UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Titulo: Juegos para anotar música usando dispositivos móviles
Autores: Ferraresso, Alejandro - Larghi, José Ignacio
Director: Díaz, Javier
Codirector: Queiruga, Claudia A.
Carrera: Licenciatura en sistemas

Resumen

A pesar del avance de la tecnología, actualmente las computadoras no poseen la capacidad de percepción ni la creatividad que los seres humanos poseen por naturaleza. El presente trabajo tiene como objetivo desarrollar un juego basado en el concepto de GWAP (Game With A Purpose), el cual permite recolectar información sobre archivos de audio de un repositorio determinado, utilizando para esto la capacidad de los jugadores y la información que estos generan en cada partida del juego.
El juego, que denominamos mGWAP (mobile Game With A Purpose) fue desarrollado para ejecutarse sobre dispositivos móviles, más precisamente smartphones, que utilizan el sistema operativo Android. El objetivo de esto fue abarcar un área aún no explotada por este tipo de juegos.
Durante el proceso de desarrollo del proyecto, se evaluaron las aplicaciones existentes en el área de GWAP y se estudiaron las herramientas a utilizar en la implementación del juego. Como resultado se obtuvo una herramienta para la recolección de información de archivos de audio, la cual fue probada y de la que se recolectaron datos estadísticas sobre los resultados obtenidos. Finalmente se implementó un prototipo para ejecutar consultas sobre los datos recolectados.

Palabras Claves

Juegos con un propósito, dispositivos móviles, Android, GWAP, etiquetado de audio, JAVA, Human Computation, CAPTCHA, Input-Agreement, Comet, Bayeux

Trabajos Realizados

Se estudiaron los conceptos de Human Computation y GWAP. También se evaluaron las aplicaciones existentes dentro del área. Una vez adquirido el conocimiento sobre los fundamentos, se comenzó a desarrollar mGWAP, definiendo sus reglas y funcionalidades, y por último se llevó a cabo la implementación del mismo. Finalizado el desarrollo del juego, se realizaron diferentes pruebas de concepto con el objetivo de obtener datos estadísticos. También se implementó un prototipo web con el objetivo de que consuma la información generada por mGWAP.

Conclusiones

La primera conclusión es que, aún queda mucho por seguir investigando, ya que son muchas las ramas que se desprenden del tema en cuestión, tanto en el área de desarrollo, como en el área de implementación de los algoritmos de procesamiento de la información generada. La segunda conclusión, luego de haber estudiado un concepto como GWAP, es que presenta un gran abanico de aplicaciones posibles, lo cual permite sacar provecho de la inteligencia humana para entrenar sistemas computacionales.

Trabajos Futuros

Como líneas de trabajo futuras se plantean:
- Mejora de la interface de usuario.
- Optimización del modo un solo jugador.
- Pruebas con un conjunto mayor de usuarios.
- Implementación del juego en otras plataformas móviles.

Fecha de la presentación: Octubre 2010

Agradecimientos

Queremos agradecer principalmente a Javier Díaz y Claudia Queiruga quienes nos guiaron a lo largo de todo el proceso de desarrollo del presente trabajo, brindándonos su apoyo.

A nuestras familias y amigos ya que sin su incondicional apoyo no hubiese sido posible llegar a esta instancia.

Gracias.

Resumen

A pesar del avance de la tecnología, actualmente las computadoras no poseen la capacidad de percepción ni la creatividad que los seres humanos poseen por naturaleza. El presente trabajo tuvo como objetivo desarrollar un juego basado en el concepto de GWAP (Games with a purpose, Juegos con un propósito), el cual permite recolectar información sobre archivos de audio de un repositorio determinado, utilizando para esto la capacidad de los jugadores y la información que estos generan en cada partida del juego. El juego, al cual se lo denominó mGWAP, fue desarrollado para ejecutarse sobre dispositivos móviles, más precisamente smartphones, que utilizan el sistema operativo Android. El objetivo de esto fue cubrir un sector aun no explotado por este tipo de juegos.

Durante el proceso de desarrollo del proyecto, se analizaron aplicaciones existentes en el área de GWAP y se estudiaron las herramientas a utilizar en la implementación del juego. Como resultado se obtuvo una herramienta para la recolección de información de archivos de audio, la cual fue probada y se recolectaron estadísticas sobre los resultados obtenidos. Finalmente se implementó un prototipo para ejecutar consultas sobre los datos recolectados por mGWAP.

ÍNDICE

CAPÍTULO 1

Introducción

Introducción

Hoy en día millones de personas alrededor del mundo dedican horas de su vida a jugar juegos de computadora on-line, ¿Qué sucedería si se utilizara todo ese tiempo y esfuerzo para resolver problemas de gran escala que hoy en día ninguna computadora puede resolver?

Como por ejemplo la clasificación de archivos de audio, de imágenes, y todos aquellos problemas que requieren de la subjetividad humana. En la actualidad existen gigantescas bases de datos que contienen miles de archivos de audio, el problema es que, realizar búsquedas sobre tales repositorios utilizando criterios subjetivos no es posible, dado que para esto previamente se tendría que haber etiquetado cada archivo de audio con diferentes palabras con un significado subjetivo.

A pesar del avance de la tecnología, actualmente las computadoras no poseen la capacidad de percepción ni la creatividad que los seres humanos poseen por naturaleza, debido a esto las computadoras de hoy no pueden realizar una clasificación subjetiva que sirva como herramienta para poder clasificar un conjunto de elementos, como archivos de audio.

Problemas similares aparecen en la clasificación de imágenes, la digitalización de documentos deteriorados, filtrado de contenidos de Internet, reconocimiento de patrones donde es necesaria la capacidad subjetiva de una persona y muchos otros. Una posible solución a estos problemas es la utilización de la técnica conocida como Human Computation [1].

Human Computation contempla el cerebro de cada individuo como un pequeño procesador dentro de un sistema distribuido, donde cada uno puede procesar un pequeña parte de un cómputo mucho mayor. Human Computation es una técnica en la cual un proceso computacional ejecuta su función delegando ciertos pasos en humanos. En la computación tradicional, los humanos emplean una computadora para resolver un problema, el humano provee una descripción formalizada de un problema a la computadora y recibe una solución para interpretar. Human Computation a

menudo revierte los roles; la computadora solicita a una persona o a un gran grupo de personas resolver un problema, entonces recolecta, interpreta e integra sus soluciones.

Esta técnica de análisis de datos se denomina Human Computation porque combina la inteligencia colectiva de un determinado número de participantes humanos para resolver una tarea que no puede ser fácilmente automatizada.

De esta manera, utilizando un enfoque basado en juegos, una población de usuarios puede resolver un problema de gran escala (Ej.: etiquetar todas las imágenes de Internet) utilizando las contribuciones voluntarias individuales (Ej.: jugando un determinado juego para etiquetar una simple imagen).

Combinando la técnica Human Computation y los millones de personas alrededor del mundo dispuestas a invertir horas jugando en línea surge lo que se conoce como GWAP [2] (Game with a purpose, Juegos con un propósito).

El concepto de GWAP podría definirse como juegos en donde cada participante realiza una porción del procesamiento de un computo mayor, el cual es resuelto combinando los procesamientos realizados por cada participante del juego, cuando decimos procesamiento nos referimos al ejercicio mental que realiza el participante para resolver una porción del computo. Ejemplos de GWAP pueden encontrase en los trabajos realizados por Luis von Ahn [3], los cuales están disponibles en el sitio Web www.gwap.com, también el buscador Google posee en versión experimental de un GWAP para la clasificación de imágenes denominado Google Image Labeler [4].

Sumado a esto tenemos los dispositivos móviles los cuales han avanzado en tecnología y popularidad de manera asombrosa en los últimos años, a partir de de esta evolución podemos pensar en un nuevo campo de desarrollo para los juegos GWAP. Una gran cantidad de los dispositivos móviles actuales permiten a sus usuarios conectarse a Internet, ejecutar aplicaciones y reproducir archivos multimedia, entre otras cosas. En la actualidad es común que un usuario se conecte a Internet, descargue un juego y juegue a través de Internet utilizando su dispositivo móvil.

La combinación de GWAP y dispositivos móviles nos da la posibilidad de realizar procesamientos de gran escala basados en el concepto de Human Computation, utilizando el tiempo y el esfuerzo de cada usuario de un dispositivo móvil alrededor del mundo que se conecte para jugar un GWAP.

Objetivo del desarrollo

Se propone desarrollar un juego de tipo GWAP para dispositivos móviles, el cuál denominaremos mGWAP (mobile Game with a purpose). El mismo permitirá llevar el concepto de GWAP tradicional al ámbito de los dispositivos móviles.

El desarrollo de mGWAP tendrá como objetivo la clasificación de archivos de audio, los cuales serán obtenidos de un repositorio determinado. Dicho desarrollo será similar a lo que proveen los juegos GWAP existentes, con la diferencia que mGWAP se ejecutará sobre dispositivos móviles y su objetivo será la recolección de información semántica para permitir posteriores búsquedas e indexación sobre los archivos de audio del repositorio utilizado.

El desarrollo se dividirá en las siguientes etapas:

- Definición de las estructuras de almacenamiento de la información recolectada por el juego: Definir cuál será la estructura que se utilizara para almacenar las anotaciones recolectadas relacionadas con cada archivo de audio, para permitir la indexación y posteriores búsquedas sobre dichos archivos.

- Aprendizaje de la plataforma de desarrollo del cliente del juego que se ejecutará sobre el dispositivo móvil.

- Implementación del cliente del juego: Análisis de los requerimientos, diseño y codificación del mismo.

- Implementación del servidor del juego: Análisis de los requerimientos, diseño y codificación.

- Implementación de un algoritmo de búsqueda sobre la información recolectada: Desarrollo de un prototipo que permita realizar búsquedas dentro del repositorio de archivos utilizando la información recolectada por el mGWAP.

Resultados esperados

- Desarrollar un juego que permita obtener información semántica sobre archivos de audio a partir del análisis efectuado por sus participantes.
- Construir un juego divertido, popularmente aceptado que permita recolectar gran cantidad de información semántica.
- Mejorar los resultados en las búsquedas realizadas sobre el conjunto de archivos de audio que fueron etiquetados a partir de la información proporcionada por los participantes.
- Mejorar la indexación de la base de datos que contiene los archivos de audio.

Herramientas de desarrollo

El desarrollo de mGWAP estará orientado a dispositivos móviles que utilicen el sistema operativo de Android [5], el cual es soportado por diferentes smartphones de las principales marcas de dispositivos móviles.

Para el desarrollo de mGWAP se utilizará el lenguaje de programación Java [6], dado que este es el lenguaje soportado por el sistema operativo de Android para el desarrollo de aplicaciones.

Las principales herramientas que se utilizarán en el desarrollo son las siguientes:

- Eclipse [7] como IDE.
- PostgreSQL [8] como motor de base de datos.
- Apache Tomcat [9] como servidor de aplicaciones.

- Eclipse ADT [10].

En principio se decidió utilizar Java como lenguaje de programación debido a que es un lenguaje conocido por los integrantes del equipo de desarrollo, además de ser un lenguaje de programación orientado a objetos e independiente de la plataforma. Java será el lenguaje utilizado para desarrollar tanto la aplicación servidor como la aplicación cliente, la cuál será desarrollada sobre la plataforma de aplicaciones móviles Android.

Android es un stack de software para dispositivos móviles que incluye un sistema operativo basado en el núcleo Linux. Inicialmente fue desarrollado por Google y luego por la Open Handset Alliance [11] (liderada por Google). Esta plataforma permite el desarrollo de aplicaciones por terceros (personas ajenas a Google). Los desarrolladores deben escribir código fuente en lenguaje de programación Java a través de la Android SDK [12], aunque también es posible desarrollar aplicaciones utilizando como lenguaje de programación los lenguajes C y C++.

Eclipse IDE es la plataforma de desarrollo utilizada para la implementación de mGWAP, dicha plataforma brinda soporte para el desarrollo de aplicaciones Java, mejorando la productividad de los desarrolladores por medio de sus funcionalidades principales. Eclipse IDE posee una arquitectura de plugins lo que permite agregar diferentes módulos con funcionalidades bien definidas dependiendo de las necesidades del proyecto. Actualmente existe un plugin para el desarrollo de aplicaciones para dispositivos móviles Android, denominado Eclipse ADT, el cuál será utilizado para el desarrollo de mGWAP.

Cabe destacar que todas las herramientas y tecnologías utilizadas en el desarrollo del presente trabajo están publicadas bajo licencias de software libre.

Introducción a mGWAP

Podemos definir a mGWAP como un juego multijugador on-line, el mismo se juega a través de Internet, donde sus participantes descargan el juego en su dispositivo móvil, se registran, y se conectan a Internet a través del mismo para comenzar a jugar.

Al comenzar los usuarios se conectan al servidor del juego y se les asigna una pareja aleatoriamente, si no existiera otro usuario disponible para jugar se le asigna un robot (o "bot") el cual jugará emulando a un participante real (esta situación es transparente para el usuario). Una vez armada la pareja se les presenta un audio a cada jugador, que puede ser o no el mismo, y ambos deben colaborar con el otro ingresando palabras que describan el audio que están escuchando en ese mismo instante.

A partir de las descripciones ingresadas por el otro jugador y el audio que está escuchando en ese momento el participante deberá determinar si ambos están escuchando el mismo audio o no. Si ambos participantes coinciden en la decisión la ronda será ganadora y de esta manera obtendrán puntos a favor.

En todo momento, en el transcurso de las tres rondas de la partida, ambos jugadores tienen en su pantalla la opción de decir "Iguales" o "Diferentes", haciendo referencia a los fragmentos de audio que están oyendo tanto el jugador como el que está oyendo su pareja.

Este comportamiento de los usuarios, de ingresar texto descriptivo del audio que están escuchando en ese instante, proporciona a mGWAP información extremadamente relevante acerca de ese fragmento de audio, a la vez que ayuda a su pareja a decidir la opción que escogerá. Como se explicará en posteriores capítulos, este es el principal objetivo de este tipo de juegos, donde los usuarios realizan el trabajo de describir objetos, principalmente, por medio de palabras o frases, de un modo entretenido.

Además los participantes podrán obtener puntos por medio del bonus, el cual consiste en proporcionarle al jugador tres piezas de audio, luego de escucharlas el jugador debe escoger la pieza que considera más diferente de las tres; si ambos participantes

escogen el mismo audio ganarán el bonus y obtendrán puntos. Esta ronda opcional sólo es presentada al usuario cuando gana todas las rondas de una partida.

El objetivo del bonus es poder obtener una estimación de similitud entre fragmentos de audio diferentes. Esta información generada por los usuarios puede ser utilizada, por ejemplo, en aplicaciones que recomienden temas musicales similares a uno determinado.

Por otra parte, esta información será utilizada por mGWAP para poder internamente decidir que fragmentos de audio proporciona a una pareja de jugadores determinada, dependiendo de su experiencia en el juego, ya que dos fragmentos similares entre sí requieren más esfuerzo para determinar si son iguales o diferentes. En cambio, dos fragmentos que fueron identificados como diferentes serán más fáciles de descubrir por jugadores inexpertos.

Procesamiento de la información

El principal objetivo de mGWAP, es etiquetar la mayor cantidad de fragmentos de audio con la mayor cantidad de palabras posibles. Esto se logra a través de la utilización del juego por parte de los usuarios, cuanta mayor cantidad de partidas se desarrollen, mayor será la cantidad de fragmentos de audio etiquetados.

Esta información recolectada por mGWAP es almacenada, y al final del proceso se obtendrá como resultado un conjunto de palabras descriptivas de cada fragmento de audio, donde además se sabrá en qué partida fue ingresada la palabra y en qué momento del audio.

A partir de este conjunto de palabras, pueden aplicarse distintos algoritmos, como por ejemplo, a partir de una determinada palabra obtener los audios relacionados ordenados por cantidad de veces que dicho audio fue etiquetado con dicho término; o a partir de un audio, determinar los audios similares utilizando las palabras relacionadas con dicho audio.

En próximos capítulos se explicará con mayor detalle que tipos de algoritmos se pueden aplicar al conjunto de palabras obtenidas a partir de la utilización de mGWAP.

Repertorio de audio utilizado en mGWAP

El repertorio de audio utilizado por mGWAP es una porción del utilizado en TagATune [13]. Este es un juego online que recolecta información sobre piezas de audio y música.

Los sonidos utilizados actualmente por TagATune consisten en 56,670 clips musicales de 30 segundos de duración obtenidos de magnatune.com y 28,715 clips de sonido obtenidos de la base de datos de FreeSound (http://freesound.org).

Los clips de magnatune.com son distribuidos bajo la licencia Creative Commons License [14] que permite su uso para fines no comerciales. Precisamente de magnatune.com son obtenidos los archivos utilizados por mGWAP.

El formato en que están almacenados los fragmentos es mp3, con una velocidad de transmisión de 32kbps, usando un solo canal (mono), y una velocidad de muestra de sonido de 16KHz. Dada la baja calidad con la que están guardados se consigue que requieran poca cantidad de almacenamiento, solo 115kb, lo que facilita su transmisión a los usuarios. Los fragmentos no cubren el total de la canción, dado que quedan segundos que no llegan a completar un fragmento y por lo tanto fueron descartados al momento de armar el repertorio.

Para las pruebas realizadas en este trabajo se decidió reducir la cantidad de fragmentos a cincuenta, aproximadamente, tomando los primeros según el orden del repertorio original. El uso de todos los fragmentos para las pruebas es excesivo y no representa ninguna diferencia en la calidad de los datos.

Estructura del trabajo

En el próximo capítulo se analizarán aplicaciones que emplean el concepto de Human Computation y GWAP, y se hará hincapié en el juego TagATune, el cual será referente para desarrollar mGWAP.

El capítulo 3 estará dedicado exclusivamente a describir la aplicación del concepto de GWAP sobre los dispositivos móviles, sus ventajas y desventajas. Como así también el estado actual de desarrollo tecnológico de los dispositivos móviles en cuanto a hardware y software.

Luego, en el capítulo 4 se detallarán las reglas de juego de mGWAP, modelo de asignación de parejas, control de trampas y demás temas relacionados al juego específicamente. Asimismo, se explicará las modalidades posibles de un jugador.

Siguiendo, el capítulo 5 estará centrado en el análisis de las tecnologías y herramientas utilizadas en el desarrollo del trabajo. Siendo los principales temas el lenguaje de programación Java y el sistema operativo de Android.

En el capítulo 6 se describirá en detalle la implementación de la aplicación cliente, sus interfaces visuales, junto con un análisis detallado de las mismas. Como también se describirá el diseño e implementación de la aplicación servidor.

El capítulo 7 estará compuesto del análisis de los resultados obtenidos en las pruebas realizadas, el análisis de diferentes algoritmos de procesamiento de los datos generados por mGWAP y un breve resumen del prototipo implementado que hará uso de algunos de los algoritmos de procesamiento planteados.

Por último, en el capítulo 8 se expondrán algunas conclusiones y se propondrán las líneas futuras de trabajo.

CAPÍTULO 2

Análisis de aplicaciones de Human Computation y GWAP

Introducción

En el presente capítulo se comenzará analizando una de las aplicaciones más destacadas dentro del ámbito de las aplicaciones Human Computation, CAPTCHA [15] la cual permite determinar si un usuario es una máquina o una persona. También se comentará el servicio reCAPTCHA [16], que proporciona CAPTCHAs para ser utilizados de manera gratuita en cualquier sitio web.

Luego se continuará con el análisis de diferentes aplicaciones del tipo GWAP, describiendo las diferentes características de cada una de ellas. Posteriormente se hará una incursión más detallada en el juego TagATune, base del desarrollo de mGWAP.

Por último se verán los conceptos de Input-agreement [17] y Output-agreement [18], dos conceptos fundamentales en lo que se refiere a aplicaciones GWAP.

Una aplicación Human Computation: CAPTCHA

CAPTCHA es el acrónimo de *Completely Automated Public Turing test to tell Computers and Humans Apart* (Prueba de Turing pública y automática para diferenciar a máquinas y humanos). Se trata de una prueba desafío-respuesta utilizada en computación para determinar si el usuario es o no humano. El término se empezó a utilizar en el año 2000 por Luis von Ahn, Manuel Blum y Nicholas J. Hopper de Carnegie Mellon University, y John Langford de IBM.

La típica prueba consiste en que el usuario introduzca un conjunto de caracteres que se muestran en una imagen distorsionada que aparece en pantalla. Se supone que una máquina no es capaz de comprender e introducir la secuencia de forma correcta por lo que solamente el humano podría hacerlo. La Figura 2.1 muestra un típico test para la secuencia "smwm" que dificulta el reconocimiento de la máquina distorsionando las letras y añadiendo un gradiente de fondo.

Figura 2.1. Ejemplo de una imagen de típica de un CAPTCHA

Los CAPTCHAs son utilizados para evitar que robots, también llamados spambots, puedan utilizar ciertos servicios. Por ejemplo, para que no puedan participar en encuestas, registrarse para usar cuentas de correo electrónico (o su uso para envío de correo basura) o, más recientemente, para evitar que correo basura pueda ser enviado por un robot (el remitente debe pasar el test antes de que se entregue al destinatario).

El sistema CAPTCHA tiene las siguientes características por definición:

Es completamente automatizados, es decir, no es necesario ningún tipo de mantenimiento / intervención humana para su realización. Esto supone grandes beneficios en cuanto a fiabilidad y coste. El algoritmo utilizado es público, de esta forma la ruptura de un captcha pasa a ser un problema de inteligencia artificial y no la ruptura de un algoritmo secreto.

No es accesible por personas con deficiencias visuales o auditivas. Además, debido a su naturaleza y misión, algunos asistentes para discapacitados (como los lectores de pantalla) no pueden interpretarlos, quedando bloqueado el acceso al recurso. En algunos sitios se permite elegir entre la validación visual o sonora. En la actualidad, el desarrollo de captchas basados en sonidos está muy por detrás de los visuales y no son tan eficientes.

Hay algunas soluciones que permiten romper CAPTCHA. Usando para reconocerlos humanos como mano de obra barata o involuntaria, un documento de la organización W3C afirma que un operador "puede fácilmente verificar cientos de ellos cada hora". Por otro lado, hay quien afirma que esto es económicamente inviable. Otra manera posible de romper CAPTCHAs es explotando bugs o errores de software en la implementación que permitan a un atacante saltarse el reconocimiento. Finalmente, podrían romperse CAPTCHAs mejorando el software de reconocimiento de caracteres.

Como por ejemplo: ingresar a la máquina que contiene base de datos del problema (captcha) y enviarlos para poder posteriormente ingresarlos.

También los CAPTCHAs son utilizados para vulnerar sitios. Esta técnica consiste en usar un script que muestre un CAPTCHA de un sitio atacado como un CAPTCHA en un sitio que pertenezca al atacante. Usuarios desprevenidos visitan este sitio y resuelven correctamente estos CAPTCHAs que luego serán utilizados por el atacante contra el verdadero sitio.

Howard Yeend[1] ha identificado algunos problemas de implementación con CAPTCHAs pobremente diseñados. Algunos sistemas de protección CAPTCHA pueden evitarse sin usar un sistema de reconocimiento óptico de caracteres (OCR) simplemente reutilizando las respuestas de las imágenes conocidas.

Los CAPTCHAs que están alojados en servidores compartidos también presentan un problema: una incidencia de seguridad en otro alojamiento virtual, podría dejar el sistema CAPTCHA vulnerable.

Algunas veces, si parte del software de generación del CAPTCHA se realiza en el lado del cliente (la validación se hace en el servidor, pero el texto que el usuario tiene que identificar es renderizado en el lado del cliente), los usuarios pueden modificar el cliente para que muestre el texto sin renderizar. Algunos sistemas usan hashes MD5 almacenados en el lado del servidor, que a menudo pueden ser crackeados.

Aunque los CAPTCHAs han sido originalmente diseñados para impedir que el software OCR reconozca los caracteres de las imágenes generadas, existen proyectos de investigación que han probado que es posible saltarse muchos CAPTCHAs con programas que han sido específicamente diseñados para un tipo determinado de CAPTCHA.

