbosMostrados` que lleva el conteo de la cantidad de verbos que se han mostrado al usuario. Considerando que existe una relación 1 a 1 entre un verbo compuesto específico y una imagen, se asigna a la variable `identificador` el valor del identificador de verbo compuesto que se presentará al usuario en esta jugada, de forma que dentro de la lista de identificadores siempre se incluya el que corresponde con el verbo compuesto (y por consiguiente con el de su imagen), posteriormente se agregan más identificadores de verbos compuestos para contar con los 4 elementos restantes que serán presentados al usuario. Así, aunque nos hayamos basado en los identificadores de verbos compuestos para rellenar el listado, en realidad cuando el arreglo `arregloParaAlmacenarOrdenImagenes[]` sea devuelto, tendremos los identificadores de las imágenes que serán vistas por el usuario en la partida. Finalmente, se reordena el arreglo aleatoriamente y devolvemos el identificador de ésta jugada para presentar la información al usuario (en caso de que ya no haya jugadas disponibles su valor será -1). Al arreglo se le asignarán los objetos a presentar al usuario por partida de acuerdo a los identificadores contenidos en `arregloDeOrdenamiento[]`; los objetos pueden ser de tipo `Imagen`, `ParteVerbo` o `EnunciadoVerboCompuesto`.

6.4. Manejo de Pantallas

Hasta ahora se ha mencionado la implementación del modelo para el mini-juego *Throwing Verbs*; ahora bien, para presentar al usuario el contenido de lo que sucede en la aplicación, el *framework* hace uso de la clase `RenderView` como se observa en la Figura 4, que se encarga de realizar el dibujado de todos los elementos de la aplicación en la pantalla, esta clase implementa la interfaz `OnTouchListener` para detectar cualquier evento de toque[11] sobre la pantalla, pero a diferencia de la propuesta de Mario Zechner [2] quien en su *framework* de juegos implementa un `Listener` para eventos de entrada para efectos de rendimiento, nosotros manejamos el famoso método `onTouch()` que recibe como parámetros, entre otras cosas, un objeto de la clase `MotionEvent` que contiene información del evento de toque por ejemplo si se ha presionado, si se está moviendo o si se ha levantado el dedo de la pantalla, además de la posición en la pantalla donde se realizó el toque, entre otras cosas. Esto con la finalidad de simplificar el manejo de los eventos de toque del usuario.

[11] Pulsación efectuada por el usuario sobre la pantalla del dispositivo, en general efectuada con el dedo.

También se controla la razón que existe entre la pantalla actual del dispositivo y el tamaño de pantalla recomendado, a saber: 480 x 320 pixeles, de modo que si se ejecuta la aplicación en un dispositivo con pantalla mayor a las dimensiones mencionadas, la imagen se alarga al tamaño adecuado del dispositivo. La escala se calcula dividiendo el tamaño del ancho de pantalla para el que se diseñó la aplicación (320 pixeles) entre el ancho de la pantalla del dispositivo actual. La Tabla 6 muestra el algoritmo para el redimensionamiento.

```
public boolean onTouch(View arg0, MotionEvent event) {
 //Ajuste de la posición de toque para el escalamiento en panta-
llas de otros tamaños
     event.setLocation(event.getX()*escalaX,event.getY()*escalaY);
 //Envío del evento de toque para manejo dentro de cada pantalla
     juego.getPantallaActual().onTouch(arg0, event);
 //Indicación de que el evento ha sido procesado con un valor true
     return true;
}
```

Tabla 6 Método `onTouch` con la operación de redimensionamiento de las coordenadas del evento.

Cada clase que hereda de `Pantalla` se encarga de dos aspectos: i) procesar los eventos de toque sobre la pantalla, y 2) envía los elementos que se deben dibujar en pantalla a la clase `Graphics`[12], los cuales varían dependiendo del estado del juego.

6.4.1. Clase PantallaCarga

La pantalla de carga se encarga de cargar en memoria todos los elementos de la aplicación ya sea desde el propio ejecutable, como desde una tarjeta de memoria externa en el dispositivo, si está disponible. Hace uso de la clase `Assets`, que contiene como atributos estáticos, una referencia a todos los objetos de imágenes, sonido, etc. que se utilizarán durante la ejecución de la aplicación.

Es importante mencionar que, `PantallaCarga` es la única clase desde la que se asignarán valores a las referencias de la clase `Assets`, en cualquier otra clase sólo se deben leer, para no provocar errores durante la ejecución de la aplicación por referencias inválidas o nulas. Al realizar la carga, se hace uso de la clase `Graphics` para convertir las imágenes desde el ejecutable de la aplicación; a continuación se muestra un fragmento del método `update()` de la clase `PantallaCarga`, desde el cual se realiza la carga de mencionada, directamente en la carpeta de `Assets/imagenesapp` de la aplicación, lo cual se aprecia en la Tabla 7.

[12] De esta forma la clase Pantalla, funge como clase abstracta hacia el usuario, recibiendo eventos del usuario y enviando las instrucciones para el dibujado de los elementos gráficos en pantalla.

```
public void update() {
     AndroidGraphics g = juego.getGraficos();
     Assets.fondo = g.nuevoBitmap("imagenesapp" + File.separator
+"fondo.png");
     Assets.pausa = g.nuevoBitmap("imagenesapp" + File.separator
+"pausa.png");
     Assets.quit = g.nuevoBitmap("imagenesapp" + File.separator
+"quit.png");
     Assets.mainscreen = g.nuevoBitmap("imagenesapp" + File.sepa-
rator +"mainscreen.png");
          Assets.about = g.nuevoBitmap("imagenesapp" +
File.separator +"about.png");
```

Tabla 7 Fragmento de código del método update() en la clase PantallaCarga

El archivo con el listado de verbos compuestos se encuentra ubicado dentro de la carpeta de aplicación `Assets` en la ruta `verbos/listaVerbos.dat` y contiene un verbo compuesto por línea y las dos palabras están separadas por un espacio como en el siguiente ejemplo de la Tabla 8:

```
Look After
Give Away
```

Tabla 8 Ejemplo del archivo listaVerbos.dat ubicado en los Assets de la aplicación.

El archivo con el listado de las oraciones o enunciados se encuentra ubicado dentro de la carpeta de aplicación `Assets` en la ruta `verbos/listaEnunciados.dat`, y contiene un formato especial, pues contiene los 2 enunciados separados por el símbolo %; en el primer enunciado se reemplaza el verbo compuesto por un símbolo * por cada verbo que lo compone; mientras que en el segundo enunciado se incluye el verbo compuesto. El primer enunciado se emplea durante la primera parte de la jugada en el mini-juego Special, para mostrar el enunciado en pantalla, dejando espacios en blanco en la posición donde debe ubicarse al verbo compuesto, de forma que el usuario pueda elegir el verbo compuesto que mejor complemente en enunciado. El segundo enunciado se emplea para que una vez que el usuario haya elegido el verbo compuesto que corresponda con el enunciado, se presente en pantalla el enunciado con el verbo compuesto conjugado y en su posición correcta, de forma que se pueda evaluar si la jugada fue correcta o incorrecta. Se muestra en la Tabla 9 un ejemplo con los enunciados que corresponden al listado de verbos anterior.

```
I have to * * my young sister % I have to look after my young sister
I will * * my old computer % I will give away my old computer
```

Tabla 9 Ejemplo del archivo con formato listaEnunciados.dat ubicado en los Assets de la aplicación.

La carga de verbos y enunciados se realiza simultáneamente y se van creando los respectivos ejemplares de las clases de nuestro modelo, a saber: `ParteVerbo`, `VerboCompuesto` y `EnunciadoVerboCompuesto`. A cada uno de estos ejemplares se les asigna el mismo identificador que comienza desde 0, de tal forma que no sea necesario guardar un identificador en cada archivo; sino que, basados en el orden en que están guardados los verbos, los enunciados y numerados los archivos de imágenes, se asigne un identificador incremental, el cual hará referencia a su posición dentro del arreglo de verbos, imágenes y enunciados de la clase `Assets`.

Los arreglos son inicializados con un tamaño dependiente de la constante `tamanoMaximoArreglos` que tiene un valor de trescientos, este valor determina la cantidad máxima de elementos (verbos, imágenes y enunciados) que soporta la aplicación. La Tabla 10 muestra un fragmento de código para la carga de verbos y enunciados, si se detecta un error durante la carga, se genera una excepción en tiempo de ejecución y no se podrá continuar con el juego, ya que los elementos verbos, imágenes y enunciados son lo más importantes para los mini-juegos.

```
BufferedReader brVerbos = new BufferedReader(new
InputStreamReader(juego.getIO().leerAsset("verbos" +
File.separator + "listaVerbos.dat")));
BufferedReader brEnunciados = new BufferedReader(new
InputStreamReader(juego.getIO().leerAsset("verbos" +
File.separator + "listaOraciones.dat")));
String cadena;
int identificador = 0;
Assets.verbos = new VerboCompuesto[Assets.tamanoMaximoArreglos];
Assets.enunciados = new EnunciadoVerboCompuesto[Assets.tamanoMaxi-
moArreglos];
try {
      while((cadena = brVerbos.readLine()) != null){
      StringTokenizer tokens = new StringTokenizer(cadena);
      Assets.verbos[identificador] = new VerboCompuesto(new
ParteVerbo(identificador,tokens.nextToken()), new
ParteVerbo(identificador,tokens.nextToken()));
      StringTokenizer tokensOraciones = new
StringTokenizer(brEnunciados.readLine(),"%");
      Assets.enunciados[identificador] = new EnunciadoVerboCompues-
to(identificador, tokensOraciones.nextToken().trim(), tokensOra-
ciones.nextToken().trim());
      identificador++;
      }
} catch (IOException e) {
      throw new RuntimeException("Error durante la carga de ver-
bos.");
}
```

Tabla 10 Fragmento de código para la carga de verbos y enunciados en la aplicación.

Las imágenes se encuentran ubicadas dentro de la carpeta `Assets/imagenes` de la aplicación, todas tienen formato `.png` numeradas desde el 1 hasta el número de elementos que contenga el listado de verbos. Para la carga de imágenes, el proceso es más sencillo ya que solo es necesario inicializar el arreglo de imágenes y cargarlas durante la creación de un nuevo ejemplar de tipo `Imagen`. El identificador se asigna de acuerdo al índice del ciclo que lo contiene, para obtener el nombre de archivo se suma 1 al índice, ya que las imágenes en la aplicación están nombradas comenzando desde 1; pero para acceder a los datos del arreglo el índice comienza desde 0.

```
Assets.imagenes[i] = new Imagen(i, g.nuevoBitmap("imagenes" +
File.separator + (i+1) + ".png"));
```

Tabla 11 Creación de los objeto Imagen durante la carga de los archivos de imágenes.

Durante la carga de elementos procedentes de la memoria externa del dispositivo móvil hacemos uso de la clase `FileIO`. Ver código de la Tabla 12.

```
InputStream inputVerbos = juego.getIO().leerArchivo ("phrasalverb-
sutility" + File.separator + "listaverbos.dat");
if(inputVerbos == null){
              //Indicación de que ningún archivo externo se leyó.
                return 0;
          }
BufferedReader brVerbos = new BufferedReader(new
InputStreamReader(inputVerbos));
```

Tabla 12 Carga de archivos desde la tarjeta externa de memoria empleando clases del framework.

Se comprueba el acceso al dispositivo externo, a través del objeto `inputVerbos`, si su valor, luego del llamado al método `leerArchivo`, es distinto de `null`, se puede proceder con la lectura asignándolo a un objeto de tipo `BufferedReader`.

Para la creación de los objetos desde los datos de una tarjeta externa de memoria, se sigue el mismo proceso que el de carga de datos desde el ejecutable, la única diferencia es que si en algún momento del proceso de carga se detecta algún error, se cancela toda la carga sin ningún tipo de comprobación, pues esto significa que el formato de los archivos que se requieren cargar no es correcto, y por lo tanto, no es necesario continuar leyendo datos de la tarjeta externa de memoria.

6.4.2. Clase PantallaThrowingVerbs

En nuestra implementación, cada mini-juego tiene su propia pantalla. Por ejemplo, desde la pantalla del mini-juego *Throwing Verbs* se llama a un ejemplar de la clase

`ThrowingVerbs` que contiene la lógica del juego, donde se destaca la dificultad del mismo. El ejemplar de la clase que implementa la lógica del mini-juego es almacenado en la variable `juegopv` de tal forma que al llamar a los métodos que se definieron para efectuar jugadas y comprobar si son correctas, podamos indicar los movimientos que realiza el usuario y así mismo, recibir la respuesta del modelo respecto a dichos movimientos.

El acceso a preferencias se realiza invocando uno de los métodos estáticos de la clase `Preferences`; cargamos la preferencia llamada `sonidoActivado` haciendo un llamado al método `cargaPreferenciasBoolean()` e indicamos el nombre de la preferencia recibiendo como valor de retorno la preferencia almacenada. Un fragmento del constructor de la clase se presenta en la Tabla 13.

```
public PantallaThrowingVerbs(AndroidGame juego) {
super(juego);
juegopv = new ThrowingVerbs(JuegoPV.DificultadesJuego.PRINCIPIAN-
TE);
modo = ModoJuego.Inicializando;
sonidoActivado = Preferencias.cargaPreferenciasBoolean(juego, "so-
nidoActivado");
```

Tabla 13 Fragmento de código del constructor de la clase PantallaThrowingVerbs.

Dentro del *framework*, las pantallas para cada uno de los mini-juegos definen cuatro estados (expresados con un valor numérico) de la aplicación, a saber: Listo (valor 101), EnEjecucion (valor 102), Pausado (valor 103), Terminado (valor 104).

Listo: Indica que la aplicación está en espera de un toque por parte del usuario para comenzar la ejecución.

EnEjecucion: Denota que el mini-juego se encuentra en ejecución y se deben realizar los cálculos y operaciones necesarias para el funcionamiento del juego.

Pausado: Nos indica que ningún cálculo ni operación debe realizarse y que la ventana de pausa debe desplegarse.

Terminado: Es un indicador de que todas las partidas del mini-juego se han presentado al usuario, con lo cual debe presentarse al usuario la puntuación obtenida y decidir si (de acuerdo a ésta) se accede a un nuevo nivel.

Estas variables permiten pasar entre diferentes estado del juego, de modo que si se encuentra en estado pausado, se muestren únicamente los botones para reanudar el juego, salir o la ventana principal. Y si el mini-juego se encuentra en ejecución, la lógica del juego toma el control de forma normal.

Como se mencionó anteriormente, nuestra clase `PantallaThrowingVerbs` hace uso de otros estados para su propia ejecución, de forma que dentro del estado `EnEjecucion` de las clases que heredan de `Pantalla` se tienen modos (representados con valores numéricos), a saber: Inicializando (valor 1001), Jugando (valor 1002), Moviendo (valor 1003).

Inicializando: Indica que todos los elementos de una partida deben ser inicializados, de forma normal se debe hacer un llamado al método `siguienteJugada(…)` del objeto `juegopv`, el cual contiene una referencia al objeto con la lógica del mini-juego, que se explicó al mostrar el constructor de la clase `PantallaThrowingVerbs`, de modo que se prepare la siguiente partida o jugada.

Jugando: Denota que el juego se encuentra en ejecución normal.

Moviendo: Sirve para indicar que el usuario ha elegido una imagen de las que se encuentran disponibles durante una partida, de modo que no se reconocerá ninguna otra entrada por parte del usuario, tal como un toque ó un arrastre. Ya que se está animando el movimiento de la imagen a través de la pantalla del dispositivo.

Para la presentación y manipulación de objetos en pantalla; así como el manejo de los eventos de toque del usuario, la clase `ElementoVisual`, que guarda o envuelve un elemento de tipo `Object`, además de un objeto del SDK de Android de tipo `Rect`, que almacena las coordenadas y dimensiones de un rectángulo, también permite calcular si dos rectángulos se intersecan o si un punto se encuentra dentro del rectángulo. De esta forma podemos manipular la posición de las imágenes y de los verbos sobre la pantalla en todo momento. También contiene el método `getDimensiones()` que devuelve el rectángulo de tipo `Rect` con las dimensiones del elemento que está siendo encapsulado. En la Tabla 14 se muestra un ejemplo de creación del objeto para control del botón de pausa, al cual se le envían, la imagen a la cual encapsula y la posición de las esquinas de éste:

```
//Parámetros que se envían: imagen de pausa,
//coordenada izquierda del rectángulo,
//coordenada superior del rectángulo,
//coordenada derecha del rectángulo.
//y la coordenada inferior del rectángulo
pausa = new ElementoVisual(Assets.pausa, 5, 5, 60, 60);
```

Tabla 14 Ejemplo de creación de un objeto de tipo ElementoVisual.

En la clase `ElementoVisual` se incluye también el método `getPulsado(MotionEvent event)`, de tal forma que con solo enviarle un objeto de tipo `MotionEvent`, que es generado automáticamente al haber un toque sobre la pantalla por parte del usuario, se calculará si el toque sucedió sobre el rectángulo del ele-

mento envuelto por la clase `ElementoVisual`, con lo cual sabemos que ha sido pulsado y se debe actuar de alguna manera.

Ya que durante la ejecución del mini-juego, el *framework* llama a los métodos `update()` y `present()` de las clases que heredan de `Pantalla`, siendo el primero para realizar los cálculos y modificaciones necesarias sobre los objetos que se presentarán en pantalla, y el segundo método para llamar a las clases que dibujarán en pantalla la parte visual del juego. A continuación se explicará cómo están organizados los métodos `present()` y `update()`.

El método `present()` lee el estado, para luego llamar a los métodos que se encargan de dibujar en pantalla los elementos que correspondan con el estado de la aplicación; el fondo de la pantalla no cambia; por lo tanto, se define desde el método `present()`. La Tabla 15 muestra los llamados a cada uno de los métodos para dibujar.

```
public void present() {
           AndroidGraphics g = juego.getGraficos();
           g.dibujaBitmap(Assets.fondo, 0, 0, false);
          switch (estado) {
           case Listo:
                dibujaInterfazListo();
                break;
           case EnEjecucion:
                dibujaInterfazEnEjecucion();
                break;
           case Pausado:
                dibujaInterfazPausado();
                break;
           case Terminado:
                dibujaInterfazTerminado();
                break;
}
```

Tabla 15 Ejemplo de organización del método present() en sub métodos.

El más sencillo es el método `dibujaInterfazListo()`, el cual dibuja un mensaje en pantalla con la leyenda *Touch to start* en espera de que el usuario toque la pantalla. La Tabla 16 muestra el código de este método.

```
private void dibujaInterfazListo() {
           AndroidGraphics g = juego.getGraficos();
           g.dibujaBitmap(Assets.touch, 150, 80, false);
     }
```

Tabla 16 Implementación de uno de los sub métodos del método present().