Para CAPTCHAs con letras distorsionadas, la solución típica es seguir los siguientes pasos:

[1] Howard Yeend: http://oxford.academia.edu/HowardYeend

1. Eliminación del ruido de fondo, usando filtros de color y detectando y eliminando líneas finas.

2. Segmentación, partiendo la imagen en segmentos que contienen una sola letra.

3. Identificación de la letra de cada segmento.

4. Formar con las letras identificadas en los segmentos la palabra contenida en la imagen.

El paso 1 es típicamente muy fácil de automatizar. En 2005, se mostró que un algoritmo de una red neuronal tiene un menor margen de error que los humanos resolviendo el paso 3.

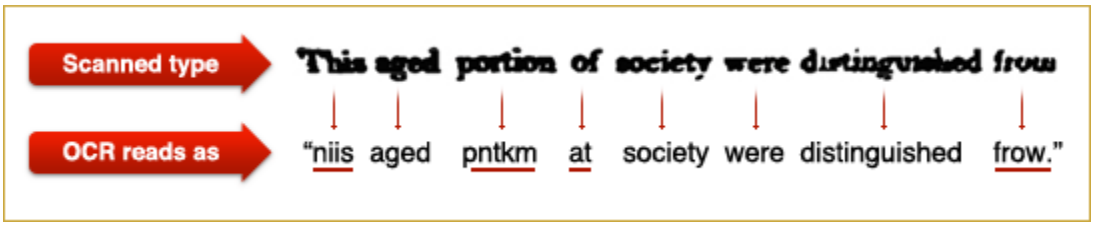
La única parte dónde los humanos superan a las máquinas es en el paso 2. Si el ruido de fondo consiste en formas similares a letras, y las letras están unidas a este ruido, la segmentación se hace casi imposible con el software actual. Por lo tanto, un CAPTCHA efectivo debería enfocarse en el paso 2, la segmentación.

Un servicio proveedor de CAPTCHAs gratuito: reCAPTCHA

El reCAPTCHA es un servicio gratuito de CAPTCHA que ayuda en la digitalización de libros, periódicos y antiguos programas de radio. Cerca de 200 millones de CAPTCHAs son resueltos por humanos alrededor del mundo cada día. Resolver un CAPTCHA lleva aproximadamente diez segundos de tiempo para la persona. Individualmente esto no es mucho tiempo, pero resolver todos esos pequeños puzzles consume más de 150.000 horas de trabajo cada día. ¿Qué sucedería si pudiéramos utilizar ese esfuerzo de manera positiva? reCAPTCHA hace exactamente eso, transformando el esfuerzo de resolver CAPTCHAs en ayudar a los sistemas OCR encargados de la digitalización de textos.

Con el fin de archivar el conocimiento humano y hacer la información más accesible al mundo, múltiples proyectos se encuentran actualmente digitalizando libros escritos antes de la era informática. Las páginas de los libros están siendo escaneadas y luego transformadas en texto usando sistemas OCR. La transformación a texto es muy útil

dado que el escaneo de los libros produce imágenes que dificultan su almacenamiento por una cuestión de espacio, y no permite la ejecución de consultas sobre dicha información. El problema es que los sistemas OCR no son perfectos al momento de trasladar las imágenes a texto. La Figura 2.2 muestra el resultado de la lectura en un sistema OCR de una imagen *scaneada*.



2.2 Ejemplo de una lectura de un sistema OCR.

El reCAPTCHA mejora el proceso de digitalización de los libros enviando a la Web en forma de CAPTCHAs las palabras que los sistemas OCR no pueden resolver para que personas puedan descifrarlos. Más específicamente, cada palabra que el sistema OCR no puede leer correctamente es puesta en una imagen y utilizada como un CAPTCHA. Esto es posible porque la mayoría de los sistemas OCR alertan cuando no pueden leer correctamente una palabra.

Pero si una computadora no puede leer una palabra, como sabrá cuál es la respuesta correcta del CAPTCHA? Cada nueva palabra que no pueda ser resuelta por el sistema OCR será enviada al usuario junto con otra palabra para la cual si se conoce la respuesta correcta. Entonces, se invita al usuario a que ingrese ambas palabras, si el usuario responde bien la palabra para la cual ya se conoce la respuesta el sistema asume que la nueva palabra también es correcta. El sistema proporciona la palabra a más de un usuario para determinar cuál es la respuesta más confiable.

Actualmente reCAPTCHA está ayudando en la digitalización de libros del proyecto Internet Archive[2] y en antiguas ediciones del New York times.

[2] http://www.archive.org

GWAP: Games With A Purpose (Juegos con un propósito)

Combinando la técnica Human Computation y los millones de personas alrededor del mundo dispuestas a invertir horas jugando juegos en línea surge lo que se conoce como GWAP (Game with a purpose, Juegos con un propósito).

El concepto de GWAP podría definirse como juegos en donde cada participante realiza una porción del procesamiento de un computo mayor, el cual es resuelto combinando los procesamientos realizados por cada participante del juego, cuando decimos procesamiento nos referimos al ejercicio mental que realiza el participante para resolver una porción del computo. Ejemplos de GWAP pueden encontrase en los trabajos realizados por Luis von Ahn, los cuales están disponibles en el sitio Web www.gwap.com, también el buscador Google posee en versión experimental de un GWAP para la clasificación de imágenes denominado Google Image Labeler.

Los juegos basados en GWAP motivan a los jugadores a generar anotaciones confiables basándose en incentivos creados por el juego.

En el juego ESP Game [19], es una de las aplicaciones de GWAP. En ESP Game un par de jugadores desconocidos se asocian y cada uno ve la misma imagen, a ambos jugadores se les pregunta "¿Qué está pensando tu compañero?". Dado que ellos no tienen manera de comunicarse, ellos ingresan palabras que tienen algo que ver con la imagen en común que están viendo. Cuando dos personas independientemente sugieren una misma palabra para describir una imagen, la palabra es asumida como confiable. Los juegos GWAP son responsables de recolectar gran cantidad de datos haciendo que el etiquetado sea una tarea entretenida para los jugadores. El juego ESP Game ha recogido más de 10 millones de anotaciones de imágenes. Estos juegos crean un sentido de comunidad y lealtad entre los usuarios y pueden ser altamente adictivos, en el buen sentido. Las estadísticas del juego ESP Game resalta que algunas personas han llegado a jugar más de 40 horas por semana. Dado que estos juegos requieren un mínimo mantenimiento y están online las 24 horas del día, estos juegos están constantemente recolectando nueva información de múltiples jugadores.

Otro juego interesante es el Peekaboom [20], que surge debido a que el ESP Game puede determinar que objetos contiene la imagen, pero no puede determinar su ubicación dentro de la imagen. El juego Peekaboom mejora la información obtenida por el ESP Game obteniendo la ubicación precisa de cada objeto dentro de una imagen. Más específicamente, Peekaboom determina los pixeles perteneces a cada objeto en la imagen.

En este juego, una pareja de jugadores elegidos al aleatoriamente son asignados al rol de "Peek" y al "Boom". Peek comienza con una pantalla en blanco mientras Boom ve una imagen junto con una palabra relacionada. Todas los pares imagen-palabra provienen del ESP Game. La Figura 2.3 muestra la pantalla que observa cada jugador de Peekboom durante un juego.

2.3 Pantalla de Peekboom

El objetivo de “Peek” es descubrir la palabra asociada a la imagen, mientras “Boom” lentamente revela la imagen. Cada vez que “Boom” hace clic en la imagen un área circular de 20 pixeles se le descubre a “Peek” quien debe descubrir cuál es la palabra tipeándola en la caja debajo de la imagen. “Boom” puede ver lo que “Peek” escribe y puede ayudarlo indicándole si se acerca a la respuesta correcta o no. Para ayudar a identificar la palabra “Boom” puede también indicarle a “Peek” si la palabra es un verbo o un sustantivo. Cuando “Peek” descubre la palabra, los jugadores reciben un

cierto número de puntos, luego cambian de roles y obtienen un nuevo par imagen-palabra. Los jugadores cuentan con cuatro minutos para descubrir la palabra.

Intuitivamente, para obtener más puntos, “Boom” tiene un incentivo para descubrir solamente las áreas de la imagen necesarias para que “Peek” descubra la palabra correcta. Por ejemplo, si la imagen contiene un auto y un perro y la palabra asociada es "perro", “Boom” podría revelar sólo la parte de la imagen que contiene al perro. Así, dado un par imagen-palabra, los datos de múltiples jugadas producen el área de la imagen perteneciente a la palabra.

GWAP, etiquetando archivos de audio

Los GWAPs son una valiosa herramienta para la recolección de información semántica. A continuación se estudiará cómo trasladar esta idea al dominio de la música para recolectar información semántica de alta calidad sobre archivos de audio. La recolección de anotaciones semánticas de alta calidad de sonidos y música es una tarea difícil y que consume mucho tiempo. Los métodos manuales de anotación consumen mucho tiempo y son costosos, además no escalan cuando la cantidad de música a anotar aumenta. La información recolectada automáticamente es a menudo inconsistente con la descripción semántica real del contenido del audio.

Antiguos trabajos en la clasificación de música usaban etiquetado manual de los autores o meta datos existentes. Mientras el etiquetado manual generalmente resulta en etiquetas de alta calidad, esto no escala fácilmente a unas cientos de etiquetas por sonido con cientos de sonidos. Existen compañías que emplean docenas de expertos musicales, quiénes trabajan full-time; esto produce etiquetas con gran vocabulario musicalmente relevantes, pero desafortunadamente, tienen poco incentivo para hacer pública esta información. También existen empresas que recolectan un gran número de documentos Web y resumen su contenido usando técnicas de minería de texto. Desde documentos Web asociados a los artistas de las canciones ellos aprenden palabras musicales relevantes asociando las palabras de los documentos con las canciones de los artistas.

Mientras que la técnica de recolección de anotaciones por medio de documentos Web es más escalable que la técnica de etiquetado manual, se observó que los datos recolectados fueron de baja calidad dado que las palabras extraídas no proveen una buena descripción de la canción. En general, cuando se escriben resúmenes de las canciones, álbum o artistas, los autores no hacen una explícita decisión sobre la relevancia de cada palabra. Además, muchos resúmenes contienen información social, histórica u opiniones que no están relacionadas con el audio de la canción.

Una tercera solución, utiliza encuestas para recolectar información semántica sobre las canciones. Los clientes de las empresas que emplean este mecanismo anotan etiquetas de música usando una encuesta estándar que contiene preguntas sobre género, instrumentos, características emocionales, etc. Usado una encuesta se pueden obtener anotaciones de mejor calidad que con la solución Web, pero es necesario pagarle a las personas que se someten al test, por su tiempo invertido. Además, las encuestas consumen mucho tiempo y son tediosas. A pesar de la motivación financiera, las personas sometidas al test rápidamente se cansan de las largas encuestas, resultando en anotaciones erradas.

A continuación se presentan diferentes implementaciones de juegos que tienen el propósito de anotar archivos de audio.

Listen Game

Listen Game [21] es un juego en línea multijugador el cual mide la relación semántica entre la música y las palabras. En modo normal, un jugador ve una lista de palabras relacionadas semánticamente (Ej. instrumentos, emociones, usos, géneros) y se lo motiva a que seleccione la mejor y la peor palabra que describe el audio que está escuchando. En el modo “free style” un usuario sugiere una nueva palabra que describa el sonido. Cada jugador recibe en tiempo real respuestas sobre las elecciones de todos los otros jugadores. La Figura 2.4 muestra la pantalla principal de Listen Game.

La anotación de imágenes a menudo tiene la finalidad de hacer una asociación binaria entre una imagen y los objetos ("Velero"), la información de la escena ("paisaje"), y las características visuales ("rojo") que la representa. El juego Listen Game plantea la subjetividad inherente en la mayoría de las etiquetas semánticas que pueden ser aplicadas a la música permitiendo a los usuarios compartir sus opiniones, en lugar de ser juzgado como correcto o incorrecto. Listen Game recolecta el significado de la asociación entre una palabra y un audio, en vez de todas o ninguna anotación binaria.

En una ronda regular, el servidor del juego selecciona un clip musical de 15 segundos (escogiendo de entre más de 250 canciones occidentales populares) y seis palabras o frases asociadas con una categoría semántica. (Ej. instrumentos, uso, género). Las palabras son escogidas aleatoriamente de un vocabulario predefinido de 174 palabras. El cliente de cada jugador, cargado en un navegador Web, reproduce el clip musical y muestra la categoría y las palabras en un orden aleatorio (para evitar acciones predeterminadas). El jugador entonces escoge la mejor y peor palabra que describen al clip. Una vez que ha escogido, el juego muestra la respuesta de todos los otros jugadores. El puntaje de un jugador "S", es determinado por la cantidad de concordancias entre su elección y la elección del resto de los participantes.

S = 100 * (la fracción de acuerdo con la mejor palabra) + 100 * (la fracción de acuerdo con la peor palabra)

Un jugador juega siete rondas regulares más una “free style” donde el juego reproduce un clip y muestra una categoría. El jugador debe ingresar una palabra o frase apropiada para el clip que está escuchando. En la próxima ronda ordinaria se utiliza el mismo clip, y la palabra es sugerida para que el resto de los jugadores puedan seleccionarla como la mejor o la peor descripción del clip. Utilizando este nuevo tipo de ronda, Listen Game puede automáticamente crecer en su vocabulario de términos musicales relevantes. Sobre el final de la ronda 8, un resumen del juego es mostrado al jugador, con su puntaje, las canciones escuchadas y varias estadísticas del juego.

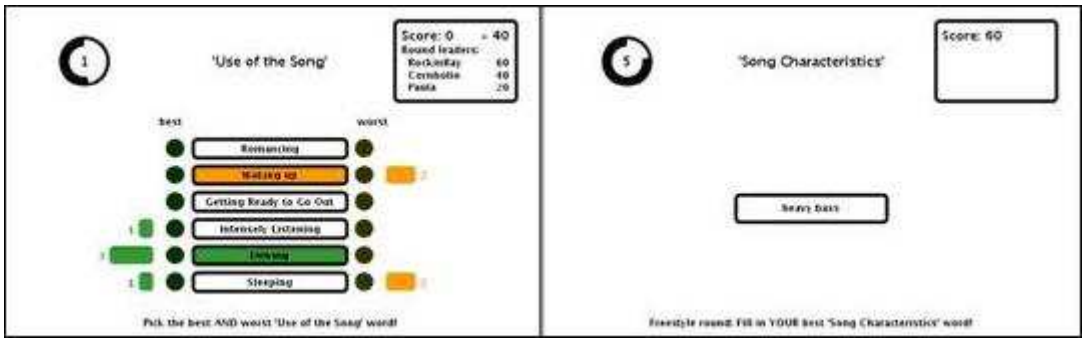


Figura 2.4 Pantalla de Listen Game

Mientras que una mejor/peor elección por parte de un jugador es binaria, el conjunto de asociaciones sonido-palabra no es binario. Uno puede interpretar estas asociaciones con un peso "valor real", proporcional al porcentaje de usuarios que aceptaron una palabra o no para describir un audio. Se calcula el peso semántico *w* como una función de los mejores votos, peores votos y potenciales votos (el número de veces que un par sonido-palabra es presentado a cualquier jugador), la Figura 2.5 muestra las fórmulas de las funciones que representan el peso semántico.

$$w = \begin{cases} 0, & \text{if } \#(\text{Best}) - \#(\text{Worst}) < 2 \\ w', & \text{otherwise} \end{cases}$$

$$w' = \max\left(0, \left[\frac{\#(\text{Best}) - \#(\text{Worst})}{\#(\text{Potential Votes})}\right]\right).$$

Figura 2.5 Función que representa el peso semántico

Para que un par audio-palabra sea confiable, se requiere que al menos dos personas hayan marcado esta asociación en alguna ronda. Se podría esperar que con más información sea posible aumentar el umbral de aciertos significativamente.

MajorMiner

MajorMiner [22] es un juego basado en la Web que recolecta descripciones de extractos musicales. MajorMiner es divertido, fácil, útil y objetivo. Los participantes describen una parte de 10 segundos de una canción, y suman puntos cuando sus descripciones coinciden con las de otros participantes. Las reglas fueron ideadas para

alentar a los jugadores a ser minuciosos y la duración de las partes de las canciones para hacer los juicios más objetivos y específicos.

Figura 2.6 Pantalla principal de MajorMiner

En la Figura 2.6 se puede observar una captura de pantalla de un juego en progreso. El jugador describe un clip de 10 segundos de una canción desconocida. En cursiva se muestra que ha sumado 1 punto, en rojo 0 puntos, y en gris que no tiene puntos, pero sumara 2 puntos si son verificados posteriormente a través de duplicación con otro jugador.

Ejemplo de modo de juego

Un ejemplo de una sesión de juego proporcionara un sentido de sus reglas y estrategia, vea la figura anterior. Primero el jugador, "mim", solicita un nuevo clip para ser etiquetado. Este clip puede ser uno que otros jugadores hayan visto antes o ser uno que es completamente nuevo, el no sabe cuál recibirá. El usuario "mim" escucha el clip y lo describe con unas pocas palabras: lento, arpa, mujer, melancólico, amor y violín.

Las palabras “arpa” y “amor” han sido usadas por exactamente un jugador, entonces cada una de ellas sumará a "mim" un punto. Además, los jugadores que primero usaron esas palabras tienen, en este momento, dos puntos agregados a sus puntajes

(independientemente de si ellos están jugando ahora). Como las palabras “mujer” y “violín” han sido usadas por al menos dos jugadores, sumarán a "mim" cero puntos. La palabra “melancólico” no ha sido usada por ningún jugador, entonces ella no sumará puntos inmediatamente, pero puede sumar dos puntos a "mim" si otro jugador llegara a usarla en otro momento.

Cuando el jugador ha finalizado de etiquetar sus clips puede ir a su resumen de juego, un ejemplo puede apreciarse en la Figura 2.7.

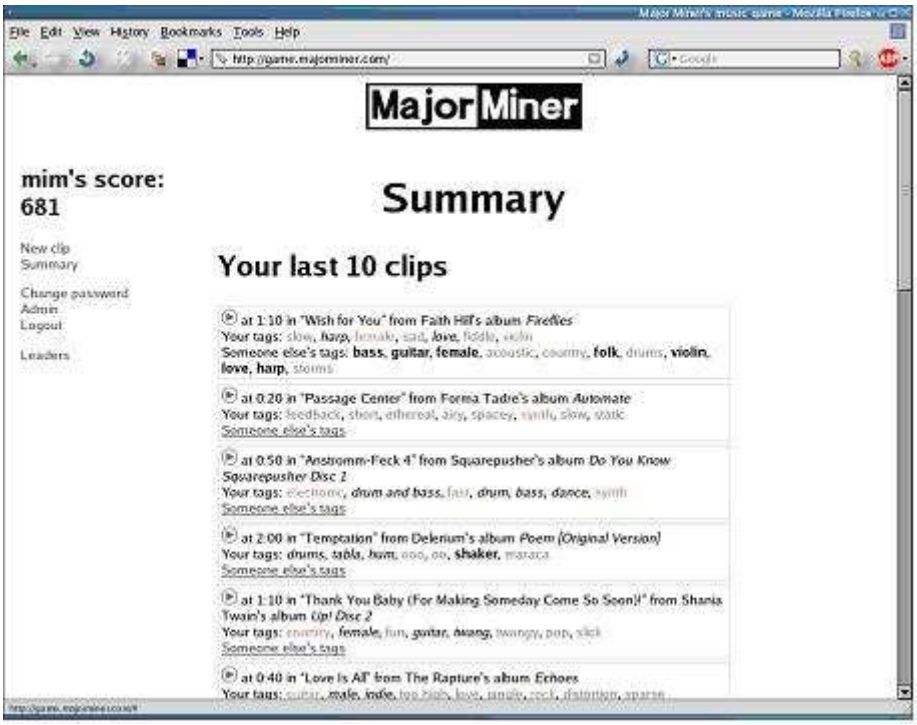


Figura 2.7 Una captura del resumen de juego de un jugador en MajorMiner. El artista, álbum, canción y tiempo de comienzo son listados para los clips que el jugador ha visto o con los que ha sumado puntos recientemente. El jugador también ve las etiquetas de otro jugador.

El resumen de la Figura 2.7muestra los clips que él ha visto recientemente y aquellos en los que ha sumado puntos recientemente, por ej. si otro jugador ha coincidido con una de sus etiquetas. Éste también revela el artista, álbum y nombre de canción de cada clip y permite a "mim" ver las etiquetas de uno de los otros jugadores para cada clip. En la Figura 2.7, el otro jugador ya ha sumado dos puntos describiendo el clip citado más arriba con “bajo”, “guitarra”, “mujer”, “folklore”, “violín”, “amor” y “arpa”,

pero no ha sumado ningún punto todavía por “acústico”, “country”, “batería” o “tormenta”. Cuando termina, “mim” sale de la sesión.

La próxima vez que él entre al juego, el sistema le informara que tres de sus descripciones han sido usadas por otros jugadores en el ínterin, sumándole seis puntos mientras estaba ausente.

La principal idea es incentivar a los jugadores a describir música meticulosamente. Este propósito modeló el diseño de las reglas para dar puntuación a los jugadores. Otro propósito importante, el cual motivo el método para introducir nuevos clips dentro del juego, fue que el juego sea divertido para jugadores nuevos y experimentados. Específicamente, los nuevos jugadores podrían estar capacitados para sumar puntos inmediatamente, y jugadores veteranos podrían ser recompensados con oportunidades de sumar puntos adicionales.

Otros propósitos de menor importancia participaron en la arquitectura e implementación del juego. El primero de estos fue evitar el problema “cold start”. Específicamente, los jugadores serian capaces de sumar puntos tan pronto como sea posible después de iniciar el juego, y un jugador individual sería capaz de jugar en cualquier momento que él quisiera, sin la necesidad de que otros estén jugando al mismo tiempo. Otra fue evitar la posibilidad de hacer trampas, confabulaciones u otras manipulaciones del sistema de puntuación o, peor, la recolección de datos.

El propósito final fue hacer el juego accesible a tanta gente como sea posible, implementándolo como una página web estándar, sin requerir ningún agregado especial, instalación o configuración.

Mientras que muchos juegos forman equipo de un jugador con un solo compañero, MajorMiner forma equipos de un jugador con todos los otros jugadores que han visto un clip en particular. Cuando un jugador es emparentado con un cooperador, es posible que los dos jugadores puedan estar en niveles de habilidad muy diferentes, tienen niveles o familiaridad diferente con el clip bajo consideración, o hasta hablar diferentes idiomas, desalentando el disfrute que tiene cada jugador del juego. También es posible que durante momentos de poca concurrencia un jugador pueda

estar jugando solo (aunque este problema ha sido abordado en otros juegos volviendo a jugar partidas anteriores almacenadas). El formato sin pareja, por otro lado, permite a los jugadores más congruentes, creativos, o expertos cooperar mutuamente asincrónicamente, desde una descripción confusa usada por un jugador quedara disponible hasta que un segundo jugador la verifique.

El diseño de las reglas de puntuación del juego refleja el propósito principal, alentar a los jugadores a describir minuciosamente clips de manera original y relevante. Para fomentar la relevancia, los jugadores sólo suman puntos cuando otros jugadores coinciden con ellos. Para alentar la originalidad, los jugadores obtienen más puntos por ser los primeros en usar una descripción particular en un clip dado. La originalidad es también motivada dando cero puntos por una etiqueta que dos jugadores ya han acordado.