El método `dibujaInterfazEnEjecucion()` permite ver la representación gráfica del juego en ejecución. Para presentar en pantalla cada una de las 5 imágenes de cada partida durante el mini-juego, se emplea un arreglo sobre el que se itera para extraer los objetos de tipo `Imagen` que se imprimen en pantalla; la Tabla 17 muestra el código de este método.

```
AndroidGraphics g = juego.getGraficos();
//Se dibujan las imágenes en su posición sobre la pantalla
for(int i=0; i<ordenImagenes.length; i++){
     Imagen img = (Imagen) Imagenes[i].getElemento();
     g.dibujaBitmap(img.getImagen(),
Imagenes[i].getDimensiones());
}
```

Tabla 17 Fragmento de código para el dibujado de las imágenes en pantalla.

Posteriormente dentro del mismo método `present()` se dibuja un cuadro de texto que incluye el verbo compuesto, y finalmente, se dibuja el tablero con el número de jugadas correctas y el tiempo restante. Esto se logra con el método de la clase `Graphics`, `dibujaTexto(…)` y el método `dibujaRectanguloTexto(…)`. A continuación se muestra un ejemplo del uso de la clase `dibujaTexto(…)`, haciendo notar que se emplea el vector `colorLetraTablero[]` para almacenar el color ARGB (alfa, rojo, verde y azul) para el texto. Para mayor detalle referirse al Manual Técnico del presente proyecto.

```
g.dibujaTexto(juegopv.getPuntaje() + " of " + juegopv.getImagenes-
ParaMostrarEnNivel() , 380, 60, 20, colorLetraTablero[0], colorLe-
traTablero[1], colorLetraTablero[2], colorLetraTablero[3]);
```

Tabla 18 Uso de la clase Graphics del framework para dibujar texto en pantalla.

El método `update()` de la clase `Pantalla` comprueba primero si el juego se ha terminado; en afirmativo, se modifica el estado de la aplicación, y se guarda en las preferencias: i) el puntaje obtenido, y ii) el nivel de juego al que se avanza, si de acuerdo el puntaje obtenido, se determina que el usuario logró pasar al siguiente nivel. La Tabla 19 muestra parte del código de este método.

```
if(juegopv.gameOver()){
       if(juegopv.nivelCompletado() && estado != EstadoJuego.Termi-
nado)
       {
           int puntaje = Preferencias.cargaPreferenciasInt (juego,
"puntaje");
           puntaje = puntaje + juegopv.getPuntaje();
           Preferencias.guardaPreferenciasInt(juego, "puntaje",
```

```
puntaje);
        }
        estado = EstadoJuego.Terminado;
}
```

Tabla 19 Comprobación del término del mini-juego y guardado de preferencias.

Si el modo de juego es `Inicializando`, simplemente se cargan todos los datos o se restablecen los valores, preparando la aplicación para la siguiente jugada.

Nuestro método `onTouch(…)` de las clases que heredan de `Pantalla`, es invocado desde el *framework*; este método llama, a su vez, a otros métodos dependiendo del estado de la aplicación, como se muestra en el siguiente fragmento de código de la Tabla 20.

```
public void onTouch(View v, MotionEvent event) {
           switch (estado) {
            case Listo:
                 touchListo(event);
                 break;
            case EnEjecucion:
                 touchEnEjecucion(event);
                 break;
            case Pausado:
                  touchPausado(event);
                 break;
            case Terminado:
                  touch Terminado(event);
                  break;
}
```

Tabla 20 Fragmento de código con la organización del método onTouch() en sub métodos.

El método más simple es `touchTerminado(event)` que sólo espera a que haya sucedido un evento de toque por parte del usuario para salir del mini-juego hacia el menú principal. La Tabla 21 muestra el código de este método.

```
private void touchTerminado(MotionEvent event) {
           if(event.getAction() == MotionEvent.ACTION_UP){
                juego.setPantalla(juego.getPantallaInicio());
           }
}
```

Tabla 21 Implementación del método touchTerminado(), sub método de onTouch().

Para la detección de clics o toques por parte del usuario, se optó por la técnica de poleo; esto es, al momento de recibir un evento de toque por parte del usuario, se utiliza al objeto de tipo `MotionEvent` que contiene las coordenadas `x` y `y` del toque, el cual determina la acción correspondiente; por ejemplo, mover una imagen, pausar la aplicación, mover un verbo, entre otras. La elección de la técnica de poleo se debe a que son pocos los elementos sobre los que se hará la comprobación de donde ocurrió el toque, y a que este tratamiento facilita el manejo de toques.

En el método `touchEnEjecucion(MotionEvent event)` se revisa que el modo de juego sea `Jugando` y se revisan todas las imágenes que se muestran en la jugada actual, recorriendo el arreglo que las contiene. Recordar que en este caso son de tipo `ElementoVisual`, y se emplea el método `getPulsado(MotionEvent event)`, para comprobar si el toque se realizó dentro del área ocupada por cada imagen. Si se ha realizado un toque sobre alguna imagen, empleamos el método `reproduceSonido()` de la clase `Assets`, al cual se le envía el identificador del sonido que se requiere reproducir para indicar que se ha elegido una imagen y se está desplazando a través de la pantalla.

La imagen que fue pulsada se asigna a la variable `imagenSeleccionada` para que se pueda mover a través de la pantalla, se cambia el modo de juego para indicar que se moverá la imagen y que los eventos de toque del usuario posteriores serán descartados; finalmente, se emplea la instrucción `break` para salir del ciclo. La Tabla 22 muestra esta parte del código.

```
private void touchEnEjecucion(MotionEvent event) {
       if (event.getAction() == MotionEvent.ACTION_DOWN){
           if (modo == ModoJuego.Jugando){
                 for (int i=0; i<Imagenes.length;i++){
                       if (Imagenes[i].getPulsado(event)){
                       Assets.reproduceSonido( Assets.sonidoThrow,
juego);
                       imagenSeleccionada = Imagenes[i];
                       modo = ModoJuego.Moviendo;
                       break;
                       }
                 }
           }
       }
}
```

Tabla 22 Detección de pulsación de las imágenes en pantalla por medio de poleo.

6.5. Demostración de la aplicación

Para las pruebas de la aplicación comenzamos desde la pantalla principal, la cual únicamente lee las preferencias de la aplicación, a saber: i) la que determina el avance a partir de la evaluación de nivel previo, y de acuerdo a esa evaluación, desbloquear los botones de otros mini-juegos; y 2) la opción de sonido, si ésta está activa o inactiva. Por supuesto revisa el botón pulsado para abrir el mini-juego adecuado.

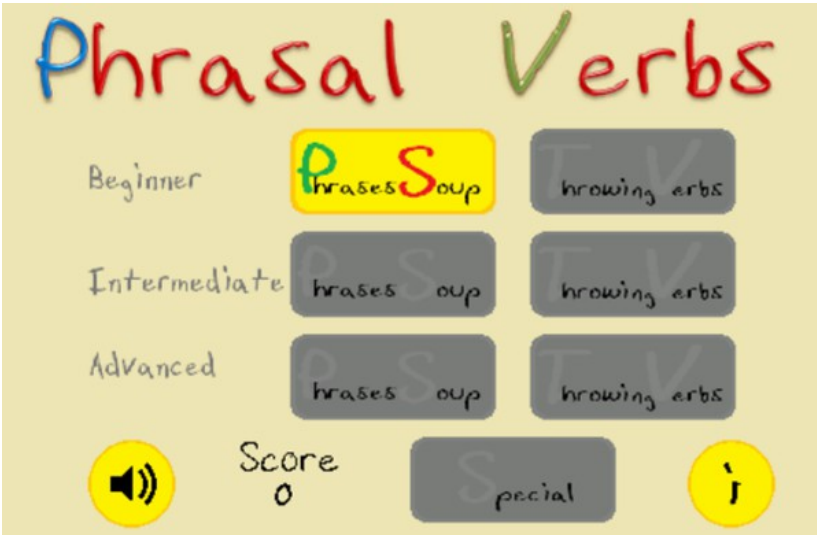

Figura 7 Pantalla principal de la aplicación.

Después de jugar el primer mini-juego y completar el nivel, se obtiene una pantalla como la mostrada en la Figura 8:

Figura 8 Nivel completado con 80% o más de jugadas correctas.

Al término de cada mini-juego se actualiza el puntaje y si se alcanzó el siguiente nivel. La pantalla principal lee también la preferencia de sonido (activado/desacti-

vado); por ejemplo, el nivel 2 desbloquea los botones de mini-juegos Phrases Soup y Throwing Verbs con la dificultad PRINCIPIANTE, ver Figura 9:

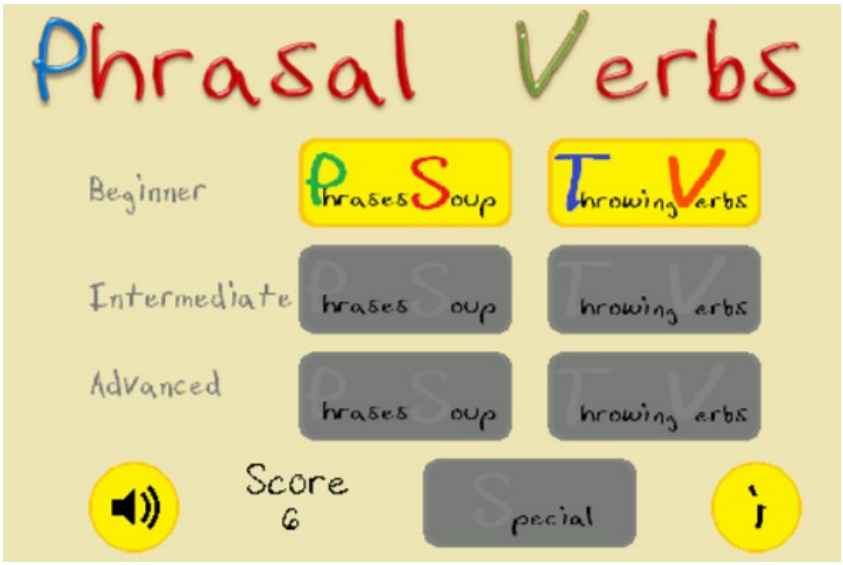

Figura 9 Muestra en pantalla principal del desbloqueo de un mini-juego.

Ahora, la Figura 10 muestra una prueba de ejecución en dificultad principiante del mini-juego Throwing Verbs, en la pantalla principal se muestra el contador de tiempo y el número de jugadas con respuestas correctas.

Figura 10 Pantalla inicial de Throwing Verbs.

Basta con pulsar el botón de pausa para detener el cronómetro y la ejecución del mini-juego. Se muestra la pantalla de pausa con los botones que permiten continuar con la ejecución del mini-juego, volver a la pantalla principal o salir de la apli-

cación. También es posible manipular la reproducción de sonidos desde la pantalla de pausa, como se observa en la Figura 11.

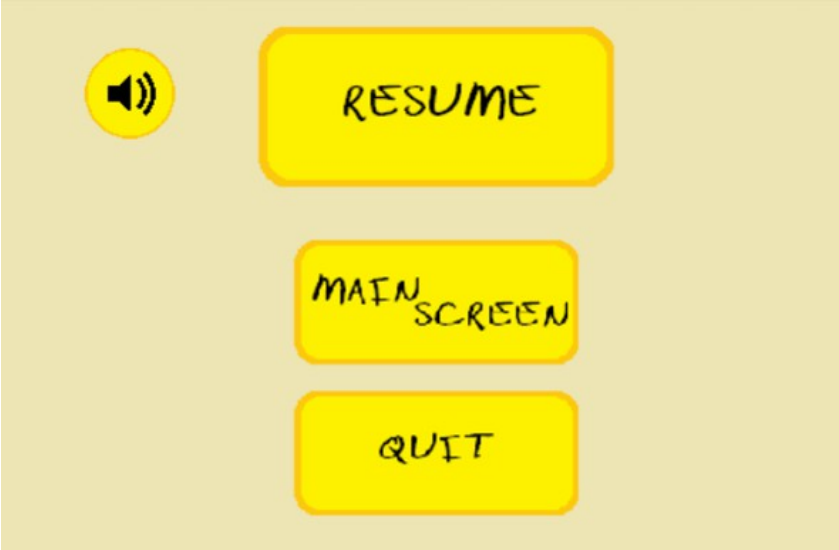

Figura 11 Pantalla de pausa durante un mini-juego.

Después de pulsar el botón *Resume* se continúa con el mini-juego y se pulsa sobre la imagen, ésta es lanzada hacia el verbo compuesto que se muestra en pantalla. Si el verbo compuesto corresponde con la imagen seleccionada, el cuadro que contiene el verbo compuesto cambia a color verde, y si el sonido está activado se reproduce un tono indicando que el movimiento es correcto y se incrementa el número de jugadas correctas, de acuerdo a la Figura 12.

Figura 12 Imagen después de efectuar una jugada correcta.

Al iniciar cada jugada, y seleccionar la imagen que corresponda con el verbo compuesto, el cuadro que contiene el verbo compuesto cambia de color hasta que la imagen ha pasado por debajo del verbo compuesto, ver Figura 13.

Figura 13 Imagen en movimiento antes de tocar pasar por el verbo compuesto.

Una vez que el cuadro que contiene el verbo compuesto ha cambiado de color, la jugada siguiente se muestra hasta que la imagen ha salido de la pantalla, ver Figura 14.

Figura 14 Imagen seleccionada saliendo de pantalla, después de un movimiento correcto.

Si por el contrario, durante una jugada se elige una imagen que no corresponde con el verbo compuesto; el cuadro que contiene el verbo compuesto cambia a color rojo y se reproduce un tono de movimiento incorrecto, ver Figura 15.

Figura 15 Throwing Verbs después de un movimiento incorrecto.

Si después de jugar todas las jugadas disponibles en el nivel no se ha obtenido al menos un 80% de jugadas correctas, se presenta una pantalla de nivel no completado, la puntuación obtenida no se acumula y no se desbloquea un nivel superior (un mini-juego distinto o con una dificultad superior), ver Figura 16.

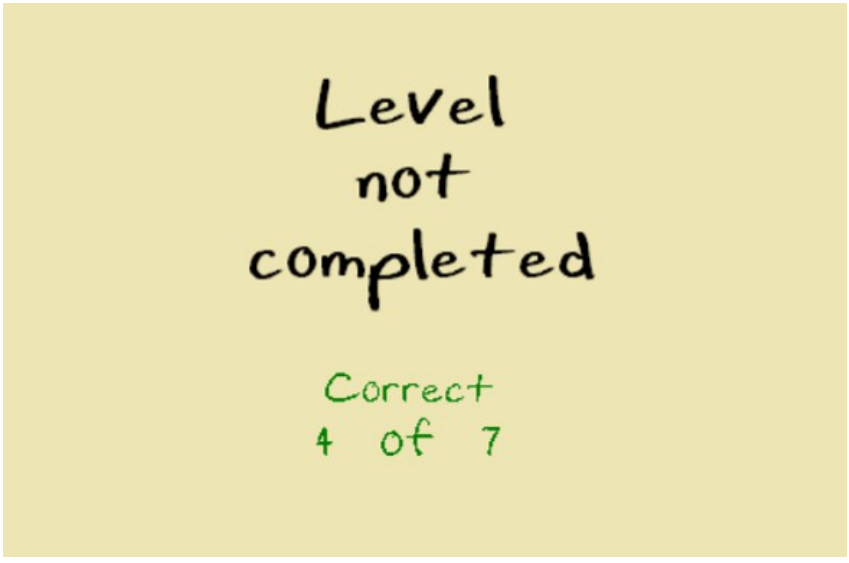

Figura 16 Nivel no completado al término de un mini-juego.

6.6. *Aplicación de Escritorio*

La aplicación de escritorio *PhrasalVerbs Utility*, agrega mayor flexibilidad a la aplicación móvil, ya que por medio de una interfaz gráfica que consiste de adición y eliminación de registros, permite agregar verbos compuestos, que aún no se encuentren en la aplicación *Phrasal Verbs*, con sus respectivas imágenes. En la Figura 17 se puede observar la vista principal de la aplicación de escritorio.

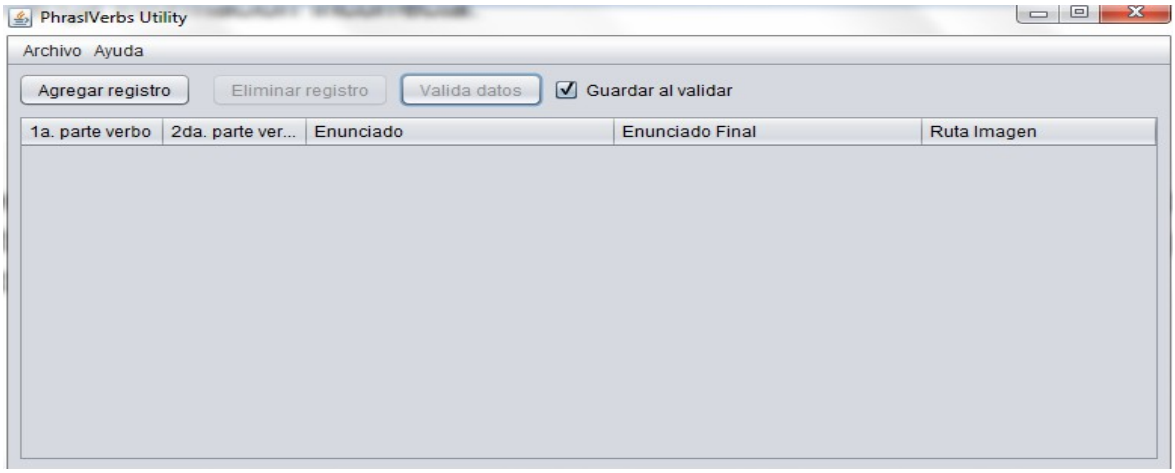

Figura 17 Pantalla principal de la aplicación de escritorio PhrasalVerbs Utility.

En el campo `1a. parte verbo` se debe incluir la primera palabra que forma el verbo compuesto, por ejemplo para el verbo *Look for*, se agregaría únicamente la palabra *Look*. En caso de requerir hacer alguna corrección, basta con dar doble clic sobre el texto ingresado para realizar cambios y editar la palabra ingresada. Si se ingresa más de una palabra en este campo o símbolos distintos de letras del inglés se recibirá un error al realizar la validación.

Para el campo `2a. parte verbo` se debe incluir la segunda palabra que forma el verbo compuesto; siguiendo con el ejemplo anterior para el verbo *Look for*, se agregaría únicamente la palabra *For*. Las validaciones y correcciones se efectúan de la misma forma que en el campo anterior.