En las reglas actuales, el primer jugador en usar una etiqueta en particular en un clip suma dos puntos cuando este es verificado por un segundo jugador, quien suma un punto. Los siguientes jugadores no suman ningún punto por repetir la etiqueta. Esta asignación de puntos (2, 1, 0) no es necesariamente fija y podría ser cambiada dependiendo de la participación y el índice en el que la nueva música es introducida dentro del juego. El número de jugadores que suman puntos por verificar una etiqueta podría ser incrementado para aumentar el puntaje global y las cantidades de puntos podrían también cambiar para influenciar el estilo general de juego. Se ha observado, sin embargo, que este esquema simple de puntuación satisface suficientemente el propósito de alentar a los jugadores a ser minuciosos. Una inquietud con este sistema es que a lo largo del tiempo los jugadores podrían ser desalentados, si todas las descripciones relevantes ya han sido usadas por otros dos usuarios, ya que no recibirían ningún punto por ellas. Al elegir cuidadosamente cuando los clips son mostrados a los jugadores, es posible evitar este problema y usar la tensión creada por las reglas de puntuación para inspirar la originalidad sin inducir a la frustración.

El juego mantiene tablas de los puntajes más altos, listando los primero 20 puntajes del último día, la última semana y de todo el tiempo. El pago principal del juego puede

ser la satisfacción de alcanzar alguna posición en estas tablas. Incluyendo las tablas con plazos cortos, le da a los nuevos jugadores alguna posibilidad de ver sus nombres.

Eligiendo clips

Cuando el jugador solicita un nuevo clip para describir, se tiene la libertad de elegir el clip de la manera que se desee. Esta libertad permite alcanzar un segundo objetivo: hacer el juego divertido para jugadores nuevos y experimentados. Para que un jugador sume inmediatamente puntos, otro jugador tiene que haberlo visto. Por consiguiente, se mantiene un grupo de clips que han sido vistos al menos una vez y entonces están listos para sumar puntos con ellos. Para los jugadores nuevos, se sacan clips de este grupo para facilitar que sumen puntos rápidamente. Para jugadores experimentados, generalmente se tomarán clips de este grupo, pero a veces se tomarán clips que nunca han sido vistos para introducirlo en el grupo. Mientras que tales clips no permitan a un jugador sumar puntos inmediatamente, ellos ofrecen la oportunidad de ser el primero en usar muchas etiquetas, así suman más puntos cuando otros coincidan.

Mientras los clips deben ser vistos por lo menos por otra persona para permitir sumar puntos inmediatamente, los clips que han sido vistos por muchas personas dificultan sumar puntos.

Revelando etiquetas

Ver las descripciones de otros jugadores es parte de la diversión del juego. También estimula a los nuevos jugadores, con las palabras con las que ellos tendrán una mejor chance de sumar puntos. Estas descripciones pueden sólo ser reveladas después que un jugador ha terminado de etiquetar un clip dado, de otra manera la integridad de los datos y el puntaje podrían ser comprometidos. Con esto en mente, se diseñó una manera para revelar las etiquetas de otros jugadores sin ceder mucha información o creando vulnerabilidades en la seguridad. Es por esto que sólo se revelan las etiquetas del primer jugador que ha visto el clip, una decisión que tiene muchas consecuencias deseables. Esta persona es excepcionalmente identificada sin tener en cuenta cuántos subsecuentes jugadores han visto el clip. En caso de ser la primera persona que etiqueta el fragmento, se revelan las etiquetas del segundo etiquetador (si hay uno).

Como se describió en la sección previa, el primer jugador en etiquetar un clip en particular es probablemente el que tiene más experiencia con el juego. Estas descripciones son buenos ejemplos para otros jugadores, ya que un jugador experimentado generalmente será bueno para describir clips. Así sus etiquetas pueden servir como un buen ejemplo para otros.

También, para evitar introducir un contexto musical adicional que pueda perjudicar al jugador, sólo se revela el nombre del artista, álbum, y canción después que finaliza el etiquetado del clip. Esto focaliza al jugador mucho más en describir los sonidos y evita que la opinión del usuario sobre el artista influya en las etiquetas. Esto también entusiasma en escuchar un clip sin saber el artista y entonces comparar el sonido con la opinión previa que uno tiene sobre el artista.

Estrategia

En el momento que es presentado un nuevo clip, el jugador no sabe que etiquetas han sido aplicadas a él. Intentando con una de las etiquetas más populares revelara cuántas veces esa etiqueta ha sido usada y así el número aproximado de veces que el clip ha sido visto. Si la etiqueta popular nunca ha sido usada o ha sido usada una sola vez, el jugador puede aplicar otra etiqueta frecuente y juntar puntos de manera relativamente fácil. Si la etiqueta ya ha sido usada dos veces, sin embargo, es probable que sea más difícil sumar puntos en ese clip. El jugador debe decidir entonces si es más original o si continúa con otro clip.

Esta estrategia conduce a dos estrategias globales: la primera es ser tan minucioso como sea posible, sumando puntos por coincidir con etiquetas existentes y por usar etiquetas originales que serán verificadas más tarde; coincidiendo con etiquetas existentes, el jugador minucioso junta puntos simples y evita que futuros oyentes sumen puntos con esas etiquetas. Por usar etiquetas originales, el jugador minucioso obtendrá muchos puntajes dobles cuando los siguientes jugadores encuentren el mismo clip. La segunda estrategia es para escuchar tantos clips como sea posible, intentando usar etiquetas populares en los clips que no han sido vistos antes.

Prevención de posibles abusos del sistema

La primera es la confabulación entre dos jugadores o la misma persona con dos nombres de usuario diferentes. Dos jugadores podrían, en teoría, comunicarse entre sí las etiquetas para clips particulares y sumar en todas ellas. Este ataque se frustra haciendo difícil para los jugadores ver clips de su elección y agregando un periodo renuente entre las presentaciones de algún clip en particular. Como los jugadores sólo pueden ver sus clips más recientes, nunca se referencian los clips con un identificador absoluto, sólo por posiciones relativas en las listas de los vistos y puntuados recientemente, haciendo más difícil memorizar que clips han sido vistos.

El otro abuso es el uso repetido de la misma etiqueta o etiquetas en todo clip, a pesar de la música. Esto es fácilmente detectado y las cuentas abusivas son deshabilitadas. Este tipo de conducta puede también ser penada agregando un costo al obtener un nuevo clip o calculando puntajes relativos al número de etiquetas ingresadas, ambos disminuyen el puntaje de un jugador que usa etiquetas sin relación con la música en sí. Para frustrar estos ataques, se podría limitar la frecuencia con que un jugador puede usar una etiqueta en particular, por ejemplo prohibiendo la repetición de una etiqueta del clip anterior.

TagATune y el mecanismo output-agreement

TagATune es un juego online desarrollado para recolectar etiquetas sobre archivos de audio. El diseño original de TagATune usaba el mismo mecanismo output-agreement que ESP Game: dos jugadores recibían una misma pieza de audio y se los invitaba a que ingresen una descripción de dicho audio. Sin embargo, rápidamente quedo claro que esa versión de TagATune podría no ser tan entretenida como lo es ESP Game.

A continuación veremos porque el mecanismo output-agreement que funciona tan bien en el ESP Game falla en la recolección de información sobre clips de audio. Además se expondrá un mecanismo general para la recolección de etiquetas sobre archivos de audio, en la cual está basado el juego TagATune y se describirán las condiciones bajo las cuales este mecanismo es aplicable.

Los juegos Listen Game y MajorMiner están basados en el mecanismo output-agreement, el cual anima a los usuarios a describir un objeto y luego recolecta las descripciones similares para realizar una evaluación posterior, este mecanismo es en cierta manera opuesto al mecanismo input-agreement que se plantea a continuación, mecanismo con el cual está implementado TagATune.

Como se menciono anteriormente, el mecanismo output-agreement usa los resultados coincidentes obtenidos de los usuarios a partir de una entrada determinada, que puede ser una imagen, un audio u cualquier otro objeto que pueda ser etiquetado. En Matchin [23], por ejemplo, dos jugadores ven un par de imágenes y se les pregunta cual prefieren. Ellos son premiados con puntos si su voto coincide. Un *ranking* global de imágenes preferidas puede ser obtenido a partir de la suma de votos. Otro ejemplo es Squigl, un juego para la recolección de segmentos de imágenes. En dicha aplicación a dos jugadores se le muestra una misma imagen y una palabra relacionada a la imagen y se les pide que tracen una línea contorneando el objeto (en la imagen) que describe la palabra. En este caso, los usuario obtienen puntos según cuán concordantes son los objetos contorneados por ambos jugadores. En PictureThis [17] los jugadores ven una palabra y una lista de imágenes, a continuación se les pregunta qué imagen consideran más relacionada con la palabra. En este caso obtienen puntos si coinciden en la elección de la imagen.

El mecanismo output-agreement ha sido también extendido a juegos de extracción de conocimiento, tales como Ontogame[3], en los cuales los jugadores obtienen varios tipos de objetos (ej. extractos de Wikipedia, videos de Youtube, subastas eBay) y una ontología[4], luego se los invita a anotar el objeto dado utilizando la ontología dada. En todos estos juegos, el sistema de premios es el mismo, originalmente introducido por ESP Game, que se basa en premiar a los jugadores cuyas respuestas coincidan.

Problemas del mecanismo output-agreement

[3] http://ontogame.sti2.at/
[4] Una ontología proporciona un vocabulario de clases y relaciones para describir un dominio, poniendo el acento en la compartición del conocimiento y el consenso en la representación de éste.

El principal problema de este mecanismo para recolectar información sobre archivos de audio es que puede resultar muy difícil para dos jugadores coincidir en una descripción. A diferencia de lo que sucede con las imágenes, que a menudo contienen unos pocos objetos identificables, los sonidos pueden ser descriptos por conceptos abstractos tales como "temperatura" (ej. frío, cálido), "animo" (ej. triste, alegre), o la situación que éste invoca (ej. calles congestionadas, festivales, etc.), así como categorías que no tienen límites claramente definidos (ej. Acid-Jazz, Jazz-Funk, Smoth Jazz, etc.).

La dificultad en sonidos no musicales es aún más notoria, donde el contenido no es siempre fácilmente reconocido. Al diseñar un juego que sea divertido la tarea del jugador no debe ser ni muy fácil ni muy difícil.

Como veremos a continuación el mecanismo output-agreement, cuando es usado sobre archivos de audio, puede resultar muy dificultoso y frustrante para los jugadores.

Las estrategias de juego de Listen Game y MajorMiner reflejan este problema subyacente. MajorMiner en lugar de de eso utiliza un mecanismo en donde una etiqueta ingresada por un jugador se la compara con las etiquetas ingresadas por todos los jugadores anteriores, mientras Listen Game utiliza el consenso de un grupo de jugadores sobre un pequeño grupo de *tags* predefinidos.

Otra desventaja de este mecanismo es que, debido a los jugadores juegan solos, esto elimina el aspecto social de los juegos *online* y limita a los jugadores a un grupo reducido de *tags* predefinidos, resultando en un juego significativamente menos placentero y poco útil.

TagATune, el prototipo

La dificultad en la coincidencia de *tags* fue observada durante la prueba del primer prototipo de TagATune, que usaba el mecanismo output-agreement. En el prototipo dos jugadores escucharon clips de audio de 30 segundos de duración y se les pidió que ingresaran descripciones de los mismos. El prototipo inicial sirvió sólo archivos de

sonido, no música. Los jugadores eran premiados cuando coincidían en la descripción. Palabras "Taboo" fueron también utilizadas, para estimular a los jugadores a que ingresaran nuevos *tags*. Aunque el prototipo fue capaz de recolectar una significativa cantidad de información semántica sobre los clips de audio, la media de aceptación en cuanto a diversión fue solo de 3.4 sobre 5, basando en una encuesta proporcionada a 54 participantes. Además, se observo que el 36% de las veces los jugadores optaban por pasar al siguiente clip en lugar de proporcionar una descripción. La Figura 2.8 muestra la pantalla durante un juego en el prototipo de TagATune.

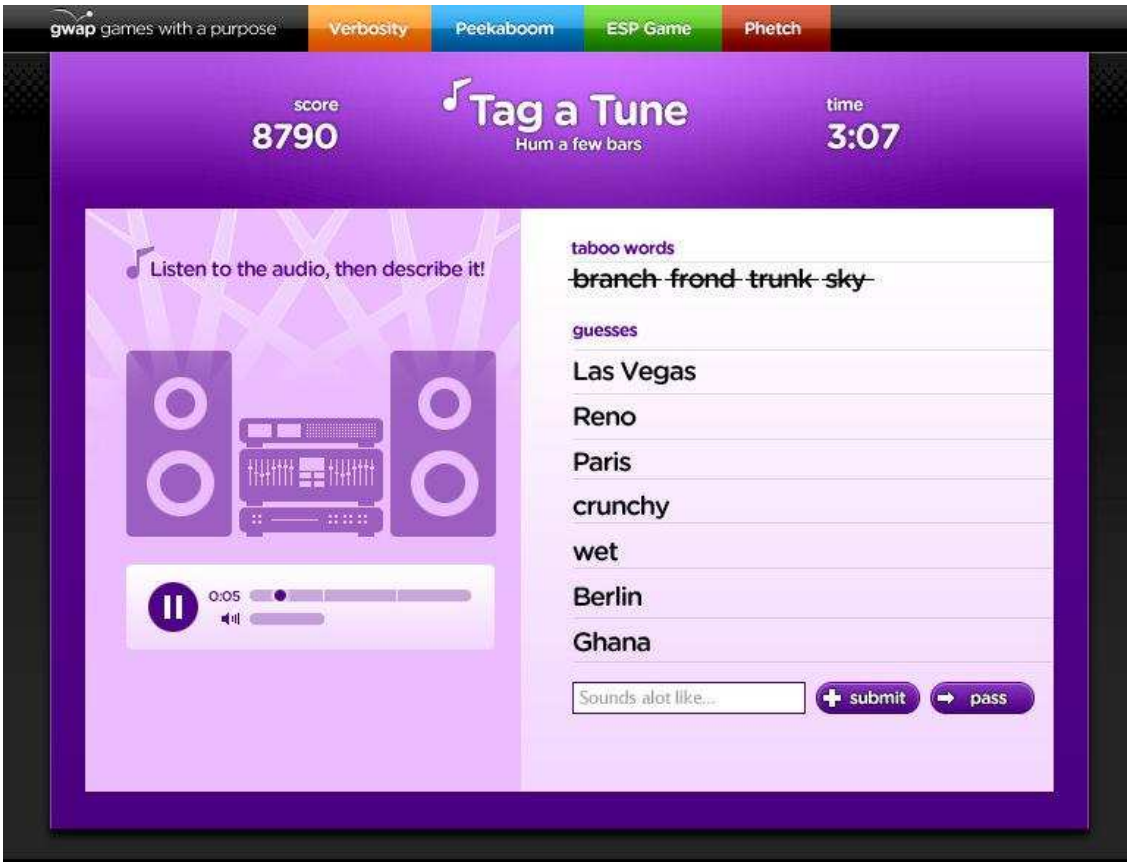


Figura 2.8 Prototipo de TagATune

Input-agreemet, un nuevo mecanismo

Ahora veremos un nuevo mecanismo para la recolección de información usando juegos. En este nuevo mecanismo, dos jugadores pueden observar el mismo objeto o diferentes; luego se les pide que ingresen una descripción de los objetos.

A diferencia del mecanismo output-agreement, donde toda la comunicación entre los jugadores está prohibida, con input-agreement todas las descripciones de un jugador son reveladas al otro. Basándose en estas descripciones, el jugador debe decidir si está observando el mismo objeto que su compañero o no.

Las descripciones que los jugadores ingresan son exactamente lo que el mecanismo busca.

Este mecanismo es denominado input-agreement, en la Figura 2.9 describe el mecanismo general; el cómputo ejecutado por los jugadores consiste en, además de decidir si son o no el mismo objeto, describir dichos objetos.

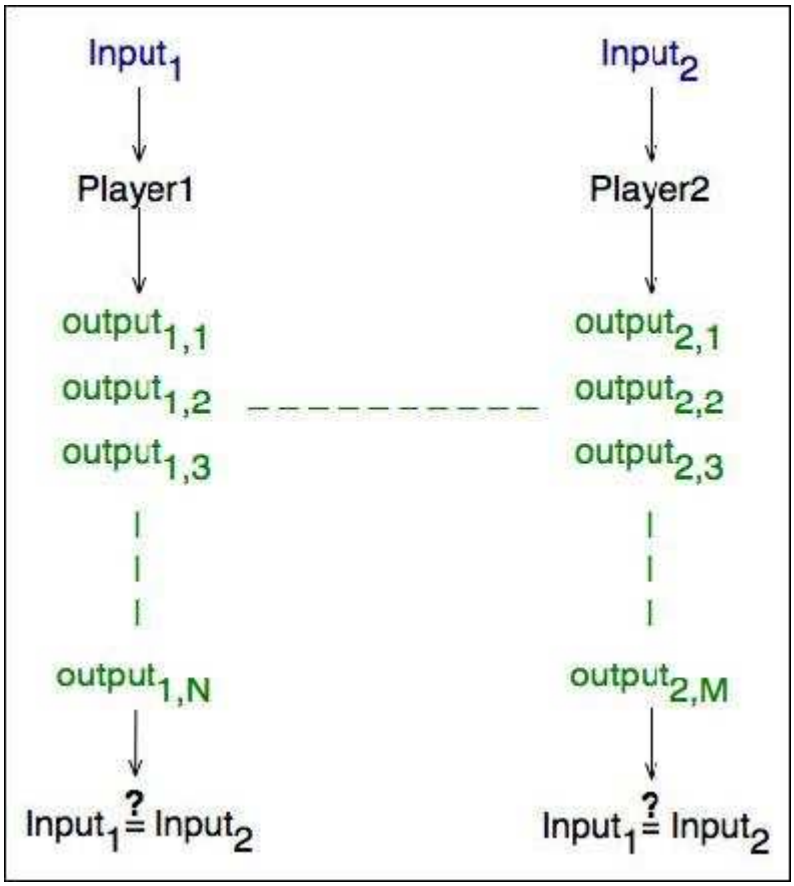


Figura 2.9 Método general del mecanismo input-agreement

La versión actual de TagATune es una instancia del mecanismo input-agreement. A continuación podemos observar un round normal de TagATune. La Figura 2.10 muestra como es la pantalla durante una partida de TagATune.

Figura 2.10 Interface de TagATune

En cada ronda, dos jugadores reciben un clip de audio, que puede o no ser el mismo. Se les provee de una pantalla básica para comenzar a jugar, parar y ajustar el volumen del clip. Cada jugador describe el clip escribiendo (tipeando) palabras, las cuales son visualizadas por el otro jugador. A partir de la descripción hecha por los dos jugadores, cada jugador debe decidir si están escuchando o no el mismo clip. Luego que ambos jugadores votan, el juego muestra el resultado y pasa a la siguiente ronda. El juego dura tres minutos en total.

La inspiración de TagATune (y del mecanismo input-agreement) viene dada por un experimento de psicología que estudia el surgimiento y evolución del sistema de símbolos gráficos. El experimento involucró una tarea de dibujo musical, en la que dos participantes escuchan clips de piano de 30 segundos de duración y luego se les pide que dibujen en una pizarra virtual. Basándose en los dibujos, los jugadores deben decidir si están escuchando la misma pieza de piano o no.

Sonidos utilizados

Los sonidos utilizados actualmente por TagATune consisten en 56,670 clips musicales de 30 segundos de duración obtenidos de http://www.magnatune.com y 28,715 clips de sonido obtenidos de la base de datos de FreeSound (http://freesound.org).

En líneas generales, los géneros de música incluyen clásica, new age, electrónica, rock, pop, música del mundo, jazz, blues, heavy mental, y punk. Todos los clips de audio se encuentran bajo la licencia Creative Commons (CC), permitiendo un uso amplio a diferencia de los licenciamientos restrictivos. La licencia CC permite que los archivos de música sean libremente distribuidos al público. Además, el uso de clips musicales no muy conocidos minimiza la posibilidad que los jugadores reconozcan la canción o el artista y simplemente describan el clip utilizando *tags* que ya son conocidos. Finalmente, la corta duración de los clips asegura que la descripción provista por los jugadores y el clip musical están relacionados.

Por cada ronda, el clip de audio es seleccionado aleatoriamente, esto asegura que los jugadores no se topen con el mismo par de clips muy a menudo.

Mecanismo de puntuación

TagATune es un juego cooperativo, como se puede observar en este mecanismo de puntuación los jugadores suman puntos sólo si ambos aciertan correctamente en si están o no escuchando la misma pieza de sonido. Ninguno de los dos jugadores obtiene puntos si uno de ellos no acierta. Esto provee un incentivo natural para que los jugadores sean sinceros el uno con el otro, y por lo tanto ingresen descripciones lo más acertadas posibles para ayudar a su pareja a responder correctamente. Si TagATune fuera un juego competitivo, cada jugador podría intentar vencer a su compañero, posibilitando esto el ingreso de *tags* incorrectos o maliciosos, lo cual tendría un resultado negativo para el propósito del juego.

En resumen, un juego que utiliza el mecanismo input-agreement debe ser colaborativo.

En la Figura 2.11 se muestra la pantalla con un resumen de todas las partidas del jugador. En la Figura 2.12 se muestra la pantalla donde el jugador puede ver una partida anterior, en donde ve los *tags* de cada jugador y puede escuchar ambos clips.

Figura 2.11 Pantalla Tabla de Puntaje de TagATune

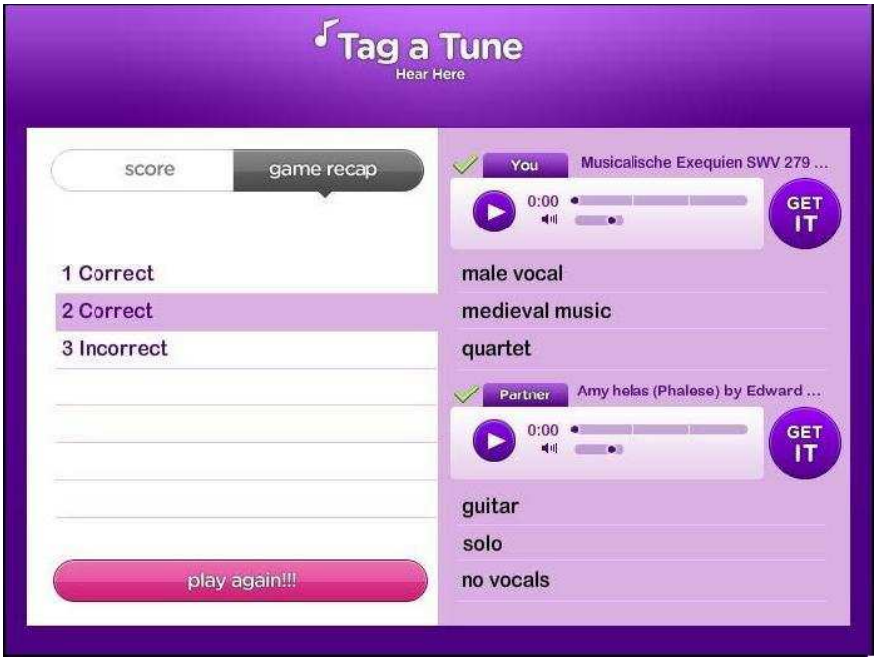


Figura 2.12 Pantalla Resumen de la Partida de TagATune

Bonus

Cuando los jugadores acumulan 1000 puntos, un *bonus round* es añadido en un minuto extra del juego. Durante el *bonus round*, los jugadores escuchan tres clips de música o sonido. Entonces, cada jugador individualmente decide cuál de los tres clips es el más diferente con respecto a los otros dos restantes. Si ambos jugadores eligen el mismo clip obtienen puntos extra. La Figura 2.13 muestra la pantalla de *bonus round* de TagATune.

Figura 2.13 Pantalla bonus round de TagATune

La razón de incluir un *bonus round*, es que produce dos tipos adicionales de información. Primero, la similitud de clips es útil para los sistemas de recomendación. Segundo, la similitud entre clips es un buen indicador del nivel de dificultad que un par de clips podrían presentar durante un *round* normal de TagATune. Más específicamente, dos clips que son similares requieren un gran número de

descripciones, más especificas, para que los jugadores puedan distinguir entre ambos clips. Esta medida de similitud entre clips puede ser utilizada para ajustar el nivel de dificultad del juego y por lo tanto aumentar el interés de los jugadores.

Implementación

El juego TagATune fue desarrollado usando Java y MySQL. El cliente del juego se desarrolló utilizando Adobe Flash, que tiene las ventajas de ser independiente de la plataforma y compatible entre diferentes navegadores de Internet (a diferencia de los Applets JAVA y de la tecnología web Ajax).

Resultados de TagATune

En esta sección se mostrarán las estadísticas de la información recolectada por TagATune en los primeros siete meses desde que se lanzo en Mayo de 2008.

Un total de 49.088 jugadas únicas fueron realizadas por 14.224 jugadores, esto equivale a 439.760 *rounds* normales. El rango de cantidad de juegos que cada persona jugó está entre 1 y 6.286 y el total de tiempo que cada jugador dedico esta en el rango de 3 minutos hasta 420 horas. El promedio de juegos realizados por cada persona es de cuatro. La Figura 2.14 muestra la "cantidad de personas por numero de jugadas". El grafico revela una fuerte tendencia: hay muchas personas que solo han participado en algunas partidas y sólo algunas personas han jugado muchas veces. Esto es un fuerte indicador de los jugadores que re visitan el juego y la frecuencia de estas visitas.

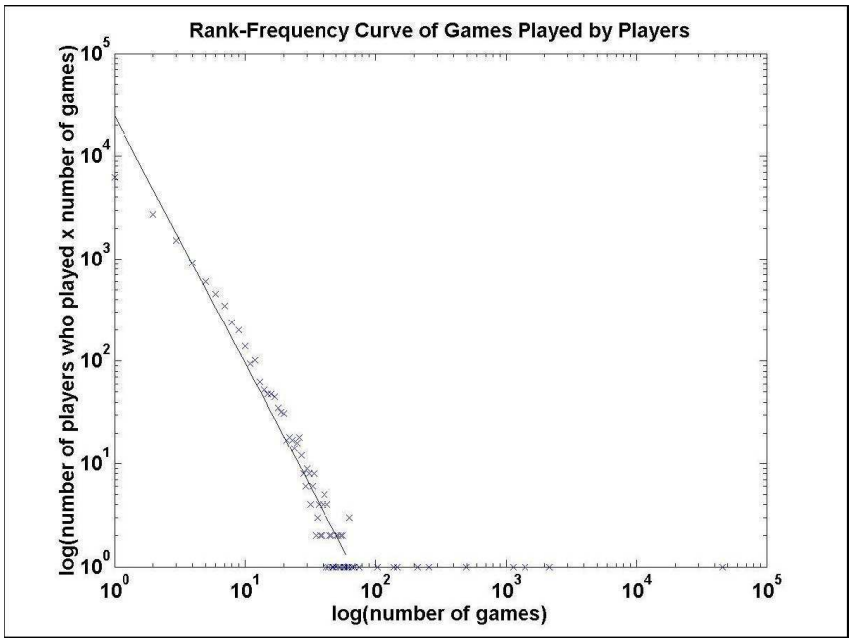


Figura 2.14 Gráfico Estadístico de la Cantidad de Jugadores por Número de Partidas en TagATune. Este gráfico fue extraído del documento Input-agreement[17]

Aciertos

De los 439.760 *round* jugados, los jugadores sólo pasaron sin responder 2.203 *rounds*, esto es el 0,5% del total de *rounds*. En contraste con el 36% que paso sin responder en el prototipo de TagATune; esto indica que los jugadores están más predispuestos a responder con el nuevo mecanismo. En el 97,36% de los *rounds*, ambos jugadores votaron igual o diferente. Se denomina a este tipo de *rounds, rounds* completos. Los 2,64% restantes son llamados *rounds* perdidos, en los que uno o ambos jugadores no emiten su voto, o más probablemente, terminan por tiempo. En la Figura 2.15 se muestra un gráfico comparativo entre los *rounds* completos y los incompletos.

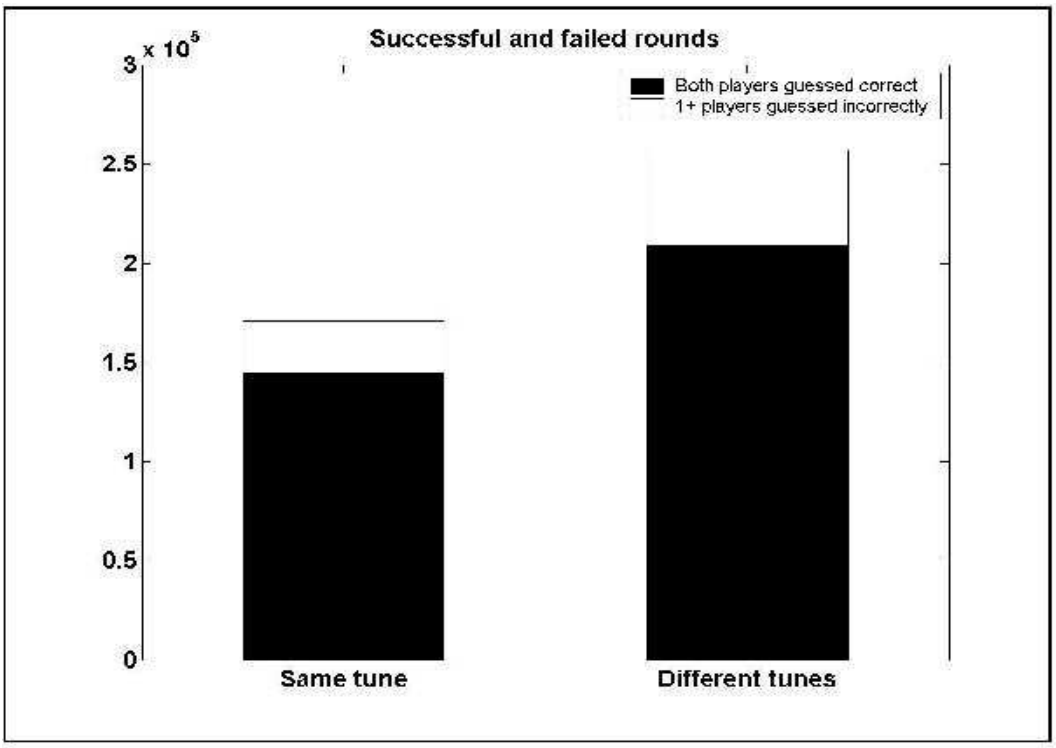


Figura 2.15 Gráfico Comparativo Votos Completos e Incompletos. Este gráfico fue extraído de Input-agreement [17]

De los *rounds* completos, el 80% tuvieron éxito, esto significa que ambos jugadores respondieron correctamente, mientras el 20% restante fallaron, esto es, uno o ambos jugadores respondieron incorrectamente. El ratio de éxito de los *rounds* en los cuales la melodía era la misma fue del 85% mientras que el ratio de éxito en *rounds* donde la melodía era diferente fue del 81%, esto sugiere que puede ser ligeramente más complicado distinguir entre melodías que son diferentes.

Estadística de Tags

Previo a la compilación de las estadísticas sobre los *tags*, un pre procesamiento básico fue ejecutado para convertir todos los *tags* a minúscula, borrar espacios en blanco, y remover puntuaciones. Luego del pre procesamiento quedaron un total de 512.770 *tags*, de los cuales 108,558 fueron verificados por al menos dos jugadores y 70.908 fueron verificados por sólo un jugador. Basándose en esto, la media de *tags* generados por minuto fue de cuatro.

Múltiples niveles de verificación

El mecanismo input-agreement permite múltiples oportunidades de verificar que los *tags* ingresados son de hecho buenas descripciones de un clip de audio. Primero, cada descripción de cada jugador es implícitamente verificada por su compañero durante el juego, esto es, su compañero sólo votara "iguales" si él cree que su compañero ingresó una descripción correcta del clip que él está escuchando. Por otro lado, el jugador votará por "diferentes" si cree que la descripción de su compañero no concuerda con el clip que él está oyendo. En otras palabras, la tarea de descubrir si los jugadores están escuchando el mismo clip o no, es un buen indicador de si los *tags* son apropiados para el clip de audio.

Un segundo nivel de verificación tiene lugar *offline*; luego que la información es recolectada, los *tags* se transforman en "oficiales" si los mismos han sido verificados por un número determinado de usuarios.

Control de fraudes

El control de fraudes es uno de los puntos más importantes en el diseño de juegos basados en Human Computation. En ESP Game, por ejemplo, un par de jugadores pueden hacer trampa si ellos acuerdan una estrategia por medio de la ingresan los mismos *tags* para forzar una coincidencia, sin importar el contenido de la imagen.

Este problema es usualmente solucionado de la siguiente manera, agregando un retardo en el proceso de armado de parejas, esto hace que dos jugadores que ingresan al juego al mismo tiempo tengan pocas chances de formar la misma pareja. Otra manera, es darles a los jugadores imágenes de las cuales ya se conoces *tags*, esto ayuda a detectar posibles tramposos.

CAPÍTULO 3

GWAP sobre dispositivos móviles

Introducción

El presente capítulo estará centrado en la idea de llevar el concepto de GWAP a los dispositivos móviles, en particular los smartphones [24] o teléfonos inteligentes, exponiendo sus características, ventajas y desventajas. También se verán los diferentes tipos de sistemas operativos existentes para dichos smartphones y el estado actual de desarrollo de los mismos. Por último se hará una descripción de las diferentes tecnologías de desarrollo de aplicaciones móviles, es de particular interés para esta tesis el uso del lenguaje de programación Java para la plataforma Android.

GWAP sobre dispositivos móviles

Teniendo presente el concepto de GWAP y aprovechando el apogeo de los dispositivos móviles, como son los smartphones, los cuales han avanzado en tecnología y popularidad de manera asombrosa en los últimos años, podemos pensar en un nuevo campo de desarrollo para los juegos de tipo GWAP.

Los dispositivos móviles actuales permiten a sus usuarios conectarse a Internet, descargar y ejecutar aplicaciones, reproducir archivos multimedia, tomar fotos y videos, entre otras cosas. En la actualidad es común que un usuario se conecte a Internet, descargue un juego y juegue en línea utilizando su dispositivo móvil.

La combinación de GWAP y dispositivos móviles nos brinda la posibilidad de realizar procesamientos de gran escala basados en el concepto de Human Computation, utilizando el tiempo y el esfuerzo de cada usuario de un dispositivo móvil alrededor del mundo que se conecte para jugar a un GWAP.

Hoy en día una persona mientras viaja de su casa al trabajo, se encuentra en una sala de espera, tiene un rato libre luego de almorzar en su trabajo o simplemente tiene un rato libre suele dedicar su tiempo a jugar video juegos, navegar por Internet o chatear utilizando su teléfono móvil. Muchas de estas actividades tienen el único fin de entretener al usuario en un momento de ocio.

Si a estos usuarios se les brindara la oportunidad de utilizar juegos que incorporen el concepto de GWAP se podría obtener una comunidad de usuarios importante. Esta comunidad además de entretenerse en sus ratos libres estaría generando información muy valiosa relacionada a la anotación de archivos de audio.

En la actualidad es posible desarrollar casi cualquier tipo de aplicación sobre dispositivos móviles, esto nos permite pensar en GWAP sobre dispositivos móviles, los dispositivos móviles de hoy permiten el desarrollo de las más variadas y sofisticadas interfaces de usuario, así como también una capacidad de procesamiento interesante, a esto le podemos sumar la mejora en cuanto a interacción con el usuario, las pantallas son más amplias, a color, táctiles, los teclados son alfanuméricos, poseen auriculares y conexiones bluetooth, entre otras cosas). La Figura 3.1 muestra ejemplos de smartphones de diferentes marcas.

Aprovechando las ventajas mencionadas anteriormente es posible desarrollar aplicaciones GWAP que aprovechen las facilidades brindadas por los dispositivos móviles existentes y abran una nueva rama en lo que se refiere a GWAP.

Figura 3.1 Imágenes de Smartphones Actuales

Otro tema importante a tener en cuenta es la distribución de las aplicaciones para dispositivos móviles. Hoy es posible descargar aplicaciones directamente desde el dispositivo móvil utilizando la conexión a Internet o compartirlo entre dispositivos utilizando la tecnologías Bluetooth, entre otras.

Esto hace que la propagación de las aplicaciones para móviles sea extremadamente eficaz haciendo más fácil la difusión.

Ventajas de GWAP sobre smartphones

Se puede decir que la principal ventaja de llevar el concepto de GWAP a los dispositivos móviles es el hecho de cubrir el campo de los juegos de dispositivos móviles y llegar a los usuarios que consumen este tipo de aplicaciones móviles, los cuales hoy podrían consumir los juegos GWAP existentes en la web. Pero se debe destacar que los juegos en la web están desarrollados pensando en computadoras personales y no en dispositivos móviles, siendo las interfaces de usuario del juego poco aptas para ser utilizadas desde un smartphone, por eso es mejor desarrollarlos pensando en que su uso será exclusivo para smartphones.

Desventajas de GWAP sobre smartphones

Hasta el momento se puede decir que la única desventaja, la cual es propia de la capacidad de los dispositivos móviles, es el hecho que las pantallas y teclados sean reducidos y menos ergonómicos que las pantallas y teclados de una computadora personal. Más allá de estas cuestiones inherentes a los dispositivos móviles, en cierta manera podrían afectar al atractivo del juego, y su utilización.

Tipos de conexión de los dispositivos móviles

Actualmente los dispositivos móviles pueden conectarse a Internet por medio de GPRS, Wireless y hasta Bluetooth. Originariamente, las primeras conexiones se efectuaban mediante una llamada telefónica a un número del operador a través de la que se transmitían los datos de manera similar a como lo hace un módem de una PC.

Posteriormente, nació el GPRS, que permitió acceder a Internet a través del protocolo TCP/IP. Mediante el software adecuado es posible acceder, desde una terminal móvil, a servicios como FTP, Telnet, mensajería instantánea, correo electrónico, utilizando los mismos protocolos que una computadora de escritorio. La velocidad del GPRS es de 54 kbit/s en condiciones óptimas y se tarifa en función de la cantidad de información transmitida y recibida.

Luego apareció la tecnología 3G. 3G es la abreviación de tercera-generación en telefonía móvil. Los servicios asociados con la tercera generación proporcionan la posibilidad de transferir tanto voz (una llamada telefónica) como datos (como la descarga de programas, intercambio de email, y mensajería instantánea). Los estándares en 3G utilizan para compartir el espectro entre usuarios, lo que antes se denominaba banda ancha. Se define un ancho de banda mayor, 5 MHz, el cual permite incrementar las velocidades de descarga de datos y el desempeño en general. Aunque inicialmente se especificó una velocidad de 384 kbit/s, la evolución de la tecnología permite ofrecer al suscriptor velocidades de descarga superiores a 3 Mbit/s.

También cabe aclarar que los dispositivos móviles más avanzados tienen la posibilidad de conectarse a las redes Wi-Fi existentes. Este es quizás el punto más importante, sin una conexión a Internet no podríamos concebir una aplicación GWAP.

Tipos de dispositivos móviles

Los dispositivos móviles son aparatos de pequeño tamaño, con capacidad de procesamiento limitada, con conexión permanente o intermitente a Internet, con memoria limitada, diseñados específicamente para una función, pero que pueden llevar a cabo otras funciones más generales.

Existe gran variedad de dispositivos móviles, desde simples teléfonos celulares básicos, los cuales permiten sólo realizar llamadas telefónicas, hasta teléfonos con múltiples funciones como cámara fotográfica, grabación de video, GPS, conexión inalámbrica a Internet, pantalla táctil y otras funcionalidades de última tecnología. Existen también los llamados smartphone o teléfonos inteligentes, los cuales están más cerca del concepto de computadora personal que de teléfono móvil.

Un smartphone es un dispositivo móvil que funciona como un teléfono móvil con características similares a las de una computadora personal; el mismo posee un sistema operativo. Casi todos los smartphones son móviles y soportan clientes de correo electrónico, agendas, navegadores web, aplicaciones de oficina, etc. Una característica importante de casi todos los teléfonos inteligentes es que permiten la

instalación de programas para incrementar sus prestaciones y mejorar sus capacidades. Estas aplicaciones pueden ser desarrolladas por el fabricante del dispositivo o por terceras partes.

Otro tipo de dispositivo que debemos mencionar son las PDA (Asistente Digital Personal) similares a los smartphones, poseen características similares. Es una computadora de mano, originalmente diseñada como agenda electrónica (calendario, lista de contactos, bloc de notas y recordatorios) con un sistema de reconocimiento de escritura. Hoy en día estos dispositivos, pueden realizar muchas de las funciones que hace una computadora de escritorio, entre ellas, es posible ver películas, crear documentos, jugar, usar el correo electrónico, navegar por Internet, reproducir archivos de audio, etc., pero con la ventaja de ser portátil. En cierto modo, los smartphones están reemplazando a las PDAs ya que éstos cumplen con la mayoría de los requisitos de una PDA y además son teléfonos móviles.

A continuación se describen las principales características de los smartphones, ciertos dispositivos pueden no tener alguna de las características mencionadas, esto no significa que no entren en el rango de los smartphones.

Touch screen o pantalla táctil

Las pantallas táctiles sirven para interactuar con el usuario por medio del contacto táctil o utilizando un lápiz desmontable, por lo general estos dispositivos tienen muy pocos botones reservados para abrir los programas más utilizados. Para agregar texto por lo general se usan uno de los siguientes métodos:

- Se usa un teclado virtual, y para agregar las letras hay que tocar cada una de ellas.
- Se puede conectar un teclado externo conectado vía USB o Bluetooth.
- Usando el reconocimiento de letras o palabras y luego traduciéndolas a letras dentro de la caja de texto seleccionada.

- Usando un reconocimiento de símbolos, en el que cierto grupo de éstos representa una letra. Por lo general, estos símbolos son fáciles de recordar.

En la actualidad se pueden ver dispositivos de última generación, los cuales incluyen un teclado conocido como QWERTY, los cuales poseen la misma distribución de teclas que los teclados tradicionales. Por lo tanto, poseen las características touch-screen mencionadas anteriormente y además un teclado físico para que el usuario pueda escoger de que manera ingresar texto en el dispositivo.

Los dispositivos más nuevos como el iPhone o el iPod Touch incluyen una nueva tecnología llamada Multitouch que reconoce simultáneamente múltiples puntos de contacto.

Tarjetas de memoria

Aunque algunos dispositivos no usan tarjetas de memoria, en la actualidad la mayoría de los dispositivos móviles permite el uso de tarjetas SD[5]. Además, muchos de ellos tienen un puerto USB. Por cuestiones de tamaño, ciertos dispositivos ofrecen slots miniSD.

Conectividad por cable

Aunque algunos dispositivos antiguos se conectaban al PC usando un cable serial, en la actualidad la mayoría usan un cable USB o mini USB. Además de permitir la conexión con el computador, sirven como puertos de alimentación de corriente eléctrica en especial el USB. Existen dispositivos que aún hoy siguen conectándose por puertos no estándares implementados por el fabricante.

Conectividad Inalámbrica

Muchos de los dispositivos móviles actuales tienen conectividad Bluetooth; esto permite conectar teclados externos, auriculares, GPS y muchos más accesorios. Además unos cuantos poseen conectividad WiFi, ésta nos permite conectarnos a redes inalámbricas y nos permiten el acceso al Internet. Los dispositivos antiguos disponían

[5] Secure Digital (SD) es un formato de tarjeta de memoria.

además de un puerto infrarrojo, sin embargo muy pocos de los actuales tienen esta tecnología, ya que es muy lento. El infrarrojo permite conectividad entre dos dispositivos o con cualquier otro accesorio que tenga uno de estos puertos.

Sincronización

Una de las funciones más importantes de los dispositivos móviles es la sincronización con las computadoras personales. Esto permite la actualización del directorio, haciendo que la información de la computadora y del dispositivo sea la misma. La sincronización también evita la pérdida de la información almacenada en caso que el accesorio se pierda, sea robado o destruido. Otra ventaja es que se puede ingresar información mucho más rápido desde la computadora y transmitirla luego al dispositivo. La sincronización se realiza mediante un programa que entregan los fabricantes de cada dispositivo móvil.

Sistemas operativos de los dispositivos móviles

Hoy en día, los dispositivos móviles poseen un sistema operativo lo cual le permite al usuario tener una experiencia similar a la que tendría con una computadora personal, además de poder agregar nuevas aplicaciones al dispositivo, a continuación se describen brevemente los sistemas operativos más destacados:

Android

Android[6] es un sistema operativo para dispositivos móviles y computadoras basado en el núcleo Linux. Inicialmente fue desarrollado por Google y luego continuado por la Open Handset Alliance (liderada por Google). Algunas de las principales características de Android son:

- Framework de aplicaciones: permite reutilización y reemplazo de componentes.

[6] http://www.android.com/

- Máquina virtual Dalvik[7]: optimizada para dispositivos móviles.
- Navegador integrado: basado en el motor de código abierto WebKit[8].
- Gráficos optimizados, con una biblioteca de gráficos 2D; gráficos 3D basado en la especificación OpenGL ES 1.0 (aceleración por hardware opcional).
- SQLite[9] para almacenamiento de datos estructurados.
- Soporte para medios con formatos comunes de audio, vídeo e imágenes planas (MPEG4, H.264, MP3, OGG, AAC, AMR, JPG, PNG, GIF).
- Telefonía GSM (dependiente del hardware).
- Bluetooth, EDGE, 3G, y WiFi (dependiente del hardware).
- Cámara, GPS, brújula, y acelerómetro (dependiente del hardware).
- Soporte para Multitouch.

También Android provee a sus usuarios de un servicio web, llamado Android Market, que permite que los desarrolladores publiquen sus aplicaciones, gratuitas o pagas, en el mercado a través de esta aplicación accesible desde todos los teléfonos con Android.

Windows Mobile

Windows Mobile [25] es un sistema operativo compacto, con una suite de aplicaciones básicas para dispositivos móviles basados en la API Win32 de Microsoft. Ha sido diseñado para ser similar a las versiones de escritorio de Windows.

Las versiones de Windows Mobile incluyen aplicaciones de Microsoft Office. Éstas incluyen Pocket Word y Pocket Excel. En Windows Mobile 5.0 se incluye Pocket PowerPoint. Estas versiones incluyen muchas de las características que se utilizan en versiones de escritorio, pero algunas otras características como la inserción de las

[7] http://www.dalvikvm.com
[8] http://www.webkit.org
[9] http://www.sqlite.org

tablas e imágenes no se han incluido versiones anteriores a Windows 5.0. ActiveSync tiene la capacidad de convertir archivos de versiones de escritorio a archivos compatibles con Pocket PC.

Outlook Mobile es también un programa que viene con Windows Mobile. Éste incluye administración de tareas, calendario, contactos, y la bandeja de entrada.

Windows Media Player para Windows Mobile se añade con el software. Windows Media Player reproduce: WMA, WMV, MP3, y AVI. Los archivos MPEG actualmente no están soportados, y se debe descargar un programa de terceras partes para reproducirlos, y los archivos WAV se reproducen en un reproductor por separado. Algunas versiones de Windows Media Player son también capaces de reproducir M4A.

iPhone OS

El iPhone OS[26] es el sistema operativo de Apple que utiliza en el iPod touch y el iPhone, diseñado por 175 ingenieros. Está basado en una variante del Mach kernel que se encuentra en Mac OS X. El iPhone OS incluye el componente de software "Animación Core" de Mac OS X que, junto con el hardware de 3D PowerVR MBX, son los responsables de las animaciones usadas en el interfaz de usuario.

iPhone OS tiene 4 capas de abstracción: la capa del núcleo del sistema operativo, la capa de servicios principales, la capa de Medios de comunicación y la capa de *cocoa touch*. El sistema operativo ocupa bastante menos de medio GB del total del dispositivo, 8 GB o el almacenaje de 16 GB. Esto se realizó para poder soportar futuras aplicaciones de Apple.