En el campo `Enunciado` se debe introducir un enunciado que emplee el verbo compuesto y de preferencia que haga referencia a la imagen que se relacionará con el verbo compuesto. El único detalle es que en la posición donde serían colocadas las partes del verbo compuesto se debe agregar un símbolo *, respetando los espacios entre palabras. Nuestro ejemplo quedaría de la siguiente manera: *I'm * * my watch. Have you seen it?*

En el caso del campo `Enunciado Final` se incluirá el enunciado con el verbo compuesto conjugado correctamente para que el enunciado tenga coherencia y sea correcto gramaticalmente. Para el ejemplo tendremos: *I'm looking for my watch. Have you seen it?*

Finalmente, en el campo `Ruta imagen`, es necesario dar clic en el botón `Seleccione imagen` para abrir el cuadro de diálogo donde se buscará la ubicación de la imagen que se asociará con el verbo compuesto, los formatos admitidos son: png (recomendado), jpg y bmp. Una vez elegido el archivo, el botón cambia e incluye la ruta del archivo de la imagen que se cargará.

El botón `Valida datos` comprueba que los datos ingresados en la ventana principal respeten el formato indicado previamente en cada campo. Al pasar la validación se puede guardar la carpeta correspondiente con archivos. Esta carpeta debe llamarse phrasalverbsutility, y debe copiarse en la tarjeta de memoria del teléfono. La aplicación con un ejemplo de datos correctos se puede apreciar en la Figura 18.

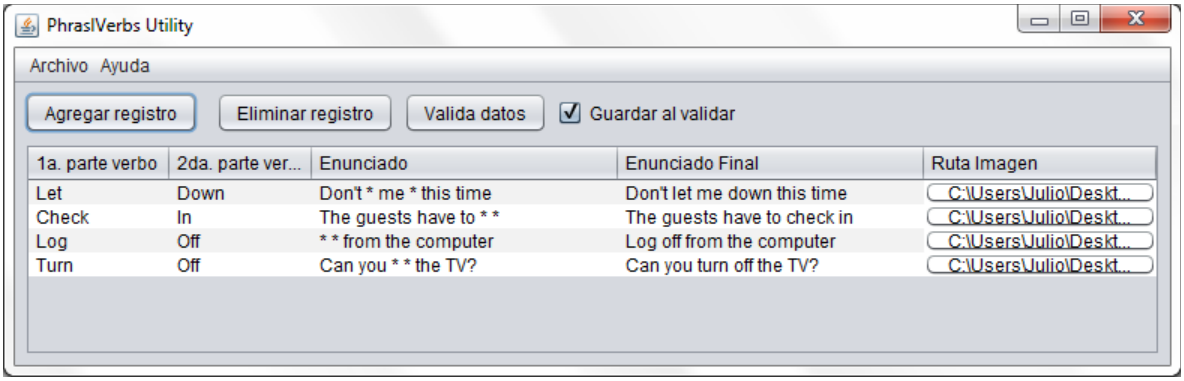

Figura 18 Datos ingresados en la ventana principal.

Al ejecutar la aplicación móvil *Phrasal Verbs*, ésta detecta si la carpeta está disponible desde la tarjeta de memoria del teléfono, si contiene archivos generados con la aplicación de escritorio y en el formato correcto para incluirlos dentro de los mini-juegos.

7. Conclusiones y Perspectivas del Proyecto

La finalidad de este proyecto fue implementar un juego para dispositivo móvil, con sistema operativo Android, para la práctica de verbos compuestos del idioma inglés *phrasal verbs*. Además se implementó una aplicación de escritorio para agregar más verbos con sus respectivas imágenes a la aplicación. A continuación se muestra la Tabla 23 con el detalle de los objetivos alcanzados en el presente proyecto.

Objetivos Particulares	Alcance
Diseñar e implementar el mini juego Throwing verbs.	Se diseñó e implementó el mini-juego, inicialmente se planteó que la imagen variara su velocidad de desplazamiento de acuerdo a la velocidad con que se pasara el dedo, por cuestiones de tiempo se configuró un desplazamiento no dependiente de la velocidad de movimiento del dedo sobre la pantalla.
Diseñar e implementar el mini juego Phrases Soup.	Se diseñó e implementó el mini-juego.
Diseñar e implementar el mini juego Special.	Se diseñó e implementó el mini-juego Special.
Integrar los mini juegos dentro de la aplicación móvil para establecer la secuencia del juego.	Se implementó la pantalla principal que de acuerdo al nivel alcanzado previamente por el usuario, desbloquea niveles de juego superiores.
Implementar un módulo de escritorio que permita agregar más verbos e imágenes a la aplicación móvil.	Se implementó el módulo de escritorio, que permite enriquecer la aplicación móvil con más verbos compuestos y sus imágenes. Este módulo genera una carpeta con los archivos: listaenunciados.dat, listaverbos.dat y archivos png enumerados del 1 hasta el número de verbos agregados, que con solo copiarse a la tarjeta de memoria externa del dispositivo móvil es leído por la aplicación *Phrasal Verbs.*

Realizar pruebas para verificar que los archivos generados por la aplicación de escritorio sean reconocidos automáticamente por la aplicación móvil al abrirla.	La aplicación *Phrasal Verbs* carga: los archivos ejecutables de la aplicación móvil y los archivos generados por la aplicación de escritorio, los cuales se encuentran almacenados en la tarjeta de memoria externa.

Tabla 23 Comparativo de objetivo cumplidos.

El haber empleado una tecnología en pleno desarrollo como Android y el lenguaje de programación java, con su enfoque hacia el paradigma orientada a objetos, facilitó el manejo de eventos, y la representación de imágenes, texto y sonido en la pantalla.

Agradecemos la participación del ilustrador Rubén de la Torre Hernández, quien se encargó de realizar las imágenes que emplea la aplicación Phrasal Verbs durante su ejecución. Es importante mencionar Rubén de la Torre Hernández trabajó durante 2 semanas, porque tuvimos un cambio de ilustrador; como consecuencia el número de imágenes con que cuenta la aplicación es de treinta y no de cien como se planteó inicialmente. Sin embargo, gracias a la flexibilidad de la aplicación, es posible agregar nuevos verbos, oraciones e imágenes a través de la aplicación de escritorio.

Dado que la aplicación busca una carpeta específica en la tarjeta externa de memoria, durante la carga, se podría en un trabajo futuro, diseñar un módulo dentro de la aplicación móvil para, en tiempo de ejecución, elegir la ruta para cargar las imágenes, así como seleccionar los verbos cargados en tiempo de ejecución que se quieren utilizar en los mini-juegos.

Las características de la aplicación *Phrasal Verbs*, permiten que en trabajos futuros se puedan agregar nuevos mini-juegos, empleando los elementos propuestos desde el diseño: `ParteVerbo`, `VerboCompuesto`, `EnunciadoVerboCompuesto` e `Imagen`.

Además, existe la posibilidad de que a través de la aplicación de escritorio, se puedan generar nuevos mini-juegos y que las reglas de juego se guarden en archivos que puedan ser leídos por la aplicación móvil *Phrasal Verbs* sin la necesidad de reprogramar la aplicación y generar un nuevo archivo ejecutable.

Incluso, podría agregarse un modo de juego contra otros usuarios de teléfonos móviles con conexión vía *bluetooth*, de modo que en cada jugada, el primero en responder correctamente gana la jugada, lo cual haría aún más interesante el juego por este grado de competitividad.

8. Bibliografía y Referencias

[1] E. Gamma, R. Helm, R. Johnson y J. Vlissides, "*Design Patterns: Elements of Reusable Object-Oriented Software*", Addison-Wesley Professional, 1994.

[2] M. Zechner, "*Beginning Android Games*", Apress, 2011.

[3] A. Weitzenfeld , "Ingeniería de Software Orientada a Objetos con UML, Java e Internet", Thomson International, 2004.

[4] Android de Google, "*Installing the SDK*", http://developer.android.com/sdk/installing/index.html, 2012.

[5] Oxford University, Really Learn Phrasal Verbs, Oxford University Press, 2003.

[6] T.Fullerton, C. Swain y S. S. Hoffman, "*Game Design Workshop: A Playcentric Approach to Creating Innovative Games*", Elsevier, 2008.

[7] Creative Commons Organization, "*Creative Commons*", http://creativecommons.org/, 2013.

[8] Licencias Creative Commons, "*Attribution 3.0 Unported (CC BY 3.0)*", http://creativecommons.org/licenses/by/3.0/, 2013.

[9] FreeSound.org, "*Freesound.org - Game Sound Correct.wav by Bertrof*", http://www.freesound.org/people/Bertrof/sounds/131660/, 2013.

[10] FreeSound.org, "*Freesound.org - Bertrof*", http://www.freesound.org/people/Bertrof/, 2013.

[11] FreeSound.org, "*Freesound.org - Error.wav by Autistic Lucario*", http://www.freesound.org/people/Autistic%20Lucario/sounds/142608/, 2013.

[12] FreeSound.org, "*Freesound.org - Autistic Lucario*", http://www.freesound.org/people/Autistic%20Lucario/, 2013.

[13] FreeSound.org, "*Freesound.org - cartoon_siren_whistle_001.wav by soundscalpel.com*", http://www.freesound.org/people/soundscalpel.com/sounds/110390/, 2013.

[14] FreeSound.org, "*Freesound.org - soundscalpel.com*", http://www.freesound.org/people/soundscalpel.com/, 2013.

[15] NRSI: Computers & Writing Systems, "*SIL Open Font License (OFL)*", http://scripts.sil.org/OFL, 2013.