Algunas de sus principales funcionalidades son: texto (SMS mensajería), calendario, fotos, cámara, videos YouTube, mapas (Google Maps), tiempo, reloj, calculadora, apuntes, ajustes, e iTunes. Además hay otras cuatro funciones, que se encuentran en la base de la pantalla: teléfono, mail, navegador web Safari e iPod.

Palm OS

Palm OS [27] es un sistema operativo hecho por PalmSource para computadores de mano (PDAs) fabricados por varios licenciatarios. Hoy en día mantenido por Palm, pero que hasta hace poco ha tenido importantes fabricantes como Sony. Sus principales funcionalidades son: libreta de direcciones, calculadora, calendario, manejo de gastos, libreta de notas, lista de tareas.

Hay muchas aplicaciones interesantes para el sistema operativo PalmOS que se pueden añadir. En agosto de 2003, había más de 19,000 aplicaciones disponibles para la plataforma Palm OS, incluyendo software libre como el lector de documentos Plucker o la base de datos Pilot-DB, aplicaciones shareware (demos para probar el programa antes de comprarlo) y aplicaciones comerciales.

Symbian OS

Symbian[28] fue diseñado para integrarse fácilmente, con el software y el hardware en dispositivos móviles. Los sistemas que utilizan Symbian pueden ser divididos en cuatro categorías: serie 60, para teléfonos que el usuario maneja con una sola mano; serie 80, para teléfonos con pantalla horizontal grande y teclado; serie UIQ, para teléfonos que utilizan una pluma electrónica similar a los asistentes digitales; y otros, para futuros desarrollos o que no están en las categorías anteriores.

Symbian es producto del trabajo conjunto de varias empresas líderes en telefonía celular, como Nokia, Ericsson, Motorola y Psion, quienes fundaron esta empresa en 1998 con el objeto de desarrollar un sistema operativo abierto para las diversas plataformas de teléfonos móviles. En 2003 Motorola vendió el 13% de su participación a Nokia, que se hizo con el 32.2% de la compañía Symbian.

El objetivo de Symbian fue crear un sistema operativo para terminales móviles que pudiera competir con Palm y Windows Mobile.

La mayoría de los teléfonos móviles con Symbian son de Nokia. Son muy pocos los modelos de teléfono móvil de otros fabricantes: Sony Ericsson P800/P910, Siemens

SX1 y un par de modelos de Samsung que nunca llegaron a comercializarse (SGH-D700, SGH-D710S)

A diferencia de cualquier computadora, la programación de los teléfonos celulares tiene sus limitaciones. Es por esto que Symbian está diseñado para residir en un espacio muy pequeño, hacer un uso dinámico de escasos recursos de memoria, administrar eficientemente la energía y soportar en tiempo real los protocolos de comunicación y telefonía, además de ser más "gentil" con el usuario y tolerante a fallas, en comparación con un sistema operativo de PC.

Técnicamente, el sistema operativo Symbian es una colección compacta de código ejecutable y varios archivos, la mayoría de ellos son bibliotecas vinculadas dinámicamente (DLL por sus siglas en inglés) y otros datos requeridos, incluyendo archivos de configuración, de imágenes y de tipografía, entre otros recursos residentes. Symbian se almacena, generalmente, en un circuito flash dentro del dispositivo móvil. Gracias a este tipo de tecnología, se puede conservar información aún si el sistema no posee carga eléctrica en la batería, además de ser factible reprogramarse, sin necesidad de separar la memoria flash de los demás circuitos.

Las aplicaciones compatibles con Symbian se desarrollan a partir de lenguajes de programación orientados a objetos como C++, Java (con sus variantes como J2ME), Visual Basic para dispositivos móviles, entre otros, incluyendo algunos lenguajes disponibles en versión libre.

Symbian es actualizable; esta tarea puede realizarla el usuario, dependiendo del modelo de su equipo, a través de los sitios en Internet de los fabricantes de teléfonos o bien, al obtener el disco compacto o tarjeta de memoria flash de los distribuidores autorizados. Adicionalmente, Symbian posee una interfaz gráfica íconica, fácil de comprender por el usuario, permitiéndole explotar las capacidades de su equipo móvil.

Paradójicamente, Symbian también es vulnerable a los virus que afectan a las computadoras personales y asistentes digitales (PDA). La manera más común de contagio es cuando el aparato está en comunicación con algún otro dispositivo contaminado en la red local (si el teléfono es compatible con la norma IEEE 802.11) o

en la red personal (si es compatible con Bluetooth). Menos frecuente es la infección por mensajes cortos (MMS); sin embargo, estos virus pueden bloquear aplicaciones e incluso el sistema de archivos, no obstante, también hay métodos para prevenirlos y eliminarlos instalando programas antivirus.

Tecnologías para el desarrollo de aplicaciones móviles

Aplicaciones para Symbian

Para desarrollar aplicaciones en Symbian es necesario tener conocimientos de programación en el lenguaje C++. Los kits de desarrollo se pueden descargar del sitio oficial de Nokia. Las ventajas que se obtiene programando bajo Symbian es la rapidez de ejecución (sobre todo en juegos), el acceso al sistema, la disponibilidad de música y varios sonidos simultáneamente o la realización de operaciones matemáticas complejas, con lo que se podrán construir juegos de gran complejidad y calidad. La desventaja más significativa es que para cada serie del sistema operativo se debe generar más de un archivo y quizás no todo tu código sea compatible. Otra gran desventaja es que pocos dispositivos móviles soportan Symbian con lo cual se estará desarrollando aplicaciones para un número limitado de terminales.

Java

Java es el más extendido de los lenguajes actuales. Java no es solo un lenguaje de programación sino que implementa una máquina virtual. Esta máquina virtual de Java lo que hace es ser un traductor entre el sistema operativo que utiliza el dispositivo móvil y el código del programa Java que se ejecuta. La principal ventaja de Java es que es compatible con la mayoría de móviles del mercado. La desventaja más destacable es la pérdida de velocidad que conlleva el proceso de interpretación del código ya que tiene que hacer una parada en el traductor para comunicarse con el sistema operativo del dispositivo móvil. Otra desventaja es que no puede aprovechar en su totalidad la capacidad del dispositivo móvil y en consecuencia las aplicaciones siempre estarán limitadas a las posibilidades que ofrece Java y no a las del sistema operativo del dispositivo móvil.

Flash

Flash se está introduciendo cada vez más en el mundo de los dispositivos móviles y actualmente los nuevos modelos ya implementan la tecnología Flash, como por ejemplo los nuevos modelos de Nokia. La principal ventaja de Flash es su simplicidad: programar es muy fácil si lo comparamos con C++ o Java y es posible construir animaciones, películas o juegos fácilmente. La desventaja: los entornos para desarrollar en Flash tienen licencias propietarias. Su velocidad de ejecución tampoco es un punto fuerte.

Aplicaciones para Windows Mobile

Lamentablemente para desarrollar aplicaciones Windows Mobile es necesario disponer de la herramienta profesional de desarrollo (Visual Studio.Net Profesional) ya que las versiones de desarrollo gratuitas ofrecidas por Microsoft (Express Edition) no incluyen estas herramientas. Del sistema se sabe que consume muchos recursos y la interfaz gráfica se asemeja mucho a la versión de PC pero con sus limitaciones. Cabe destacar que los dispositivos móviles con Windows Mobile también soportan aplicaciones hechas en Java.

Aplicaciones para Android

Finalmente el más prometedor de todos es, Android. Para desarrollar aplicaciones para Android se utiliza el lenguaje Java y actualmente soportan el SDK de Google los entornos de desarrollo Eclipse y Netbeans, aunque es posible que con el paso del tiempo otros entornos lo soporten. Las ventajas: para los que programan en Java y sobre todo j2me el código les parecerá familiar, el apoyo y el acuerdo firmado entre Google y varias empresas fabricantes de dispositivos móviles da al desarrollador la seguridad que en poco tiempo este sistema operativo estará implementado en muchos dispositivos, y otra gran ventaja es que a pesar que utiliza una máquina virtual para interpretar el código java, se podrá alcanzar a utilizar todo el potencial del dispositivo. Las desventajas: Nokia, principal fabricante de dispositivos móviles, no forma parte de este consorcio pero no ha cerrado la puerta a Android, seguramente porque será su rival más fuerte.

El futuro de los dispositivos móviles según el líder del proyecto Android

Andy Rubin, líder del proyecto Android de Google, deja sus apreciaciones[10] sobre el futuro de los dispositivos móviles.

"El celular que tienes en tu bolsillo, es probablemente más poderoso que la computadora que tenías en tu escritorio hace 8 o 9 años (asumiendo que tenías una PC ya que la mayoría de los usuarios nunca tuvieron una)", afirmó el jefe de la división de dispositivos móviles de Google. También afirmó que actualmente "Hay 3.200 millones de usuarios de celulares en todo el mundo, más que la cantidad de autos y tarjetas de crédito juntas.

"El celular es el producto de consumo más extendido que se haya inventado", lanzó Rubin y luego aseguró que la mayoría de los celulares tienen una cantidad de sensores similares a la de un vehículo de exploración marciana. Entonces, ¿Qué vamos a hacer capaces de hacer con nuestros celulares en 10 años?

- Alertas inteligentes: hoy en día ya pasa con alertas de productos o noticias, pero serán más inteligentes y personalizadas para entregarte la información relevante.

- Realidad aumentada: el celular entenderá tu contexto y te brindará información sobre él.

- Crowdsoursing: todo el contenido como fotos y reviews de los usuarios estará disponible desde los móviles para consultarlo. Podemos ver comentarios que dejaron sobre un restaurant, por ejemplo.

- Sensores en todas partes: los celulares pueden captar información del mundo y eso se puede compartir para mejorar los pronósticos del clima o tráfico por ejemplo.

[10] http://googleblog.blogspot.com/2008/09/future-of-mobile.html

- Herramienta para el desarrollo: los móviles son fundamentales para muchos oficios en el mundo ya que pueden brindar información para poder cosechar, precios de productos en los comercios y demás.

- Dispositivos a prueba del futuro: con actualizaciones automáticas de las aplicaciones, los celulares mejoran constantemente.

- Software más seguro mediante la confianza y la verificación: el celular ayudará a mantener el control de la información del usuario, eligiendo qué compartir y qué no.

Conclusión

Se llegó al final del capítulo y como reflexión podríamos decir que el ámbito de los dispositivos móviles, los smartphones más precisamente, es un lugar muy propicio para el desarrollo de juegos del tipo GWAP. Teniendo en cuenta los avances tecnológicos en el área, el desarrollo de mGWAP (mobile GWAP) es una alternativa innovadora ya que combina el concepto GWAP con la utilización de juegos en los dispositivos móviles.

CAPÍTULO 4

Modo de juego, reglas y mecanismos de mGWAP

Introducción

En el presente capítulo se explicará el modo de juego de mGWAP, junto con sus reglas y los mecanismos empleados para diferentes tareas, como por ejemplo la prevención de trampas, selección de las parejas y la modalidad de un solo jugador.

Descripción del juego

Como se explicó en capítulos anteriores mGWAP es un juego multijugador on-line, desarrollado para que se pueda jugar a través de Internet, donde sus participantes se pueden conectar utilizando sus dispositivos móviles, más precisamente smartphones compatibles con el sistema operativo de Android.

Para comenzar a jugar mGWAP un jugador debe registrarse la primera vez que ingresa. De esta manera el jugador podrá acumular puntos durante las diferentes partidas en las que participa y mantener un perfil dentro del juego, con sus datos e historial de juego.

Esto, a su vez, habilitará diferentes modos de juego para determinados usuarios, quienes obtendrán estas posibilidades dependiendo de su desempeño en las diferentes partidas y de la cantidad de puntos que vayan acumulando a lo largo de su vida dentro del juego.

Cuando un usuario ingresa al juego, luego de registrarse, se identifica y se le asigna una pareja, más adelante en este capítulo se explicará la manera en que se determina la pareja para cada usuario. Dado que en este tipo de aplicaciones no se puede tener la seguridad de que el usuario está prestando atención a lo que sucede, a menos de que éste lo indique, una vez seleccionada la pareja se le solicita a ambos que confirmen si se encuentran preparados para iniciar la partida. En caso que uno de los usuarios demorara más tiempo del estipulado en confirmar su presencia, se les notificará a ambos usuarios la cancelación de la partida. Luego que ambos usuarios confirman que están listos para jugar se inicia la partida.

Una vez iniciada la partida se les presenta un audio a los participantes y ambos deben colaborar entre sí ingresando palabras que describan el audio que están escuchando en ese instante de tiempo. A partir de las descripciones ingresadas por ambos jugadores, cada participante deberá determinar individualmente si ambos están escuchando el mismo audio o no. Si ambos jugadores coinciden en la decisión y ésta es la correcta, obtendrán puntos a favor. El objetivo del juego es obtener la mayor cantidad de puntos.

Selección de parejas

Al comienzo de cada partida se asignan las parejas, éstas son desconocidas para los participantes durante el juego. A continuación se estudiarán los diferentes mecanismos pensados para la asignación de las parejas, entre ellos el que finalmente fue utilizado.

En una primera instancia se pensó en armar parejas a medida que entraban los usuarios. De esta manera se reducen los tiempos de espera y no se realiza demasiado procesamiento que pudiera perjudicar el rendimiento de la aplicación. Pero este mecanismo presenta varias desventajas. En primer lugar, da la posibilidad que dos personas puedan arreglar las partidas previamente para jugar entre sí, tratando de obtener alguna ventaja o de perjudicar al juego. La segunda desventaja es que se asigna parejas sin contemplar la experiencia de las mismas, por lo tanto se podría asociar a una persona que está jugando sus primeras partidas con alguien de mediana experiencia, lo podría generar descontento para ambos.

El siguiente mecanismo que se pensó consiste en armar parejas teniendo en cuenta los puntos acumulados de cada usuario, seleccionando usuarios con puntaje similar. Esto se realiza poniendo límites a las diferencias de puntajes permitidas, por ejemplo, si un usuario acumula 500 puntos y el límite permitido para una partida es de 100 puntos, este usuario podrá jugar con parejas cuyos puntos acumulados estén entre 400 y 600. La principal desventaja de este sistema está en la demora que puede tener un usuario si no ingresa nadie dentro de su rango de puntos, algo bastante posible si la cantidad de usuarios es reducida. Para solucionar ese problema se decidió que el límite se vaya

incrementando a medida que pasa el tiempo, con lo que se consigue un rango mayor y aumenta la posibilidad que algún usuario coincida. Además, se introdujo un límite de tiempo en la espera de un usuario que desea jugar, si se cumple dicho plazo y no se le consigue pareja, se le asigna una pareja aunque no cumpla con las características. Si se diera el caso de no haber otro usuario en espera, se crearía una partida en modo de un solo jugador para que el usuario no se quede esperando indefinidamente. Este sistema presenta otra desventaja, los usuarios que recién comienzan a jugar serán asignados entre sí, esto puede desalentarlos a jugar ya que ambos ingresarán pocas etiquetas, algunas posiblemente erróneas, producto de la poca experiencia, haciendo que sea más difícil acertar. Para solucionar ese problema se modificó el mecanismo. Se determinó que los jugadores con pocas partidas jugadas tuvieran prioridad para ser asignados con los más experimentados, de esta manera se aumentan las posibilidades que tienen de ganar las partidas y así crear un interés mayor por el juego. Para conseguir esto se incluyó un plazo de espera, en el cual se busca al jugador más experimentado y se lo asigna con el jugador novato.

De esta manera se arman las parejas que jugarán las partidas. Luego de esto se crean las rondas seleccionando los fragmentos de audio.

Características y selección del fragmento de audio

La partida consta de tres rondas idénticas. En cada una de ellas se le proporciona un fragmento de audio a cada jugador, pudiendo ocurrir que ambos jugadores reciban el mismo fragmento o no. Todos los fragmentos tienen una duración de 30 segundos. La duración del clip responde a varios motivos.

El primer motivo es meramente tecnológico, si bien la velocidad de transferencia de los dispositivos móviles ha aumentado en los últimos años al punto de permitir realizar una gran variedad de actividades a través de Internet, como ver videos, chatear o recibir e-mails, esta velocidad depende mucho de, por ejemplo, la zona en que se encuentra el usuario o la saturación de la antena que maneja su señal. Por lo tanto si un usuario tuviera que esperar la descarga de un tema completo antes de iniciar cada

ronda de la partida, dicha espera resultaría aburrida y molesta en la mayoría de los casos debido a su lenta conexión a Internet.

El otro motivo por el cual los fragmentos son de corta duración, éste de índole práctico, es que las piezas de audio podrían variar mucho en su duración, mientras que unas duran sólo un par de minutos, algunas pueden durar hasta 20 minutos o más. De esta manera una partida podría extenderse demasiado en terminar, además de variar mucho el tiempo de duración entre diferentes partidas, algo que resulta molesto para un jugador ya que no podría saber a priori el tiempo que le tomará completar su partida.

Por último, acortar el tiempo de los clips que se presentan a sólo 30 segundos provoca que el usuario ingrese etiquetas más específicas y relacionadas con la melodía o sonido que está escuchando en ese preciso instante. También intenta evitar que el usuario reconozca la canción y que introduzca etiquetas relacionadas con el autor o el nombre del tema.

A continuación se estudiará la mecánica utilizada para la selección de los clips que los usuarios escucharán en cada una de las rondas de la partida. Se debe destacar que el mecanismo a utilizar contempla la motivación del usuario para participar en el juego, como así también la calidad y cantidad de etiquetas que cada archivo de audio obtendrá de los jugadores con este mecanismo.

El primer punto que se mencionó hace referencia a la idea de motivar al usuario a que continúe participando en el juego, armando rondas con nivel de dificultad bajo y nivel de dificultad alto, de manera intercalada, dándole altas chances de ganar en determinadas rondas, y lo exija intelectualmente en otras rondas, haciendo que el mismo ingrese mayor cantidad de etiquetas y de mejor calidad. Este nivel de dificultad está directamente relacionado con el nivel de similitud que existe entre un par determinado de clips de audio. El segundo punto que se tiene en cuenta es el hecho de obtener más y mejores etiquetas para los archivos de audio. Una pareja de jugadores cualquiera eventualmente ingresará más y mejores etiquetas en situaciones donde los clips que está escuchando tienen un nivel de similitud alto, en cambio, si los clips tienen un nivel de similitud muy bajo, las etiquetas que ingresarán serán menores en

cantidad y calidad, dado que serán necesarias pocas etiquetas para que ambos se percaten que están escuchando clips distintos. El mecanismo en una primera instancia debe decidir si ambos jugadores escucharán el mismo clip o no, esto se decide de manera aleatoria con un 50% de probabilidades para cada caso, y se evalúa para cada ronda independientemente de las anteriores. Se hace de esta manera debido a la cantidad de fragmentos que posee la base de datos (de miles de fragmentos), si se eligiera el segundo fragmento independientemente del primero en todos los casos, la mayoría de las rondas tendrían diferentes clips debido a las pocas posibilidades que habría que el mismo fragmento fuese elegido dos veces, por lo tanto los jugadores podrían elegir siempre que el clip es diferente con muy altas posibilidades de acertar sin necesidad de intercambiar datos con los otros usuarios.

En el caso que se decida que ambos jugadores serán provistos del mismo clip, el mecanismo finaliza en esta instancia y comienza la ronda.

En el caso que se decida que cada usuario obtendrá un clip diferente, el mecanismo entrará en su segunda etapa, decidir el nivel de similitud que habrá entre ambos clips. Esta decisión estará determinada por la ronda anterior que jugaron ambos usuarios, esto es, si en la última ronda en la cual participaron ambos usuarios el nivel de similitud fue alto, entonces el nivel de similitud de la próxima ronda será bajo, caso contrario será alto.

Si se da la situación que es la primera vez que ambos usuarios juegan en pareja se tomará como referencia la última ronda jugada por el usuario con más puntaje, beneficiando en cierta medida a los jugadores más activos del juego.

La selección del primer clip se hará teniendo en cuenta la cantidad de veces que dicho clip fue utilizado. Esto se debe al objetivo principal del juego mGWAP, que es etiquetar todos los archivos de audio del repositorio. De esta manera se les da prioridad a los clips que menos etiquetas tengan.

Lo mencionado en el párrafo anterior puede provocar que en las primeras instancias de la vida del juego se diera el caso que no exista información de similitud de un clip

determinado con el resto. En este caso el segundo clip será seleccionado de manera aleatoria.

El nivel de similitud de los clips estará determinado por la información ingresada por los jugadores tanto en las partidas ordinarias como en el bonus.

Mecanismos alternativos que se tuvieron en cuenta:

Previo a la elección del mecanismo de selección explicado anteriormente, se estudiaron dos mecanismos alternativos, los cuales fueron descartados por cuestiones que mencionaremos a continuación.

El primer mecanismo considerado fue el de la selección según la experiencia de los usuarios, éste mecanismo se basa en la experiencia de ambos jugadores para determinar el nivel de similitud que tienen dos clips de una ronda determinada. El mecanismo fue descartado ya que presentaba una desventaja importante, la cual podría impactar negativamente en el interés de los usuarios por el juego. El problema fue el siguiente, si a un usuario con un nivel de experiencia medianamente avanzado se le presentaban los clips utilizando este mecanismo, siempre participaría de rondas donde el nivel de similitud de los clips es alto, esto podría impactar negativamente sobre el juego ya que eventualmente el usuario se aburriría del alto nivel de exigencia del juego y abandonaría. Es por esto, que se implemento el intercalado de niveles de dificultad.

El segundo mecanismo considerado fue el de seleccionar el par de clips según el desempeño de ambos participantes en su historial de partidas, este mecanismo el cual se caracteriza por aumentar el nivel de exigencia en cada ronda evaluando el desempeño previo de ambos participantes también sufría del problema del mecanismo mencionado anteriormente, el aumento progresivo en el nivel podría provocar el aburrimiento del usuario y su posterior abandono del juego.

A lo comentado en los párrafos anteriores, se puede agregar que ambas alternativas tenían otra desventaja en común, y es que los usuarios no siempre tienen el mismo nivel de experiencia en el juego y por lo tanto lo que podría resultar de dificultad

moderada para un jugador de la pareja podría resultar de un nivel muy exigente para el otro usuario de la pareja.

Descripción de la partida

Una vez que cada usuario tiene listo su clip se inicia la ronda, se espera a que ambos usuarios tengan el clip completamente descargado en su dispositivo para iniciar la ronda dado que ambos deben escuchar el tema al mismo tiempo, ya que ambos deben cooperar en tiempo real para descubrir si están frente al mismo clip o no, además la duración del clip determina el tiempo que tiene el usuario para ingresar etiquetas y determinar si está escuchando el mismo fragmento que su compañero.

Con el comienzo de la ronda se inicia la reproducción del clip, durante la misma el jugador deberá ingresar palabras que representen lo que está escuchando. A medida que ingresa las palabras, éstas son enviadas a su compañero y mostradas en un sector de la pantalla, de esta forma ambos jugadores reciben en tiempo real las palabras que está ingresando su compañero. El jugador también tendrá a la vista en todo momento las palabras que él mismo ingresó. El plazo de tiempo que tiene el jugador para ingresar palabras finaliza cuando termina el clip.

Una vez que el clip ha finalizado se le dará a los jugadores una cantidad determinada de segundos para que decida si el fragmento que escuchó es el mismo que escuchó su compañero. Los jugadores sumarán puntos en caso que coincidan en la opción elegida y que acierten con la respuesta correcta. De esta manera si alguno, o ambos, no aciertan en la respuesta, ninguno de los dos obtendrá puntos en dicha ronda.

Se decidió no darle puntos al jugador que acierte en la respuesta aunque su compañero no lo hiciera, dado que el objetivo del juego es lograr que sea colaborativo y no competitivo, de esta manera cada jugador deberá preocuparse tanto por acertar en su respuesta final, como en describir lo mejor posible su clip para aumentar las posibilidades que su compañero también acierte y de esa manera ambos sumen puntos.

Un jugador podría decidir si está escuchando el mismo clip o no que su pareja antes de que dicho clip finalice, esto bloquearía el ingreso de palabras con lo que reduciría la posibilidad de acierto, tanto suya como de su compañero. Aunque esta opción puede provocar que el jugador ingrese menos palabras, nótese que en estos casos se agrega un dato nuevo, el instante de tiempo en que decidió. En algunos casos hay etiquetas que son totalmente opuestas al clip que un jugador está escuchando, en esos casos el jugador podría decidir que no necesita más palabras para tomar su decisión, por lo tanto en un procesamiento posterior de la información se podrían determinar palabras que son potencialmente opuestas al clip sin ningún tipo de dudas. En todo momento, durante la ronda, ambos jugadores saben si su compañero ya escogió su respuesta o no.

Una vez que ambos jugadores eligieron su respuesta, se muestra en pantalla el puntaje que reciben por la ronda y también se muestra si la respuesta de cada jugador fue correcta o incorrecta. En caso que un jugador no elija una respuesta, el sistema asumirá que la respuesta es incorrecta y por lo tanto ninguno de los jugadores sumará puntos.

Provocar que el juego sea colaborativo y no competitivo, otorgando puntos sólo si ambos jugadores aciertan, provee un incentivo natural para que los jugadores ingresen datos que describan correctamente el clip de audio. Si por el contrario, fuese un juego competitivo, por ejemplo otorgando puntos al jugador que acierte aunque su compañero no lo hiciera, cada jugador se vería motivado a ganar perjudicando a su pareja, de este modo ingresaría datos incorrectos y maliciosos para despistar al otro jugador y así provocar su error. Esto desembocaría en demasiados datos incorrectos producto de la competencia implícita del juego.

Puntuación

La puntuación de los jugadores, que como ya se dijo recibirán la misma cantidad de puntos, se asigna de la siguiente manera:

Al primer acierto, es decir, cuando ambos jugadores contestan de manera correcta, se le dará 60 puntos a cada jugador. Al segundo acierto recibirán 70 puntos cada uno y en caso de tener un tercer acierto recibirán 80 puntos. De esta manera una pareja de jugadores que acierten en los tres fragmentos recibirá 210 puntos cada uno. Los aciertos no necesitan ser consecutivos, es decir, si una pareja acierta el primer fragmento (por lo que recibe 60 puntos), falla el segundo fragmento y vuelve a acertar en el tercer clip, recibirá 70 puntos por ser el segundo acierto de la partida, de manera que totalizarán 130 puntos por la partida.

Con este sistema de puntuación se intenta estimular la atención del usuario a lo largo de toda la partida, ya que un usuario que acierta las tres rondas de una partida recibe más puntaje que un usuario que acierta 3 rondas en diferentes partidas. Por ejemplo, un usuario que juega una partida y acierta las 3 rondas recibe, como se explicó, 210 puntos. En cambio, un usuario que acierta 3 rondas, pero en 3 partidas diferentes, solo recibe 180 puntos. De esta manera, el usuario que mantiene un buen desempeño durante toda una partida recibe más beneficios.

Control de trampas

Se deben dejar en claro dos puntos. Primero, el motivo fundamental que llevaría a un jugador a hacer trampa es la posibilidad de aumentar su chances de sumar puntos o de perjudicar a sus oponentes. Y segundo, la finalidad del juego es recolectar etiquetas, por lo que las trampas que perjudican al juego son aquellas que generan etiquetas erróneas.

Aclarados ambos puntos veremos que un jugador no puede obtener ventajas al ingresar etiquetas erróneas intencionalmente. Esto es así por una razón muy simple, durante una partida (el único momento en que el jugador puede ingresar etiquetas) ambos jugadores deben colaborar entre sí para poder obtener la mayor cantidad de puntos posibles. Por lo tanto, si un jugador ingresa etiquetas falsas para tratar de inducir a que el otro jugador cometa un error, indirectamente se perjudica a sí mismo, porque si el compañero elige la opción incorrecta ninguno de los dos sumará puntos. De esta manera, en cada clip ambos jugadores intentarán describir su clip de la mejor

manera posible para poder asegurarse que su compañero tendrá toda la información necesaria para poder determinar correctamente si ambos escuchan el mismo clip o no.

Se debe tener en cuenta que, si bien para describir y así aumentar sus posibilidades de sumar puntos, ambos jugadores intentarán ingresar las etiquetas lo mejor posible, el hecho que ambos vean lo que el otro jugador escribe genera un medio de comunicación entre ambos jugadores. Esto deja abierta la posibilidad a que intenten ponerse de acuerdo en la opción que van a seleccionar, o bien indicar al otro jugador que opción van a optar. Como cada palabra o frase que ingresan es una etiqueta, las que utilicen para ponerse de acuerdo o indicar su elección también serán incluidas como etiquetas. Aunque esto perjudique el juego, en el procesamiento posterior de la información, no será muy difícil identificar la mayoría de estas etiquetas incorrectas y descartarlas, esto se debe a que el conjunto de palabras que necesitan es muy reducido y por lo tanto fácil de encontrar.

Partida bonus

Otra manera de juego es la que se conoce como la partida bonus. La partida bonus se activa cuando durante una partida común, los jugadores aciertan las tres rondas obteniendo el máximo puntaje posible para una partida. En caso que los jugadores acierten la modalidad ronda bonus también recibirán puntos.

Esta partida no genera etiquetas sobre los clips sino que sirve para establecer una relaciona entre ellos. Cuando ambos jugadores aciertan las tres rondas de una partida común, automáticamente se les notifica a ambos jugadores que a continuación podrán participar en una ronda bonus pudiendo ambos rechazar la propuesta.

Si ambos jugadores aceptan jugar la partida bonus, el sistema seleccionará tres clips de audio que se les proporcionará. Una vez que ambos jugadores hayan descargado los tres clips (se espera a que ambos posean los clips por las razonas expuestas anteriormente) comienzan a reproducirse. Al finalizar la reproducción de los tres fragmentos de audio los jugadores tendrán diez segundos para decidir cuál de los tres fragmentos es el más diferente, si ambos coinciden en la elección recibirán 50 puntos.

El motivo para incluir la partida bonus es producir información adicional que en principio, servirá para dos objetivos particulares. El primero, con el dato de la similitud entre dos fragmentos se puede generar y perfeccionar un sistema de recomendación de música. El segundo, al tener la información sobre la similitud entre dos fragmentos de música se tiene una idea aproximada del nivel de dificultad que se presentará en una ronda normal para un determinado par de clips. Específicamente, cuando dos clips son similares los jugadores deberán ingresar un mayor número de etiquetas y más específicas para poder diferenciarlos. Esta información sobre la dificultad que puede llegar a tener una partida dada según la similitud de los fragmentos de audio se puede utilizar en la selección de clips para aumentar la complejidad a medida que los jugadores aumentan su experiencia en el juego de manera que se incremente el nivel de motivación.

Para que esta información comience a ser tomada en cuenta para la selección de clips de una ronda ordinaria deberá llevarse disputada una aceptable cantidad de partidas dependiendo de la cantidad de fragmentos que posea la base de datos del sistema. Si la cantidad de partidas es reducida las posibilidades que dos clips hayan sido comparados una o más veces es muy pequeña.

Modalidad de un jugador

Existe la posibilidad que un jugador ingrese al juego en un momento determinado, en el cual no haya ningún otro jugador para formar pareja, o el número total de jugadores sea par y esto haga imposible asignarle una pareja a dicho usuario. Para evitar esta situación y que el usuario se quede sin poder jugar se diseño una modalidad de juego de un solo jugador.

La modalidad de un solo jugador permite que en cualquier momento un usuario pueda iniciar una partida, independientemente de la cantidad de usuarios conectados al juego. Esta modalidad es transparente para el usuario, dado que la pareja del usuario es reemplazada por un “bot” el cual reproduce una serie de rondas que ya han sido jugadas por alguna pareja real de usuarios. El “bot” es un algoritmo que reproduce el comportamiento que un usuario determinado tuvo en una ronda anterior, la cual fue almacenada.

Al ingresar al juego, si no hay pareja disponible para el jugador el juego selecciona una partida guardada dependiendo del nivel de experiencia del usuario, es requisito indispensable que las rondas hayan tenido un resultado positivo, esto es, que ambos usuarios hayan coincidido en la elección final de la ronda y que la elección haya sido la correcta y además en la partida debe haberse jugado bonus. Esto es necesario para poder asumir que los tags ingresados sean considerados válidos.

Una vez seleccionadas las rondas se le notifica al usuario de la asignación de una pareja, el “bot” (el usuario nunca sabrá que en realidad está jugando contra un “bot”). Durante el transcurso de cada ronda el “bot” ingresa los tags en la misma secuencia que fueron ingresados por el usuario que está reproduciendo. Una vez finalizada la reproducción del fragmento de audio, el juego evalúa los tags ingresados por el usuario real y a partir de éstos determina qué debe seleccionar, si están escuchando el mismo fragmento o no.

Como se puede observar, el juego intenta reproducir el comportamiento de un usuario real, comparando los tags ingresados por su pareja con los tags que ya conoce sobre el fragmento que él está escuchando y a partir de esto tomar una decisión.

Para determinar si un fragmento de audio es el mismo que está escuchando su pareja, el "bot" analizará el porcentaje de concordancia entre el conjunto de tags ingresados por su pareja y el conjunto de tags del fragmento de audio que él está escuchando. El porcentaje requerido por el juego para que el "bot" considere que ambos fragmentos son el mismo es del 70%, esto es, de la cantidad de tags ingresados por el usuario real, al menos el 70% de los mismos deben estar incluidos dentro del conjunto de tags relacionados con el fragmento de audio. Este porcentaje es arbitrario y se puede configurar dentro de la aplicación.

El bonus de la partida en modo un sólo jugador también estará basado en una partida almacenada previamente. El juego seleccionará una partida bonus previa y el "bot" escogerá la misma opción que escogió el usuario original de la partida.

La principal ventaja de poseer un modo de un sólo jugador es que en ningún momento un jugador se quedará sin poder jugar. Otra ventaja importante es que el resultado obtenido de una partida en esta modalidad puede ser utilizado al igual que el resultado del resto de las partidas ordinarias, dado que el usuario real de la partida escogerá si están escuchando o no el mismo fragmento de audio a partir de los tags que ingreso el "bot" el cual sólo reproduce los tags ingresados por un usuario real. Por otro lado, el "bot" tomará su decisión a partir de la comparación de los tags ingresados por un usuario real, su pareja, y los tags relacionados al fragmento de audio, los cuales fueron ingresados por usuarios reales previamente.

Una desventaja a tener en cuenta en la modalidad de un sólo jugador, es la imposibilidad de llevarla a cabo si no existen partidas previamente jugadas. Dado que es necesario contar con al menos tres rondas almacenadas para poder emular una partida completa.

Inicialmente se había pensado en reproducir una partida anterior exactamente como se había jugado originalmente, esto fue rápidamente descartado ya que se consideró

erróneo ya que el “bot” no puede escoger la misma opción que se escogió en la partida original dado que los tags que ingresa su pareja podrían no ser los mismo que los que ingresó la pareja del jugador original. Es por esto que se optó por utilizar una cota de coincidencia entre los tags ingresados por la pareja del “bot” y los tags ingresados por el usuario original, al cual el “bot” representa.

Otras opciones del juego

Todos los jugadores registrados tienen la posibilidad de ver las partidas que disputaron a lo largo de su vida dentro del juego. En esta opción los jugadores verán en pantalla el resultado de cada una de las rondas jugadas y podrán seleccionar una ronda en particular para poder tener en pantalla las etiquetas que ingresaron, tanto él mismo como las que ingresó su compañero. También sabrá que opción eligió cada uno (si escuchaban el mismo o diferente fragmento), pudiendo además reproducir ambos fragmentos. De esta manera el jugador puede revisar sus decisiones y así aprender de sus errores.

En otra pantalla el jugador podrá ver un resumen de su historial como jugador. En este resumen se mostrará el total de puntos, la cantidad de partidas que disputó, un promedio de puntos obtenidos por partida.

Conclusión

En el presente capítulo se estudiaron las diferentes opciones de juego, ventajas y desventajas de cada estrategia analizada, como así también las decisiones tomadas en consecuencia. Cabe destacar la importancia en la selección de los clips de audio como así también el armado de las parejas, ambas características fundamentales del juego. El desarrollo del modo un sólo jugador es otra característica importante del juego, junto con la implementación del bonus. En los próximos capítulos se estudiará la implementación de estas características entre otros temas.

CAPÍTULO 5

Tecnologías y herramientas utilizadas en el desarrollo de mGWAP

Introducción

A continuación se describen las tecnologías utilizadas en el desarrollo del juego mGWAP, lenguajes de programación y las herramientas empleadas. Cada tecnología utilizada fue escogida teniendo en cuenta las ventajas que cada una de éstas proporciona, facilidad de uso debido al conocimiento previo de las mismas por parte de los integrantes del proyecto y licencia de software libre de todas ellas.

Lenguaje Java

Java [29, 30, 31] es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process [40], si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora

software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es).

La tecnología Java se creó como una herramienta de programación para ser usada en un proyecto denominado the Green Project en Sun Microsystems en el año 1991. El equipo (Green Team), compuesto por trece personas y dirigido por James Gosling, trabajó durante 18 meses en Sand Hill Road en Menlo Park en su desarrollo.

El lenguaje se denominó inicialmente Oak (por un roble que había fuera de la oficina de Gosling), luego pasó a denominarse Green tras descubrir que Oak era ya una marca comercial registrada para adaptadores de tarjetas gráficas y finalmente se renombró a Java.

Los objetivos de James Gosling eran implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a C++. Entre junio y julio de 1994, tras una sesión maratoniana de tres días entre John Gaga, James Gosling, Joy Naughton, Wayne Rosing y Eric Schmidt, el equipo reorientó la plataforma hacia la Web. Sintieron que la llegada del navegador web Mosaic, propiciaría que Internet se convirtiese en un medio interactivo, como el que pensaban era la televisión por cable. Naughton creó entonces un prototipo de navegador, WebRunner, que más tarde sería conocido como HotJava.

En 1994, se les hizo una demostración de HotJava y la plataforma Java a los ejecutivos de Sun. Java 1.0a pudo descargarse por primera vez en 1994, pero hubo que esperar al 23 de mayo de 1995, durante las conferencias de SunWorld, a que vieran la luz pública Java y HotJava, el navegador Web. El acontecimiento fue anunciado por John Gage, el Director Científico de Sun Microsystems. El acto estuvo acompañado por una pequeña sorpresa adicional, el anuncio por parte de Marc Andreessen, Vicepresidente Ejecutivo de Netscape, que Java sería soportado en sus navegadores. El 9 de enero del año siguiente, 1996, Sun fundó el grupo empresarial JavaSoft para que se encargase del desarrollo tecnológico. Dos semanas más tarde la primera versión de Java fue publicada.

La promesa inicial de Gosling era Write Once, Run Anywhere (Escríbelo una vez, ejecútalo en cualquier lugar), proporcionando un lenguaje independiente de la

plataforma y un entorno de ejecución, la JVM) ligero y gratuito para las plataformas más populares de forma que los binarios (bytecode) de las aplicaciones Java pudiesen ejecutarse en cualquier plataforma.

El entorno de ejecución era relativamente seguro y los principales navegadores web pronto incorporaron la posibilidad de ejecutar applets Java incrustadas en las páginas web.

Java ha experimentado numerosos cambios desde la versión primigenia, JDK 1.0, así como un enorme incremento en el número de clases y paquetes que componen la biblioteca estándar.

Desde J2SE 1.4, la evolución del lenguaje ha sido regulada por el JCP (Java Community Process), que usa Java Specification Requests (JSRs) para proponer y especificar cambios en la plataforma Java. El lenguaje en sí mismo está especificado en la Java Language Specification (JLS), o Especificación del Lenguaje Java. Los cambios en los JLS son gestionados en JSR 901.

En el 2005 se calcula en 4,5 millones el número de desarrolladores y 2.500 millones de dispositivos habilitados con tecnología Java.

El lenguaje Java se creó con cinco objetivos principales:

1. Debería usar la metodología de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Para conseguir la ejecución de código remoto y el soporte de red, los programadores de Java a veces recurren a extensiones como CORBA (Common Object Request Broker Architecture), Internet Communications Engine u OSGi respectivamente.

Una de las características más importante del lenguaje Java es la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de

escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, “write once, run everywhere”.

Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como “bytecode” (específicamente Java bytecode), instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está “a medio camino” entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hilos o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por el compilador JIT (Just In Time).

Hay implementaciones del compilador de Java que convierten el código fuente directamente en código objeto nativo, como GCJ. Esto elimina la etapa intermedia donde se genera el bytecode, pero la salida de este tipo de compiladores sólo puede ejecutarse en un tipo de arquitectura.

Las primeras implementaciones del lenguaje usaban una máquina virtual interpretada para conseguir la portabilidad. Sin embargo, el resultado eran programas que se ejecutaban comparativamente más lentos que aquellos escritos en C o C++. Esto hizo que Java se ganase una reputación de lento en rendimiento. Las implementaciones recientes de la JVM dan lugar a programas que se ejecutan considerablemente más rápido que las versiones antiguas, empleando diversas técnicas, aunque sigue siendo mucho más lento que otros lenguajes.

La primera de estas técnicas es simplemente compilar directamente en código nativo como hacen los compiladores tradicionales, eliminando la etapa del bytecode. Esto da lugar a un gran rendimiento en la ejecución, pero tapa el camino a la portabilidad. Otra técnica, conocida como compilación JIT (Just In Time, o “compilación al vuelo”), convierte el bytecode a código nativo cuando se ejecuta la aplicación. Otras máquinas

virtuales más sofisticadas usan una “re compilación dinámica” en la que la maquina virtual es capaz de analizar el comportamiento del programa en ejecución y recompila y optimiza las partes críticas. La re compilación dinámica puede lograr mayor grado de optimización que la compilación tradicional (o estática), ya que puede basar su trabajo en el conocimiento que de primera mano tiene sobre el entorno de ejecución y el conjunto de clases cargadas en memoria. La compilación JIT y la re compilación dinámica permiten a los programas Java aprovechar la velocidad de ejecución del código nativo sin por ello perder la ventaja de la portabilidad en ambos.

La portabilidad es técnicamente difícil de lograr, y el éxito de Java en ese campo ha sido dispar. Aunque es de hecho posible escribir programas para la plataforma Java que actúen de forma correcta en múltiples plataformas de distinta arquitectura, el gran número de estas con pequeños errores o inconsistencias llevan a que a veces se parodie el eslogan de Sun, "Write once, run Anywhere" como "Write once, debug everywhere" (o “Escríbelo una vez, ejecútalo en cualquier parte” por “Escríbelo una vez, depúralo en todas partes”)

El concepto de independencia de la plataforma de Java cuenta, sin embargo, con un gran éxito en las aplicaciones en el entorno del servidor, como los Servicios Web, los Servlets, los Java Beans, así como en sistemas empotrados basados en OSGi, usando entornos Java empotrados.

Recolector de basura (Garbage Collector)

En Java el problema de las fugas de memoria se evita en gran medida por el recolector de basura (o garbage collector). El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java (Java runtime) es el responsable de gestionar el ciclo de vida de los objetos. El programa, u otros objetos pueden tener localizado un objeto mediante una referencia a éste. Cuando no quedan referencias a un objeto, el recolector de basura de Java borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas (ejemplo: un objeto creado y únicamente usado dentro de un método sólo tiene entidad dentro de éste; al salir del método el objeto es eliminado). Aun así, es posible que se produzcan fugas de memoria si el código almacena referencias a objetos que ya no son necesarios, es decir, pueden aún

ocurrir, pero en un nivel conceptual superior. En definitiva, el recolector de basura de Java permite una fácil creación y eliminación de objetos, mayor seguridad y puede que más rápida que en C++.

Sintaxis

La sintaxis de Java se deriva en gran medida de C++. Pero a diferencia de éste, que combina la sintaxis para programación genérica, estructurada y orientada a objetos, Java fue construido desde el principio para ser completamente orientado a objetos. Todo en Java es un objeto (salvo algunas excepciones), y todo en Java reside en alguna clase (recordemos que una clase es un molde a partir del cual pueden crearse varios objetos).

Java EE

Java Platform, Enterprise Edition o Java EE, es una plataforma de programación que forma parte de la Plataforma Java, utilizada para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por The Java Community Process / JCP.

Java EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets, JavaServer Pages y varias tecnologías de servicios web. Esto permite al desarrollador crear una Aplicación Enterprise portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes

desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

Android

Android [5] es un stack de software para dispositivos móviles y computadoras basado en el núcleo Linux. Inicialmente fue desarrollado por Google y continuado luego por la Open Handset Alliance (liderada Google). La presentación de la plataforma Android se realizó el 5 de noviembre de 2007 junto con la fundación de Open Handset Alliance, un consorcio de 48 compañías de hardware, software y telecomunicaciones comprometidas a la promoción de estándares abiertos para dispositivos móviles.

Esta plataforma permite el desarrollo de aplicaciones por terceros. Los desarrolladores deben escribir código en lenguaje de programación Java a través del SDK proporcionado por el mismo Google. Una alternativa es el uso de la NDK (Native Development Kit) de Google para hacer el desarrollo en C en código fuente.

La mayoría del código fuente de Android ha sido publicado bajo la licencia de software Apache, una licencia de software libre y código fuente abierto.

Características

Las principales características de Android son:

- Framework de aplicaciones: permite reutilización y reemplazo de componentes.
- Máquina virtual Dalvik: optimizada para dispositivos móviles.
- Navegador integrado: basado en el motor de código abierto WebKit.
- Gráficos optimizados, con una biblioteca de gráficos 2D; gráficos 3D basado en la especificación OpenGL ES 1.0 (aceleración por hardware opcional).
- SQLite para almacenamiento de datos estructurados.
- Soporte para medios con formatos comunes de audio, vídeo e imágenes planas (MPEG4, H.264, MP3, OGG, AAC, AMR, JPG, PNG, GIF).
- Telefonía GSM (dependiente del hardware).
- Bluetooth, EDGE, 3G, y WiFi (dependiente del hardware).
- Cámara, GPS, brújula, y acelerómetro (dependiente del hardware).

- Ambiente rico de desarrollo incluyendo un emulador de dispositivo, herramientas para depurar, perfiles de memoria y rendimiento, y un complemento para el IDE Eclipse.
- Pantalla táctil.

Arquitectura de Android

A continuación se presenta un resumen de los componentes principales del sistema operativo de Android.

Aplicaciones:

Las aplicaciones base incluyen un cliente de email, programa de SMS, calendario, mapas, navegador, contactos, y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.

Framework de aplicaciones:

Los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura del framework de aplicaciones está diseñada para simplificar el reutilización de componentes, cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

Bibliotecas:

Android incluye un set de bibliotecas C/C++ usadas por varios componentes del sistema Android. Estas características se exponen a los desarrolladores a través del framework de aplicaciones de Android; algunas son: System C library (implementación biblioteca C standard), bibliotecas de medios, bibliotecas de gráficos, 3D, SQLite, entre otras.

Runtime de Android:

Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik.

La máquina virtual Dalvik ha sido escrita de forma tal que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual Dalvik está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato .dex por la herramienta incluida "dx".

Núcleo Linux:

Android depende de Linux versión 2.6 para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, stack de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto del stack de software.