[16] Openfontlibrary.org, "*Openfontlibrary.org - vonStrong*", http://openfontlibrary.org/en/member/vonStrong, 2013.

[17] Documentación Android de Google, "Activity - *Android Developers*", http://developer.android.com/reference/android/app/Activity.html, 2013.

Anexo

1. Manual del Usuario

1.1. Pantalla Principal

La primera vez que se ejecuta la aplicación desde el dispositivo móvil se aprecia la pantalla principal en la Figura 19, la cual contiene los botones para iniciar los mini-juegos, el de activación de sonido, la puntuación y la ventana de créditos.

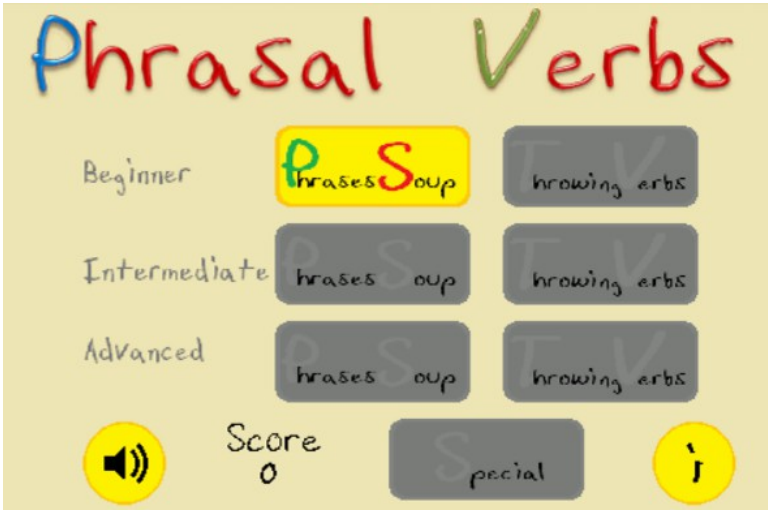

Figura 19 Pantalla principal.

Inicialmente los botones de acceso a los juegos están desactivados excepto el primero, *Phrases Soup* en versión principiante, ya que al obtener un 80% de respuestas correctas se desbloqueará el mini-juego *Throwing Verbs* en versión también principiante. El desbloqueo de los mini-juegos posteriores funcionará de la misma forma.

El botón para activar el sonido que se aprecia en la Figura 20, habilita o deshabilita la reproducción de sonidos (por ejemplo al obtener respuestas correctas, incorrectas, o al efectuar ciertos movimientos). El ícono indica si el sonido está activo o inactivo, desde la ventana de pausa durante la ejecución de un mini-juego se puede acceder también al botón.

Figura 20 Botón sonido.

La sección *Score* presenta al usuario la puntuación obtenida, ya que al completar un mini-juego se acumulará la cantidad de respuestas correctas obtenidas durante una mini-juego. Se muestra un ejemplo de la puntuación inicial en la Figura 21.

Figura 21 Sección Score.

El botón *Info* proporciona información acerca de los créditos y datos adicionales de la aplicación. Ver Figura 22.

Figura 22 Botón informativo.

1.2. Mini-Juegos

Para iniciar cualquier mini-juego se pedirá al usuario tocar la pantalla para indicar que se encuentra listo para iniciar, tal como se muestra en la Figura 23.

Figura 23 Inicio mini-juegos.

En los mini-juegos se presenta al usuario un tablero con el número de respuestas correctas del total de partidas disponibles, así como el tiempo restante para responder o efectuar una jugada. Ver Figura 24.

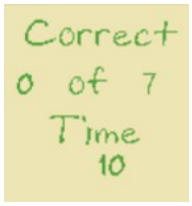

Figura 24 Tablero de información.

Todos los mini-juegos incluyen un botón de pausa para suspender momentáneamente la ejecución del mini-juego, como se observa en la Figura 25, la cual llevará a un nuevo menú de opciones.

Figura 25 Botón de pausa.

El menú de opciones, durante una pausa, muestra nuevamente un botón para activar/desactivar el volumen, así como tres opciones para control de la ejecución del mini-juego, ver Figura 26. El botón *Resume* continúa con la ejecución del mini-juego, el botón *Main Screen* termina la ejecución del mini-juego actual y vuelve a la pantalla principal para iniciar un nuevo mini-juego. Finalmente el botón *Quit* termina la ejecución de la aplicación.

Figura 26 Pantalla de pausa.

1.3. Mini-juego Phrases Soup

La finalidad del mini-juego *Phrases Soup* es formar un verbo compuesto con 2 de las 6 palabras que aparecen en pantalla y que sean acordes con la imagen que se presenta. La pantalla principal se presenta en la Figura 27.

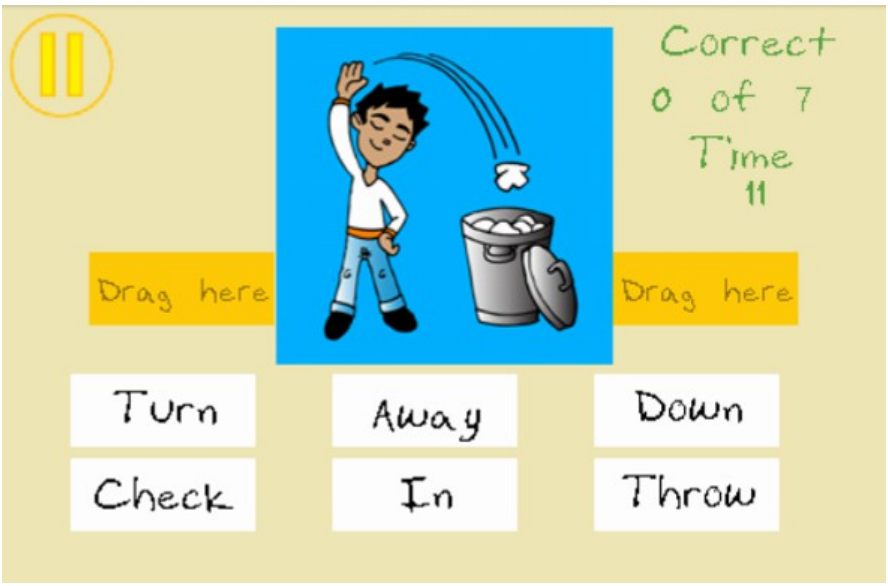

Figura 27 Inicio de ejecución Phrases Soup.

Esto se logra arrastrando una de las palabras hasta el cuadro que derecho o izquierdo (con la leyenda *Drag here*) según se requiera formar el verbo compuesto. Una vez que se haya soltado la palabra, ésta puede volver a colocarse en su lugar original si quiere hacerse alguna corrección con solo sacarla del cuadro naranja y dejar de tocar la pantalla.

Figura 28 Elección de la primera palabra.

Si se forma el verbo correctamente, corresponde con la imagen y aún no se ha terminado el tiempo, los cuadros laterales a la imagen cambiarán a color verde y después de algunos segundos se mostrará la partida siguiente como se muestra en la Figura 29. En caso de agotarse el tiempo se mostrará la siguiente partida y en caso de una respuesta incorrecta los cuadros laterales se tornarán color rojo y después de algunos segundos se mostrará la partida siguiente si la hay.

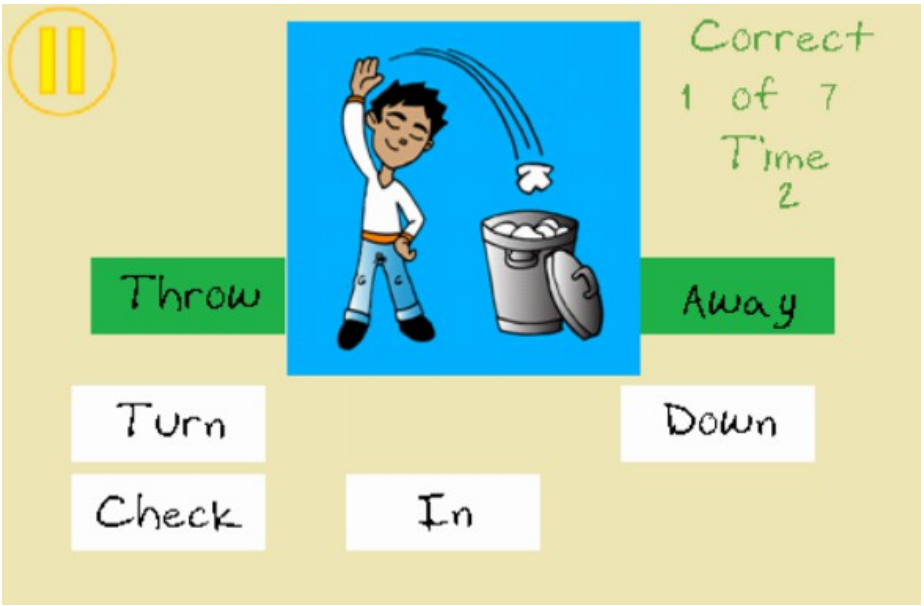

Figura 29 Movimiento correcto al formar el verbo compuesto adecuado.

1.4. Mini-juego Throwing Verbs

Durante el mini-juego *Throwing Verbs* se muestran 5 imágenes y un verbo compuesto. Se requiere asociar una de las imágenes con el verbo compuesto que mejor describa la acción que se realiza. En la Figura 30 se presenta la pantalla de inicio del mini-juego.