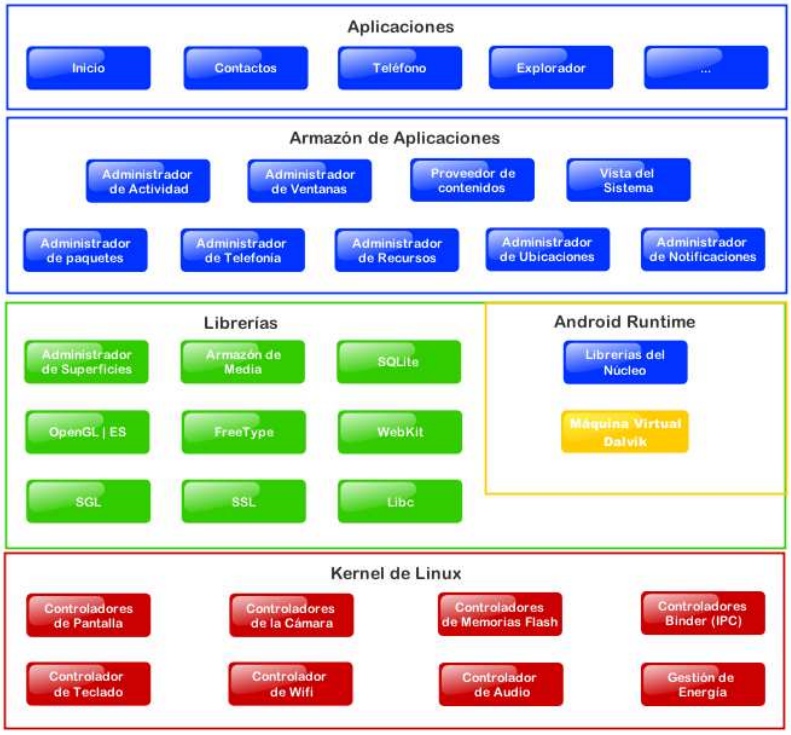


Figura 5.1 Arquitectura Android

Jerarquía de clases de kas aplicaciones Android:

La principal clase de Android es Activity (android.app.Activity). Una actividad hace múltiples de cosas, pero por ella misma no presenta nada en la pantalla. Para conseguir que tenga una representación visual en la pantalla es necesario diseñar el UI (User Interface) con *views* y *viewgroups*, que son las clases que se utilizan para crear la interfaces de usuario.

Views:

Una view es un objeto de la clase **android.view.View**. Es una estructura de datos cuyas propiedades contienen la información específica del área rectangular de la pantalla. Una view tiene: layout, drawing, focus change, scrolling, etc.

La clase **View** es la clase base de los **Widgets**. Éstos son las subclases concretas de **View** provistas por el framework que dibujan elementos en la pantalla. Los objetos **widgets** contienen sus propias medidas y se pueden usar para construir más rápidamente una interface de usuario. La lista de **widgets** ya implementados incluye Text, EditText, InputMethod, MovementMethod, Button, RadioButton, CheckBox, y ScrollView.

Viewgroups:

Un **viewgroup** es un objeto de la clase **android.view.Viewgroup** y como su propio nombre indica, un **viewgroup** es un **view** especial cuya función es contener y controlar la lista de **views** y de otros **viewgroups**. Los **viewgroups** permiten añadir estructuras a la interface de usuario y acumular elementos complejos en la pantalla.

La clase **Viewgroup** es la clase base de las clases **Layouts**. Éstas los tipos más comunes de layouts de pantalla. Los layouts proporcionan una manera de construir una estructura de posicionamiento para una lista de views.

Árbol estructurado de la interface de usuario:

En la plataforma Android el desarrollador define una **Activity** usando un árbol de nodos **View** y **Viewgroups**, como se observa en la Figura 5.2. Este árbol puede ser tan

simple o complejo como se necesite hacerlo y se puede desarrollar tanto usando los **widgets** y **layouts** que proporciona Android como creando los propios.

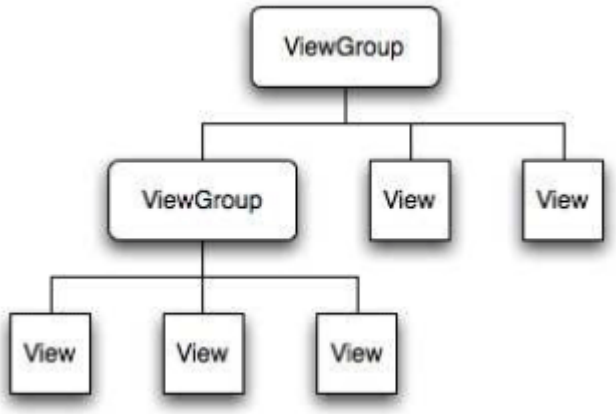


Figura 5.2 Jerarquía visual de Android

Para añadir el árbol interface del usuario a la pantalla, el **Activity** invoca al método **setContentView()** y pasa como parámetro una referencia al objeto nodo principal. Una vez que el sistema Android ha referenciado el objeto nodo principal ya puede trabajar directamente con el nodo para dibujar el árbol. Cuando el **Activity** está activo y recibe el foco el sistema notifica al **Activity** y pide al nodo principal medidas y dibuja el árbol. El nodo principal entonces pide que sus nodos hijos se dibujen, y a partir de ese momento cada nodo **viewgroup** del árbol es responsable de pintar sus hijos directos.

Como se ha dicho anteriormente, cada **viewgroup** es el responsable de tomar medidas sobre el espacio que tienen, preparando a sus hijos e invocando al método **draw()** para cada nodo hijo que se muestra a sí mismo. El nodo hijo hace una petición sobre el tamaño y la localización del padre, pero el objeto padre toma la última decisión sobre el tamaño que cada hijo puede tener.

Protocolo Bayeux

El principal objetivo del protocolo Bayeux [33] es soportar la comunicación bidireccional entre el servidor y el cliente web.

Bayeux es un protocolo de envío de mensajes asincrónicos (en principio sobre HTTP), con baja latencia entre el servidor web y el cliente. Los mensajes son enviados a través de canales nombrados y pueden ser enviados de la siguiente manera:

- Desde el servidor al cliente.
- De un cliente hacia otro cliente.
- De un cliente hacia otro cliente, pasando por el servidor.

El mecanismo de envío de mensajes asincrónicos desde el servidor hacia el cliente, es a menudo denominado "server-push".

El protocolo Bayeux busca reducir la complejidad del desarrollo de aplicaciones Comet[11], permitiendo a los desarrolladores realizar la comunicación entre el cliente y el servidor más fácilmente, resolviendo los problemas comunes de distribución y enrutamiento de los mensajes.

La especificación del protocolo Bayeux utiliza algunos términos para referirse a los roles de los participantes y objetos en la comunicación Bayeux:

- Cliente: un programa que inicia la comunicación. Un cliente Bayeux inicia el intercambio de mensajes Bayeux, el cual por lo general se ejecuta dentro de un cliente HTTP.

- Servidor: un servidor Bayeux acepta y responde al intercambio de mensajes iniciado por un cliente Bayeux.

- Mensaje: un mensaje Bayeux, es un objeto JSON, el cual es enviado entre el cliente y el servidor Bayeux.

- Evento: dato que es enviado sobre el protocolo Bayeux cuando la aplicación lo especifica.

- Canal: especifica el destino y/o la fuente de eventos. Los eventos son publicados en el canal y recibidos por aquellos clientes que se suscribieron al previamente canal.

Canales Bayeux

[11] Comet es un neologismo para describir un modelo de aplicación web en el que una petición HTTP mantenida abierta permite a un servidor web enviar datos a un navegador por tecnología push, sin que el navegador los solicite explícitamente.

Los canales son identificados por nombre con un formato tipo URI. El canal consiste de una barra inicial (“/”) seguida por una secuencia opcional de segmentos separados cada uno por una barra. Los canales que comienzan con "/meta/" están reservados para el protocolo Bayeux.

Ejemplos de canales son:

/foo
/foo/bar
/foo-bar/(foobar)

Canales Meta

Sólo el servidor Bayeux puede suscribirse a los meta canales, éstos se identifican con "/meta/". Los mensajes publicados en un meta canal no pueden ser distribuidos a los clientes remotos de Bayeux. Un servidor manejador de un meta canal puede publicar mensajes de respuesta que son entregados sólo al cliente que envío el mensaje original. Si un mensaje publicado en un meta canal contiene un campo ID, entonces el mensaje de respuesta enviado al cliente debe contener el mismo ID.

Canales Service

Los canales que comienzan con "/service/" son canales especiales diseñados para asistir en el envío de mensajes del estilo request/response.

Los mensajes publicados en un canal service no son distribuidos a ningún cliente remoto Bayeux. El manejador del canal puede publicar una respuesta que será recibida sólo por el cliente que realizó el requerimiento. El servidor no almacena ninguna suscripción a un canal service.

Suscripción de un cliente a un canal

Un cliente Bayeux puede enviar un mensaje de petición de suscripción para registrarse en un determinado canal y de esta manera solicitar que todos los mensajes publicados en dicho canal le sean enviados.

El mensaje de suscripción al canal debe ser enviado al canal "/meta/suscribe" indicando el nombre del canal al cual se suscribirá y el ID de cliente. De manera similar puede cancelar su suscripción enviando un mensaje al canal “/meta/unsuscribe”.

Implementación Bayeux utilizada en mGWAP

En el proyecto mGWAP se utilizó la implementación del protocolo Bayeux denominada CometD [34] para la implementación del lado cliente, la cual fue desarrollada por la Dojo Foundation. Del lado del servidor se usó la implementación de Jetty Continuations [35], ambas librerías están implementadas como en Java. En la Figura 5.3 se muestra como es la secuencia de comunicación entre el cliente y el servidor utilizando Comet.

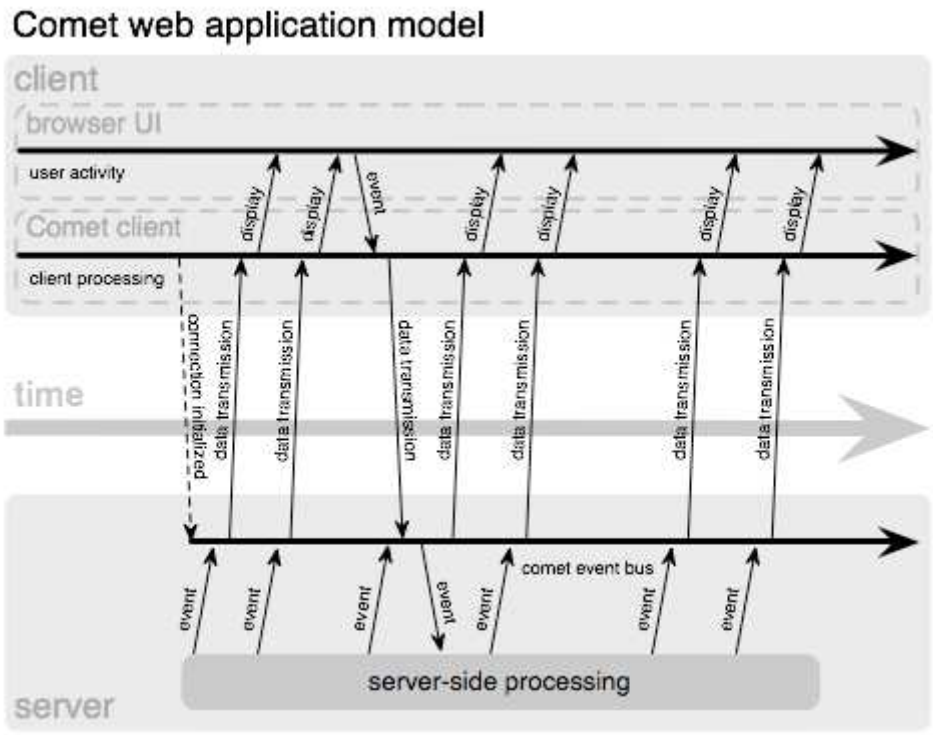


Figura 5.3 Ejemplo de secuencia de comunicación de Comet

PostgreSQL

PostgreSQL [8] es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD.

Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Algunas de sus principales características son, entre otras:

Alta concurrencia:

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS [41].

Otras características:

- Claves ajenas también denominadas Llaves ajenas o Claves Foráneas (foreign keys).
- Disparadores (triggers).
- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

- Funciones.

Hibernate

Hibernate [36] es una herramienta de Mapeo objeto-relacional para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL.

Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la Programación Orientada a Objetos. Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por SQL.

Hibernate está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.

Hibernate para Java puede ser utilizado en aplicaciones Java independientes o en aplicaciones Java EE, mediante el componente Hibernate Annotations que implementa el estándar JPA [37], que es parte de esta plataforma.

Java Persistence API (JPA) proporciona un estándar para gestionar datos relacionales en aplicaciones Java SE o Java EE, de forma que además se simplifique el desarrollo de la persistencia de datos.

El mapeo objeto-relacional (es decir, la relación entre entidades Java y tablas de la base de datos, queries con nombre, etc.) se realiza mediante anotaciones en las propias clases de entidad. No se requieren ficheros descriptores XML. También pueden definirse transacciones como anotaciones JPA.

JSON

JSON [39] (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript.

JSON es un formato de texto que es completamente independiente del lenguaje de programación pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

Una colección de pares de nombre/valor. En varios lenguajes de programación esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.

Una lista ordenada de valores. En la mayoría de los lenguajes de programación, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

Tomcat

Tomcat [39] (también llamado Jakarta Tomcat o Apache Tomcat) es un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

Tomcat es un servidor web con soporte de servlets y JSPs. Pero no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila páginas JSP convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat es mantenido y desarrollado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software Licence.

En mGWAP se usó Tomcat como contenedor de los servlets encargados de manejar la lógica y el almacenamiento de datos.

Eclipse

Eclipse [7] es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido" y aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones.

Eclipse dispone de un editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes para creación de proyectos, clases, test, etc., y refactorización.

Asimismo, a través de "plugins" libremente disponibles es posible añadir control de versiones con Subversión e integración con Hibernate.

Para el desarrollo de la aplicación se instaló un “plugin” para Eclipse llamado “Android Development Tools” (ADT)[12] que facilita el desarrollo de aplicaciones Android. El mismo es desarrollado por Google, contiene el SDK necesario para el desarrollo y un emulador para probar las aplicaciones.

[12] http://developer.android.com/sdk/eclipse-adt.html

CAPÍTULO 6

Implementación de mGWAP

Introducción

En el presente capítulo se expondrán las características de diseño e implementación de la aplicación cliente y servidor del juego mGWAP. En este mismo capítulo también se describirán las tecnologías utilizadas, así como también las diferentes librerías que fueron necesarias incluir en el desarrollo.

Por otra parte, se abordará el diseño visual del juego, junto con un análisis de las diferentes consideraciones que se tuvieron en cuenta al momento de la implementación.

Tecnología utilizada en la implementación del cliente mGWAP

Como se mencionó en los capítulos anteriores, el cliente mGWAP fue desarrollado para la plataforma de teléfonos móviles Android. Dicha plataforma utiliza como lenguaje de desarrollo principalmente Java y también el lenguaje XML para la definición de las interfaces visuales.

A continuación explicaremos los principales conceptos de la plataforma de desarrollo de Android y cómo esos conceptos fueron aplicados en la implementación de **mGWAP** cliente.

Activities Android en mGWAP

En el modelo MVC, la clase **Activity** representa el rol de controlador de la vista. Una aplicación Android puede contener una o varias **activities**, las cuales implementan la lógica de la aplicación. En la aplicación cliente **mGWAP** se definieron las siguientes **activities**:

LoginActivity: es el **Activity** responsable de la autentificación del usuario, tiene la opción “Entrar” para ejecutar el inicio de sesión del usuario y la opción “Registrarme” para que un nuevo usuario pueda registrarse en el juego.

SignUpActivity: es el **Activity** responsable de la registración de un nuevo usuario. A partir de un nombre de usuario, una contraseña y una dirección de mail se da de alta un usuario nuevo.

MainActivity: esta **Activity** es la encargada de mostrar y administrar la pantalla principal del usuario, la cual se muestra apenas se ingresa al juego. Esta pantalla tiene las opciones "Jugar", que le permite al usuario iniciar una nueva partida, "Partidas Jugadas" para ver las partidas jugadas anteriormente y la opción "Salir" para abandonar el juego.

ListPastGames y ViewPastGame: son **activities** responsables de mostrar al usuario las partidas jugadas anteriormente. La primera lista las partidas jugadas, mientras que la segunda muestra el detalle de la partida.

GameRoomActivity: es una de las **activities** centrales de la aplicación. Es la responsable de administrar una partida determinada, contiene la lógica para el ingreso de etiquetas en cada ronda de la partida, la reproducción de los clips y las opciones de selección de cada ronda. También implementa el control del tiempo de reproducción y elección.

BonusActivity: es la **Activity** encargada de la ronda bonus que existe en determinadas partidas. La partida de bonus reproduce tres clips de audio. Esta **Activity** se encarga de la lógica necesaria para que el usuario pueda reproducir los tres clips en el orden que desee y finalmente pueda decidir cuál es el más diferente eligiendo uno de los tres botones existentes para tal fin.

MGwapActivity: es una clase abstracta, superclase de todas las **activities**. Contiene métodos y propiedades comunes a todas las **activities** del juego, como por ejemplo métodos para mostrar y ocultar un diálogo informativo. Durante el desarrollo de **mGwap** se observó que existía un comportamiento común entre todas las **activities** del juego, fue por esto que se implementó la clase MGwapActivity.

La mayor parte de la lógica del cliente mGWAP se encuentra implementada dentro de las **activities** mencionadas. Pero junto a éstas existe también una clase llamada **GameHttpClient** responsable de toda la comunicación entre el cliente y el servidor. Dicha clase será explicada más adelante en este mismo capítulo.

Views Android en mGWAP

Las **activities** por sí mismas no se muestran en pantalla, para esto existen las **Views**. Éstas se escriben en lenguaje XML y definen una estructura de capas y componentes.

Cuando se instancia la **Activity** se le indica cual será la **View** que utilizará para mostrar en pantalla. Las **Views** son equivalentes a las páginas HTML de una aplicación Web.

En la implementación del cliente **mGWAP** se definieron las siguientes **Views**:

Login.xml: es la primera **View** con la que se encuentra un usuario cuando ingresa al juego. En la misma se puede ingresar el nombre del usuario y la contraseña para acceder al juego o en caso de no estar registrado aún, se puede seleccionar la opción "Registrarme" para ir a la pantalla de registración. Esta **View** es utilizada por **LoginActivity**.

Signup.xml: es la **View** responsable de mostrar el formulario de registración de usuarios. Los campos del formulario son: nombre de usuario, contraseña y dirección de mail. Esta **View** es utilizada por **SignUpActivity**.

HomePastGame.xml, ListPastGame.xml y PastGame.xml: son las **Views** que muestran la lista de partidas jugadas por un usuario y el detalle de las mismas.

Home.xml: es la **View** que contiene las opciones principales, las cuales son mostradas al usuario cuando ingresa al juego, luego de iniciar sesión. Las opciones son "Jugar", para iniciar una nueva partida, "Partidas anteriores" para ver sus partidas anteriores y "Salir" para abandonar el juego. Esta **View** es utilizada por **MainActivity**.

Game.xml: esta **View** se corresponde con la pantalla principal del juego, donde se desarrollan las partidas. Muestra el puntaje acumulado hasta el momento en la partida, el tiempo, una barra de progreso que refleja la reproducción del clip, una zona

principal dividida en dos donde se muestran las etiquetas ingresadas por ambos usuarios, un campo de entrada para que el jugador ingrese los tags y tres botones, “Enviar” para enviar un tag o etiqueta, y las dos opciones “Iguales” o “Diferentes” para indicar qué elige el usuario. Esta **View** es utilizada por **GameRoomActivity**.

Bonus.xml: esta **View** se corresponde con la pantalla del bonus del juego, utilizada por **BonusActivity**. Muestra en forma similar a Game.xml una barra con el puntaje, el tiempo transcurrido y tres barras de progreso que se corresponden con cada clip del bonus, por último tiene tres botones que representan a cada clip en la elección de cuál es el más diferente de los tres.

El conjunto de **Views** detallados componen la interface gráfica de **mGWAP**.

A continuación se explicará cómo las **activities** se comunican entre sí asincrónicamente, luego se detallaran las características de comunicación con el servidor como así también la manera en que el cliente **mGWAP** maneja los tiempos. Por último se hará una breve descripción del componente reproductor utilizado.

Handlers de Android

Android utiliza *handlers* para el pasaje de mensajes entre **activities**, los cuales son objetos de la clase *Handler*. Una **Activity** puede contener más de una instancia de objetos *handlers,* los cuales pueden ser invocados desde otros objetos para enviarle mensajes al **Activity** de manera asincrónica. La utilización de los *handlers* es sencilla, se implementan como propiedades de las **activities** y son invocados por medio de los métodos definidos en la clase *Handler*.

Los *handlers* son utilizados en el cliente **mGWAP** para comunicar las **activities** del cliente **mGWAP** con el proceso responsable de recibir y enviar la información al servidor. Cuando se recibe una respuesta del servidor, el proceso responsable de la comunicación, instancia de GameHttpClient, invoca al *Handler* correspondiente y comunica esta situación.

Protocolo Bayeux y su utilización en mGWAP

Como se mencionó en el capítulo 5, **mGWAP** utiliza el protocolo Bayeux para la comunicación entre el cliente y el servidor del juego. Existen diferentes implementaciones del protocolo Bayeux, en **mGWAP** se utilizó la implementación CometD, de la Dojo Fundation[42].

El cliente **mGWAP** define una clase llamada **GameHttpClient** que es la responsable de la comunicación entre el cliente y el servidor. El objeto **GameHttpClient** utilizado por el cliente administra las conexiones con el servidor por medio del protocolo Bayeux, se encarga de las suscripciones y descripciones a los canales y del envió y recepción de los mensajes. Esta clase también implementa la lógica necesaria para invocar los servlets que se utilizan para la autenticación del usuario y las peticiones de archivos de audio, como se verá más adelante.

Combinando la utilización de *handlers* y el protocolo Bayeux obtenemos la arquitectura necesaria para realizar la comunicación asincrónica entre el servidor y los clientes. Es importante notar que al utilizar el protocolo Bayeux es posible la comunicación de los clientes con el servidor y de los clientes con otros clientes. Lo anterior se puede entender mejor con un ejemplo:

Cuando se está desarrollando una ronda cualquiera y un jugador ingresa una etiqueta determinada, dicha etiqueta debe ser enviada al servidor para su registración, pero también debe ser enviada al otro jugador para que éste la vea. Con el protocolo Bayeux esto es posible ya que existe la posibilidad que un cliente Bayeux se suscriba a un canal determinado y reciba por dicho canal mensajes desde el servidor y desde otros clientes.

Un cliente Bayeux puede suscribirse a canales ordinarios y a servicios, los servicios son una vía de comunicación bidireccional entre el servidor Bayeux y el cliente. En cambio, un canal ordinario tiene N suscriptores, donde todos reciben una copia de lo que se publica en dicho canal.

Pantallas de mGWAP

A continuación se muestran las pantallas del cliente **mGWAP**, junto con una breve descripción de sus elementos.

Pantalla de inicio de sesión

Figura 6.1 Pantalla de inicio de sesión de mGWAP

En la pantalla de la Figura 6.1, el usuario puede ingresar al juego o solicitar la registración.

Pantalla de registración

Figura 6.2 Pantalla de registración a mGWAP

La Figura 6.2 es la pantalla de registración; una vez registrado, el usuario puede comenzar a jugar.

Pantalla principal

Figura 6.3 Pantalla principal de mGWAP

En la pantalla principal de mGWAP, Figura 6.3, se muestran las opciones de mGWAP. Jugar, permite comenzar una partida, Juegos Anteriores, permite ver las partidas que jugó, y Salir, que sale del juego.

Pantalla de Juego

Figura 6.4 Pantalla de juego

La Figura 6.4 muestra la pantalla de juego es donde tienen lugar las partidas.

Pantalla de Bonus

Figura 6.5 Pantalla de Bonus

En la pantalla de Bonus, Figura 6.5, es donde se desarrolla el juego Bonus.

Pantalla de Partidas Anteriores.