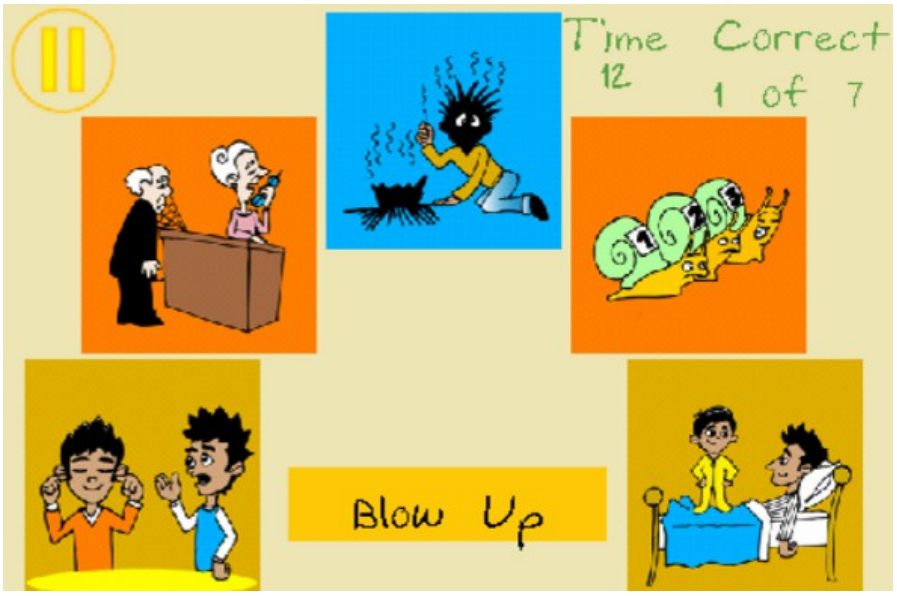

Figura 30 Inicio de ejecución Throwing Verbs.

Con sólo tocar una de las imágenes, ésta será lanzada hacia el verbo compuesto y al pasar por debajo del verbo compuesto, éste cambiará el color del cuadro que lo contiene: color verde si la asociación es correcta ó color rojo en caso contrario, tal como se muestra en la Figura 31. Después de que la imagen salga de la pantalla se mostrará la jugada siguiente. En caso de agotarse el tiempo se muestra la siguiente partida si aún hay partidas por mostrar al usuario.

Figura 31 Movimiento incorrecto al elegir la imagen.

1.5. Mini-Juego Special

Para el mini-juego *Special* se presentan al usuario 3 imágenes con su verbo compuesto y un enunciado en inglés al cual le falta un verbo compuesto para tener coherencia. En la Figura 32 presentamos la pantalla de inicio del mini-juego:

Figura 32 Inicio de ejecución de Special.

El usuario debe arrastrar la imagen por encima del enunciado y soltarla para formar el enunciado adecuado. Si el verbo compuesto corresponde con el enunciado mostrado, el enunciado cambia el color de fondo por verde como se observa en la Figura 33, en caso contrario se cambia por color rojo. Después de algunos segundos se presenta al usuario una nueva partida, si la hay, de forma que se conozca el motivo por el cual la respuesta elegida es correcta o incorrecta.

Figura 33 Movimiento correcto, el verbo se muestra conjugado correctamente.

1.6. Terminación de los mini-juegos

Después de presentar al usuario todas las partidas disponibles, la pantalla de fin de mini-juego, indicará los puntos obtenidos durante éste y en caso de haberlo completado con un 80% de respuestas correctas los puntos se acumulan en la puntuación total y, además se desbloquea el mini-juego siguiente o uno posterior con mayor dificultad.

Siempre se informa al usuario si el nivel no se ha completado satisfactoriamente como se muestra en la Figura 34, o si de acuerdo a la puntuación obtenida se ha completado el nivel, ver Figura 35.

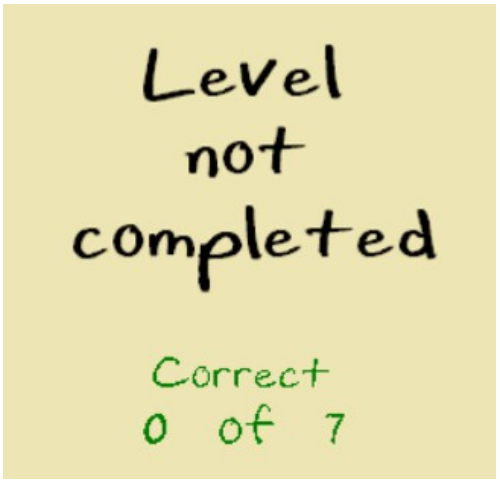

Figura 34 Pantalla de fin de mini-juego con nivel no completado.

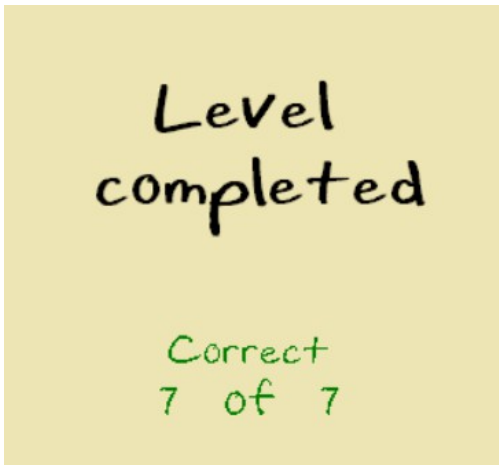

Figura 35 Pantalla de fin de mini-juego con nivel completado.

1.7. Aplicación de Escritorio

La aplicación de escritorio *PhrasalVerbs Utility*, agrega mayor flexibilidad a la aplicación móvil, ya que permite agregar y eliminar nuevos verbos compuestos, que aún no se encuentren en la aplicación *Phrasal Verbs*, con sus respectivas imágenes. La aplicación generara archivos dentro de una carpeta que debe llamarse *phrasalverbsutility*, la cual se debe copiar en la tarjeta de memoria del teléfono (el medio de transferencia se deja a elección del usuario de acuerdo a las características del teléfono, por conexión USB o *bluetooth*).

Al ejecutar la aplicación móvil, ésta detecta si la carpeta está disponible desde la tarjeta de memoria del teléfono, si contiene archivos generados con la aplicación de escritorio y en el formato correcto, para incluirlos dentro de los mini-juegos.

Se hace notar que el uso de la aplicación de escritorio se recomienda solo para profesores del idioma, de forma que el material con el que se enriquezca el juego no contenga información incorrecta. Además, para el caso de verbos compuestos *Phrasal Verbs* y efectos de éste proyecto sólo se utilizan aquellos formados por 2 partículas o palabras.

1.7.1. Creación un nuevo listado

Una vez en la aplicación se pueden agregar nuevos verbos compuestos dando clic en el botón `Agregar registro` o también, desde el menú `Archivo` seleccionando `Nuevo` para empezar todo nuevamente. La pantalla principal se presenta en la Figura 36.

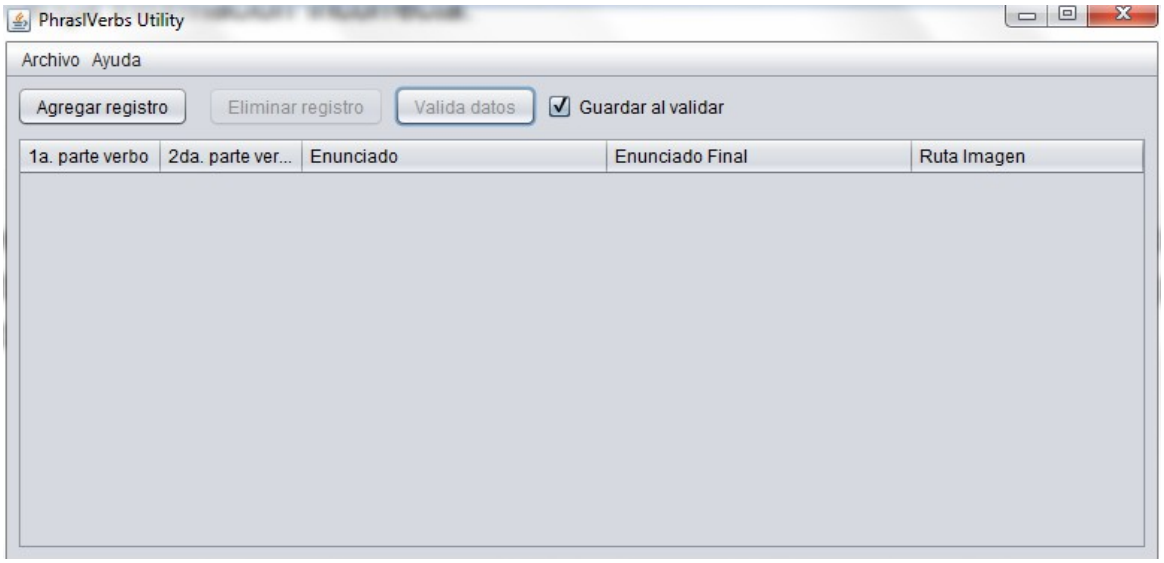

Figura 36 Pantalla principal de la aplicación de escritorio.

En el campo `1a. parte verbo` se debe incluir la primera palabra que forma el verbo compuesto, por ejemplo para el verbo *Look for*, se agregaría únicamente la palabra *Look*. En caso de requerir hacer alguna corrección, basta con dar doble clic sobre el texto ingresado para realizar cambios y editar la palabra ingresada. Si se ingresa más de una palabra en este campo o símbolos distintos de letras del inglés se recibirá un error al realizar la validación.

Para el campo `2a. parte verbo` se debe incluir la segunda palabra que forma el verbo compuesto; siguiendo con el ejemplo anterior para el verbo *Look for*, se agregaría únicamente la palabra *For*. Las validaciones y correcciones se efectúan de la misma forma que en el campo anterior.

En el campo `Enunciado` se debe introducir un enunciado que emplee el verbo compuesto y de preferencia que haga referencia a la imagen que se relacionará con el verbo compuesto. El único detalle es que en la posición donde serían colocadas las partes del verbo compuesto se debe agregar un símbolo *, respetando los espacios entre palabras. Nuestro ejemplo quedaría de la siguiente manera: *I'm * * my watch. Have you seen it?*

En el caso del campo `Enunciado Final` se incluirá el enunciado con el verbo compuesto conjugado correctamente para que el enunciado tenga coherencia y sea correcto gramaticalmente. Para el ejemplo tendremos: *I'm looking for my watch. Have you seen it?*

Finalmente, en el campo `Ruta imagen`, es necesario dar clic en el botón `Seleccione imagen` para abrir el cuadro de diálogo donde se buscará la ubicación de la imagen que se asociará con el verbo compuesto, los formatos admitidos son: png (recomendado), jpg y bmp. Una vez elegido el archivo, el botón cambia e incluye la ruta del archivo de la imagen que se cargará. Si se presiona el botón nuevamente, se muestra la imagen seleccionada y el verbo al cual está asociada, como se observa en la Figura 37.

Figura 37 Presentación de la imagen que se ha seleccionado para asociar con un verbo.

En la Figura 38 se muestra la aplicación de escritorio con datos ingresados en un registro.

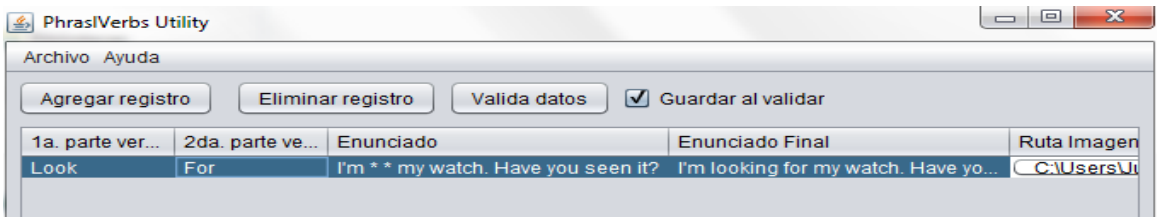

Figura 38 Los datos de un nuevo verbo compuesto en la aplicación de escritorio.

Si por algún motivo se requiere eliminar un registro, basta con seleccionar la fila requerida y pulsar el botón `Eliminar registro` para quitarlo del listado.

1.7.2. Guardar un listado

Una vez agregados los datos necesarios para nuevos verbos compuestos, se activa el botón `Valida datos`, el cual revisa que los campos tengan el formato adecuado y que la imagen seleccionada esté disponible. Si algún campo contiene errores, un aviso de error indica la fila con la celda que contiene errores. Ver Figura 39.

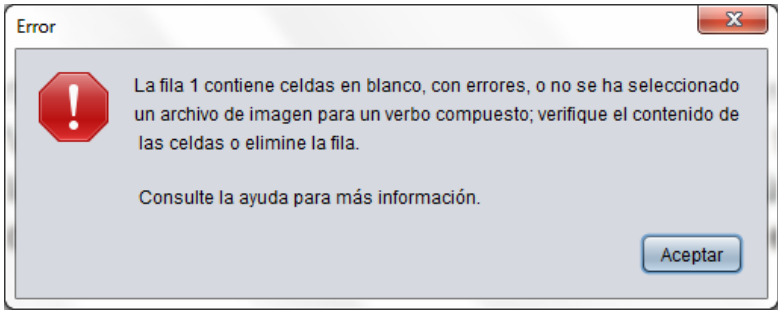

Figura 39 Error en la validación de los datos.

La casilla de verificación `Guardar al validar` permite guardar un nuevo listado de verbos compuestos al terminar la validación, de forma que si todos los datos son correctos, aparece una ventana de guardado para elegir la ubicación de la carpeta que contendrá los datos del listado, como se aprecia en la Figura 40.

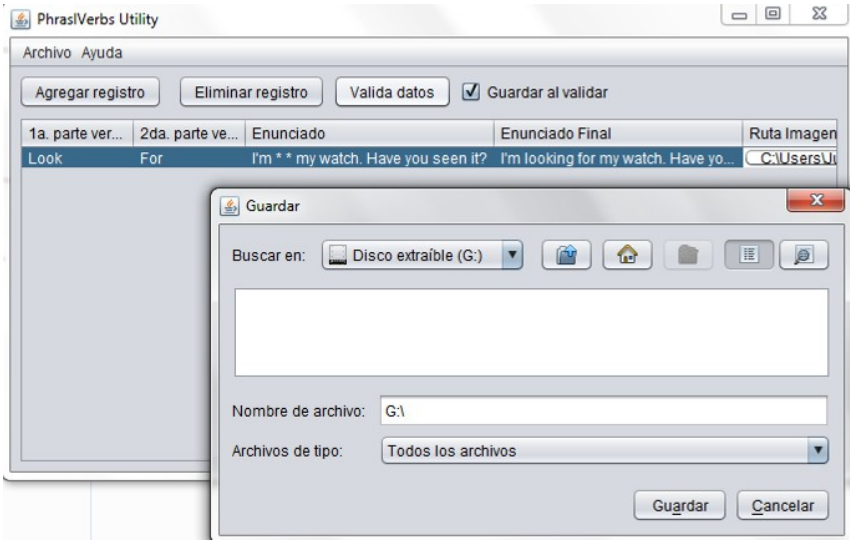

Figura 40 Ventana de guardado una vez pasado el proceso de validación.

Al terminar el proceso de guardado y elegir una carpeta valida, los datos son copiados a esa carpeta; las imágenes se comprimen para reducir sus dimensiones y el tamaño de archivo, y aparece una ventana confirmando el proceso. La Figura 41 muestra dicha ventana de confirmación.

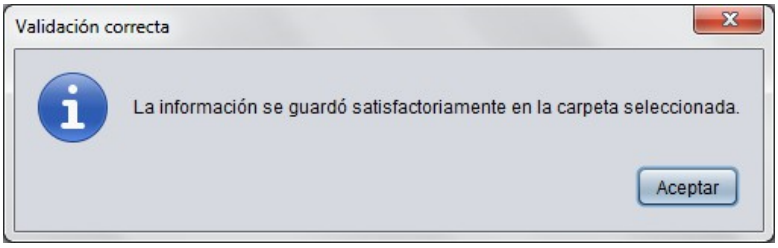

Figura 41 Ventana de confirmación de guardado de listado.

La aplicación genera en la carpeta seleccionada los archivos: `listaenunciados.dat` y `listaverbos.dat`, además dependiendo del número de verbos generados, una secuencia de imágenes con formato png comenzando por `1.png`, `2.png`, `3.png`, etc.

1.7.3. Cargar un listado ya existente

El proceso de cargar un listado existente, desde la aplicación de escritorio, es sencillo; basta con acceder al menú `Archivo` y seleccionar `Cargar`, con lo cual aparece una ventana de diálogo para elegir la carpeta que contiene los archivos a cargar en la aplicación. Una vez elegida la carpeta y si contiene los archivos

requeridos por la aplicación de escritorio y con el formato correcto se muestran en la tabla de registros como se observa en la Figura 42.

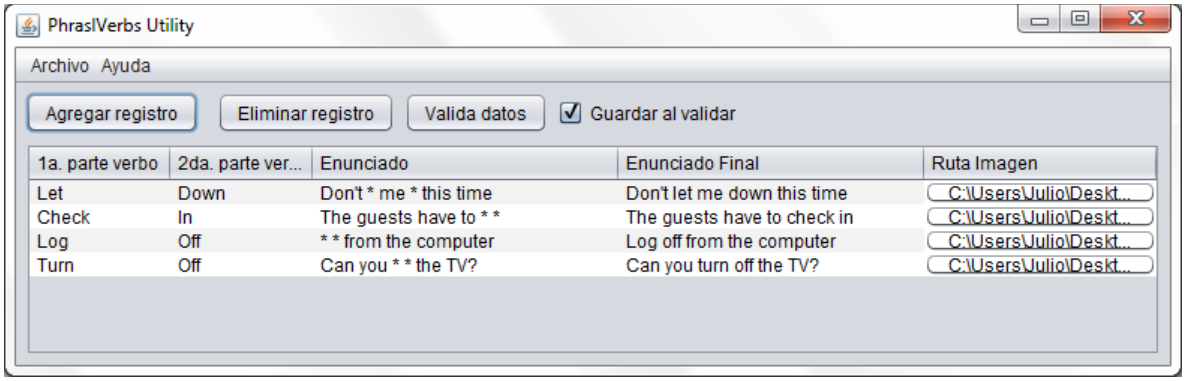

Figura 42 Pantalla de la aplicación con registros creados previamente, cargados desde una carpeta

Ya que se han cargado los registros se pueden realizar las mismas operaciones que para nuevos listados, esto es, editar campos, eliminar filas, etc. Así mismo, es posible volver a guardar el listado, ya sea sobre la carpeta de origen o una nueva, la cual puede seleccionarse al pulsar el botón `Valida datos` cuando la casilla de verificación `Guardar al validar` esté seleccionada.