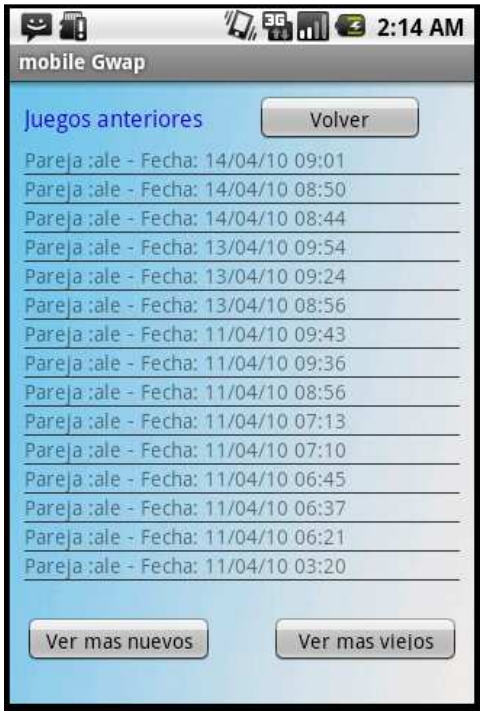


Figura 6.6 Pantalla de Partidas Anteriores

Manejo de los timeout en el cliente mGWAP

Como último tema dentro del lo que podríamos denominar la estructura del cliente **mGWAP**, queda mencionar la manera en que se implementó el manejo de los timeout dentro del cliente mGWAP.

Para poder controlar si un usuario permaneció demasiado tiempo inactivo esperando por una respuesta, si ha superado el tiempo estipulado para elegir una opción en una ronda determinada o si ha superado el tiempo para elegir jugar el bonus, se utilizaron hilos Java puros. Éstos se ejecutan en paralelo y controlan que estas situaciones no ocurran, en caso de ser así notifican a la aplicación.

Las clases **GameRoomActivity** y **BonusActivity** implementan métodos que son los responsables de la instanciación de los hilos que controlarán los tiempos dentro del juego.

GameRoomActivity contiene un hilo que controla la reproducción de un determinado clip dentro de una ronda, por cada un segundo que transcurre el hilo notifica a la interface gráfica para que dicho suceso se refleje en la barra de progreso del clip.

Por otra parte, **MGwapActivity** contiene otro hilo que controla constantemente que el usuario no pase el límite de tiempo permitido de espera, este límite de tiempo es impuesto por cada Activity concreta, por ejemplo, el tiempo estipulado en **MainActivity** no es el mismo que en **GameRoomActivity**. Dicho hilo se ejecuta cada vez que se realiza una petición al servidor, si la respuesta llega antes que se cumpla el tiempo estipulado, el hilo es interrumpido. En cambio, si la respuesta no llega a tiempo, el hilo notificará a la aplicación que transcurrió el tiempo estipulado de espera y se procederá a la interrupción del juego y posterior notificación al usuario.

Otro hilo dentro de **GameRoomActivity** es el responsable de controlar que una vez que se completo la reproducción del clip de audio el usuario escoja una opción dentro de los 5 segundos posteriores, en caso de que esto no suceda, el juego envía un aviso al servidor que el usuario no escogió opción.

Por último existe el hilo del bonus, implementado por **BonusActivity**, este hilo controla que el usuario elija una opción dentro del tiempo estipulado, dicho tiempo va desde que comienza a reproducirse el primer tema del bonus hasta una cantidad de segundos determinada después que finalizó el último clip.

El reproductor de audio de Android

MediaPlayer es la clase que implementa la plataforma Android para la reproducción tanto de audio como de video. Esta clase fue la que se utilizó en la implementación del cliente **mGWAP** para manejar la reproducción de los clips de audio. Como **MediaPlayer** no tiene la capacidad de dibujarse en pantalla, se utilizó para esto el componente **ProgressBar**; dicho componente es una barra de progreso que el cliente **mGWAP** va incrementando a medida que el audio transcurre.

MediaPlayer provee métodos para inicializar el reproductor a partir de un archivo mp3, comenzar la reproducción del audio, pausar y parar.

Implementación del servidor mGWAP

A continuación se expondrán las características de diseño e implementación de la aplicación servidor del juego **mGWAP**. En el mismo capítulo también se describirán las tecnologías utilizadas, así como también las diferentes librerías que fueron necesarias incluir en el desarrollo del mismo.

A continuación se estudiará el modelo de clases JAVA server-side junto con el mecanismo de persistencia. Siguiendo luego con el estudio de la lógica implementada dentro de los servicios Bayeux y por último se describirá la interface de comunicación con los clientes.

Finalizando el capítulo se explicarán los detalles de implementación de los algoritmos más importantes de la aplicación, como ser la selección de parejas, la selección de clips y el manejo del modo un sólo jugador.

Modelo de clases del servidor mGWAP

La Figura 6.7 presenta una descripción de las clases implementadas por el servidor **mGWAP** las cuales representan el modelo de la aplicación:

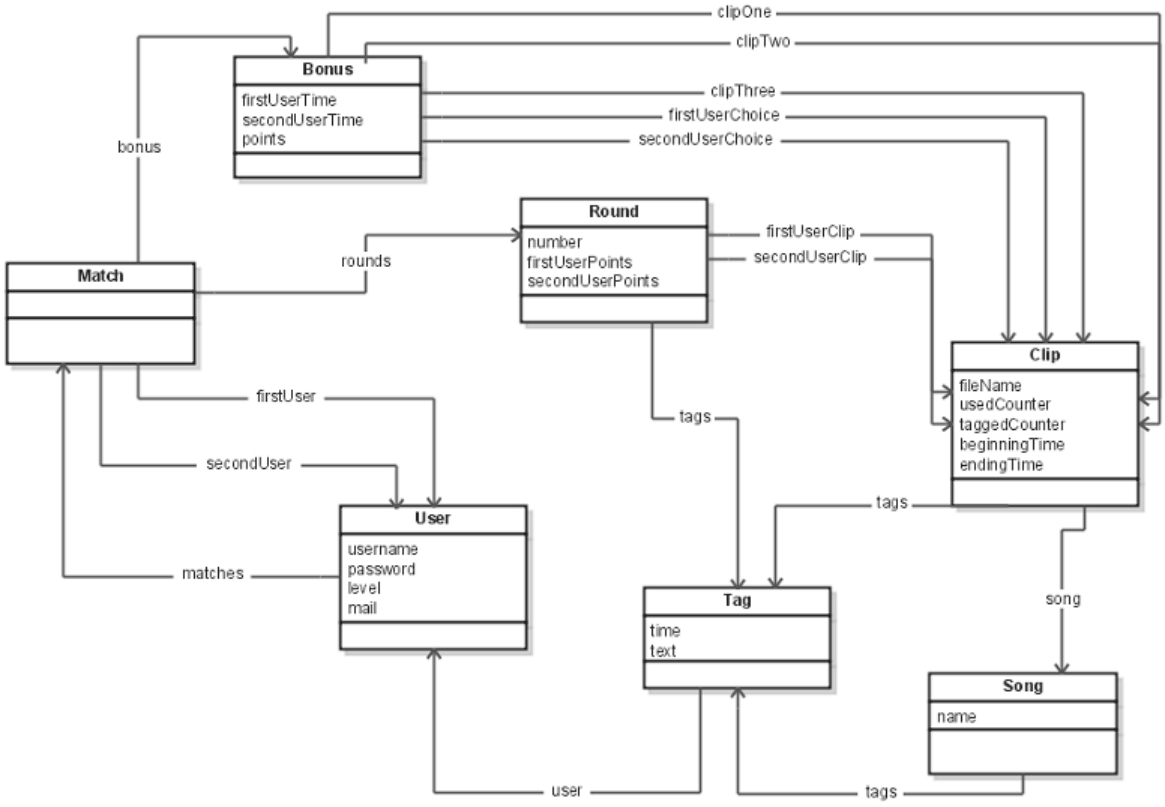


Figura 6.7 Modelo de clases mGWAP

- La clase **Tag** representa una etiqueta ingresada por un usuario determinado, y describe un clip determinado. Las instancias de **Tag** contienen la etiqueta propiamente dicha y el momento, en segundos, en el cual fue ingresada.

- La clase **Song** representa un archivo de audio. La misma contiene el nombre del audio.

- La clase **Clip** representa cada uno de los fragmentos de audio que componen la clase **Song**. Contiene el nombre del archivo en disco, la cantidad de veces que se utilizó en una ronda, la cantidad de veces que fue etiquetado y el tiempo de inicio y fin dentro de la canción.

- La clase **Bonus** representa el bonus de una partida determinada, contiene los tres Clip involucrados en el bonus, los Clips elegidos por cada jugador y el tiempo en que se hizo la elección.

- La clase **Round** representa una ronda dentro de la partida, contiene los puntajes obtenidos por los usuarios, los dos Clips utilizados, el número de round y los tags ingresados en dicha ronda.

- La clase **Match** representa una partida del juego, contiene los usuarios involucrados, los rounds jugados y el Bonus.

- La clase **User** representa un usuario del juego, contiene información del perfil, nombre de usuario, clave, matchs jugados, mail y el nivel adquirido por el usuario.

El modelo de la aplicación es persistido utilizando la librería de mapeo Objeto-Relacional, Hibernate. Para encapsular la lógica de persistencia se creó una clase llamada **DB**, la cual contiene todo el comportamiento necesario para persistir, actualizar y realizar consultas en la base de datos sobre el modelo.

Implementación de la lógica del servidor mGWAP

La implementación de la lógica en el servidor **mGWAP** está concentrada principalmente en los servicios Bayeux, los cuales son clases Java que heredan de la clase **BayeuxService**. Esta clase contiene métodos que permiten enviar y recibir mensajes por medio del protocolo Bayeux, además de poder crear, suscribirse y anular la suscripción a los canales de comunicación Bayeux. Las subclases de **BayeuxService** son provistas por la librería CometD, que implementa el protocolo Bayeux. A su vez, ciertas partes importantes de la lógica del servidor están implementadas sobre servlets Java. Y por último, los algoritmos más relevantes de la aplicación se encuentran implementados en clases Java separadas.

Los servicios Bayeux implementados.

GameService: hereda de la clase **BayeuxService**. Este servicio contiene la lógica para permitir a un usuario comenzar un juego, solicitar pareja y desarrollar su juego dentro de **mGWAP**. La clase **GameService** es la responsable de crear la partida e ir administrando la misma a medida que va recibiendo mensajes de los jugadores. Algunos de los métodos más importantes de la clase **GameService** son, solicitar el inicio de una partida por parte de un usuario, crear una partida, iniciar la misma, recibir los tags enviados en cada ronda, recibir las opciones elegidas, controlar la desconexión de alguno de los dos jugadores y guardar las partidas jugadas, entre otros.

BonusService: como **GameService**, hereda de la clase **BayeuxService**. Este servicio contiene la lógica necesaria para que los jugadores puedan jugar la ronda bonus. Permite aceptar y jugar el bonus como así también manejar el rechazo al bonus de uno o ambos jugadores. Los principales métodos de esta clase son, recibir la aceptación de jugar bonus por parte de los usuarios, recibir la opción elegida y definir el resultado del bonus.

Servlets Java implementados.

Por otra parte se encuentran los servlets Java, los mismos son accedidos por los usuarios por fuera de lo que se considera el protocolo Bayeux. En lugar de enviar mensajes por medio de los canales Bayeux, el acceso es por medio de requerimientos HTTPs simples. A continuación se detallan los servlets implementados.

El servlet **GetClipById** es responsable de proveer a los clientes de **mGWAP** los archivos de audio. Implementa la lógica necesaria para, a partir de un id de Clip buscar y retornar el archivo de audio perteneciente a dicho Clip.

El servlet **GetMatchById** contiene la lógica para retornar un **Match** a partir de un id determinado. El **Match** es devuelto al cliente por medio de un objeto serializable, llamado **MatchDTO**, el cual contiene información de la partida minimizada.

El servlet **LoginUserAction** contiene la lógica necesaria para que un usuario pueda autenticarse en el juego por medio de un nombre de usuario y clave.

Por último, el servlet **SignUpAction** es el responsable de la registración de un nuevo usuario en el juego. Éste recibe el nombre de usuario, clave y dirección de mail, realiza las validaciones necesarias y crea el nuevo usuario en la base de datos del juego.

El servlet **GetRoundById** contiene la lógica necesaria para retornar un **Round** a partir de un id determinado. El **Round** es devuelto al cliente por medio de un objeto serializable, llamado **RoundDTO**, el cual contiene información de la ronda.

El servlet **GetMatchesByUsername** contiene la lógica para retornar una lista de todas las partidas donde participó el usuario con id igual al enviado por parámetro.

Descripción de los principales algoritmos del servidor mGWAP.

Determinada lógica, como es la selección de usuarios y armado de las parejas, la selección de los clips a utilizar en una determinada partida o el armado de una partida en modo un sólo jugador, son los algoritmos que más se destacan dentro de la implementación del juego.

Algoritmos de selección de parejas

A continuación detallaremos algunos aspectos de la implementación del mecanismo de selección de parejas. Para administrar la selección de parejas se creó una clase, llamada **PartnerSelector**, que realiza sólo esta función. **PartnerSelector** recibe los usuarios a medida que van solicitando partidas y notifica al juego cómo son las parejas que va armando para que se creen las partidas correspondientes. El **PartnerSelector** se implementó utilizando el **patrón Singleton** debido a que se necesitaba que sea una sola instancia la encargada de manejar la información.

El **PartnerSelector** utiliza una cola para guardar los usuarios que fueron solicitando una partida; esta estructura de datos garantiza que el primero en entrar a la cola es el primero en salir para guardar los usuarios que fueron solicitando una partida, y de esta manera los objetos quedan ordenados de acuerdo al momento en que solicitaron la partida, quedando primero el que hace más tiempo que está esperando para poder

jugar y último el que lleva esperando una menor cantidad de tiempo. Cuando el **PartnerSelector** recibe un nuevo usuario que solicita una partida, primero valida si hay usuarios esperando, de no ser así lo agrega a la cola de usuarios a la espera de otro o que se finalice el tiempo de espera y se le cree una partida de un sólo jugador. Si se encuentran usuarios esperando, el **PartnerSelector** recorre la cola de usuarios para ver si el nuevo usuario es una buena opción para alguno de los que se encuentran esperando.

Para determinarlo se realizan las siguientes validaciones:

- Primero verifica si el usuario que se encuentra en espera es novato, de ser así corrobora si el nuevo usuario es experto, si ambas cosas se cumplen entonces determina que es una buena opción.

- Segundo verifica si se cumple la inversa, si el nuevo usuario es novato y el que se encuentra en espera es experto, entonces determina que es una buena opción.

- Por último, si el nuevo usuario se encuentra dentro del límite de puntos del usuario que está esperando, entonces determina que es una buena opción.

Si se cumple alguna de esas tres condiciones, el **PartnerSelector** quita al usuario de la cola de espera y notifica al juego para que se cree una partida con ambos usuarios. Si el usuario en espera no cumple con las condiciones, se le incrementa el límite de puntos, de esta manera se aumentan las posibilidades que se encuentre un usuario dentro del rango.

Un usuario tiene asidnado un tiempo de espera máximo, si este tiempo se cumple se le debe asignar una partido al usuario con otro usuario aunque no sea la mejor opción, y de no haber usuarios, se crea una partida de un sólo jugador.

Para controlar esta situación, el **PartnerSelector** crea un hilo que monitorea, una vez por segundo, si el usuario en espera superó el tiempo de espera máximo. Al estar los usuarios almacenados en una cola FIFO (First In First Out), es suficiente con verificar el tiempo de espera del primer usuario de la cola. El **PartnerSelector** guarda junto con la

información de cada usuario el momento en que solicitaron una partida. Si se detecta que a un jugador se le venció el tiempo de espera, el **PartnerSelector** le crea una partida con el segundo usuario de la cola de espera, es decir, el segundo usuario que lleva más tiempo esperando. Si el usuario al que se le terminó el tiempo de espera es el único que se encuentra en espera, el **PartnerSelector** indica que se le asigne una partida utilizando el modo de un sólo jugador. De esta manera se asegura que ningún usuario se quede esperando indeterminadamente.

En la próxima sección detallaremos algunos aspectos de la implementación del mecanismo de selección de parejas.

Algoritmos de selección de clips para cada ronda

Para la selección de clips se implemento la clase **ClipSelector**, la misma contiene la lógica necesaria para determinar qué clips serán asignados a la pareja en cada una de las rondas de una partida, como así también qué clips serán asignados al bonus en caso de que éste se presente al final de la partida. El primer método de la clase es el encargado de obtener un clip arbitrario, basándose en la cantidad de veces que el mismo fue utilizado y teniendo una colección de clips "taboos" los cuales no pueden ser elegidos ya que están asignados a una ronda previa de la partida. Este método es utilizado para obtener el primer clip de una ronda determinada. Otro de los métodos implementados por la clase **ClipSelector** es el responsable de seleccionar un clip a partir de los juegos previos de los jugadores y del primer clip seleccionado para la ronda. Este método asume que el clip buscado debe ser diferente al primero, el comportamiento del método es el siguiente:

Determina si la última ronda jugada por la pareja fue de dificultad baja o no. En caso de haber sido de dificultad baja, buscará un clip similar al primero con el objetivo de provocar una ronda de dificultad alta, y si la última ronda fue de alta dificultad buscará un clip que tenga poca similitud. En caso de que sea la primera ronda de la pareja buscará en el historial de partidas de los jugadores. Una vez determinada la dificultad que debe tener la ronda, ejecuta una consulta sobre la base de datos buscando un clip que cumpla con las condiciones definidas. En caso de no encontrar un clip que cumpla

con las condiciones, tomará un clip arbitrario utilizando las condiciones aplicadas a la selección del primer clip.

Para determinar la dificultad de una ronda se utiliza la similitud de los clips involucrados, más específicamente se determina el porcentaje de coincidencia de los *tags* de ambos clips. La clase **ClipSelector** define una constante que contiene un porcentaje limite; si la similitud supera dicho porcentaje se considera que los clip son similares, caso contrario se consideran diferentes. Además de lo mencionado, para reforzar la determinación de si dos clips son similares o no, se utiliza la información recolectada por las rondas de bonus. La clase **ClipSelector** define una segunda constante que se usa para determinar si dos clips son diferentes. Esta constante determina el porcentaje mínimo de rondas de bonus necesarios en los que uno de los clips haya sido escogido como diferente por ambos jugadores, sobre el total de rondas de bonus en las que participaron ambos clips. Por ejemplo, si ambos clips participaron juntos en diez bonus y en tres de ellos los jugadores escogieron uno de los dos clips como diferentes, el porcentaje de diferencia en el bonus será del 30%, dicho porcentaje será cotejado contra la constante definida en **ClipSelector**. Si la constante es mayor, los clips seguirán considerándose similares, caso contrario serán considerados diferentes.

Por último debemos mencionar que para la selección de los tres clips en la ronda de bonus, se aplica la regla “primero, el menos utilizado” y de esta manera beneficiar a los clips menos utilizados en **mGWAP**.

Implementación del modo un solo jugador

Para implementar la lógica del modo un sólo jugador, se definió una clase denominada **SinglePlayer**. Esta clase contiene los métodos necesarios para llevar adelante una partida con un sólo jugador a partir de una partida previa almacenada en la base de datos del juego.

A continuación se describen los pasos que se llevan a cabo para desarrollar una partida de un sólo jugador.

Una vez que el **PartnerSelector** define que se debe ejecutar el modo un sólo jugador, éste inicia el juego enviando sólo un jugador, es entonces donde interviene la clase **SinglePlayer** y crea la partida a partir de una existente. La clase **SinglePlayer** busca en la base de datos una partida existente que cumpla con las siguientes dos condiciones: que haya culminado exitosamente, esto es, que los participantes hayan coincidido en la repuesta correcta en todas las rondas y que la partida haya tenido un bonus. Seleccionada la partida, se comienza a jugar, la clase **SinglePlayer** es responsable de emular el comportamiento del jugador número uno de la partida original, debe enviar los *tags* en el mismo instante que lo hizo el jugador en la partida original, como así también la elección final de cada ronda y la elección del bonus. Una vez culminada cada ronda, la clase **SinglePlayer** debe determinar su elección basándose en los *tags* ingresados por el jugador real y los *tags* existentes en la base de datos relacionados con la partida original.

Repertorio de música

Para culminar el capítulo de implementación se describirá el repertorio de música utilizado en **mGWAP**. El mismo es una porción del utilizado en **TagATune**. Dicho repertorio consta de aproximadamente 27 mil fragmentos de canciones, distribuidos bajo una licencia que permite su uso y copia mientras no sea para uso comercial. Los archivos de los segmentos de canciones siguen la siguiente nomenclatura, por ejemplo:

barbara_leoni-human_needs-03-another_time-146-175.mp3

Siendo:

barbara_leoni: el intérprete de la canción.

human_needs: el nombre del álbum.

03: el número de la pista en el álbum.

another_time: el nombre de la canción.

146: el inicio del clip dentro de la canción, en segundos.

175: el final del clip dentro de la canción, en segundos.

.mp3: indica que están grabados en formato mp3.

El formato en que están almacenados los fragmentos es mp3, con una velocidad de transmisión de 32kbps, usando un sólo canal (mono), y una velocidad de muestra de sonido de 16KHz. Dada la baja calidad con la que están guardados, se consigue que requieran poca cantidad de almacenamiento, sólo 115kb, lo que facilita su transmisión a los usuarios. Los fragmentos no cubren el total de la canción, dado que quedan segundos que no llegan a completar un fragmento y por lo tanto fueron descartados al momento de armar el repertorio.

Para las pruebas realizadas en este trabajo se decidió reducir la cantidad de fragmentos a mil (1.000) aproximadamente, tomando los primeros, según el orden del repertorio original. El uso de todos los clips para las pruebas es excesivo y no representa ninguna diferencia en la calidad de los datos.

Conclusión

A lo largo de este capítulo se describieron los detalles de la implementación del cliente y el servidor **mGWAP**. Se utilizaron las herramientas proporcionadas por la plataforma Android. Dichas herramientas permitieron de manera eficiente culminar el desarrollo de manera exitosa, se puede decir que la plataforma provee suficiente documentación para el aprendizaje de la misma desde cero. Por otra parte, las herramientas proporcionadas fueron suficientes para cubrir todos los requerimientos del proyecto.

CAPÍTULO 7

Estadísticas de juego y posibles algoritmos de procesamiento de los datos generados por mGWAP

Introducción

En el presente capítulo se detallarán las pruebas realizadas sobre **mGWAP** que tuvieron como objetivo recabar información estadística. También se estudiarán algunos de los posibles algoritmos de procesamiento de la información generada por **mGWAP**, como así también sus posibles aplicaciones. Se debe recordar que el objetivo principal de los juegos **Gwap** es el de poder tomar la información generada por los usuarios del juego y utilizarla para entrenar a otros sistemas de cómputo. También se explicarán diferentes estrategias de *sanitización* de datos, para eliminar etiquetas erróneas, repetidas o que son consideradas inservibles.

Estadísticas de juego

Para llevar a cabo una serie de partidas de prueba y poder recabar información estadística del mismo se consideró un subconjunto de los fragmentos de audio existentes en la base de datos de **mGWAP**. Este subconjunto consta de 50 fragmentos musicales de diferentes canciones, el motivo de utilizar sólo un conjunto reducido para las pruebas es básicamente obtener como resultado clips altamente etiquetados y utilizados en las partidas. Si en cambio se hubieran utilizado todos los fragmentos del repositorio, hubiese sido necesario ejecutar un gran número de partidas para poder etiquetar dicha cantidad de clips y poder obtener una cantidad suficiente de información para realizar la evaluación de resultados.

Las partidas fueron jugadas siempre por el mismo par de usuarios, quienes no tenían ningún tipo de acuerdo previo y se prestaron como voluntarios para participar. Se debe destacar que al comenzar con las pruebas, los 50 fragmentos utilizados no tenían ningún tipo de información previa de etiquetado.

Cabe destacar que las pruebas realizadas fueron hechas sobre el emulador proporcionado por el SDK de Android, ya que al momento del desarrollo no se contaba con ningún dispositivo móvil que soportará el sistema operativo Android.

A continuación se exponen algunos datos estadísticos recolectados de las pruebas realizadas:

Escenario:

Jugadores involucrados en las pruebas: 2
Total de clips utilizados: 50
Los clips pertenecían a 9 temas musicales diferentes.

Partidas Jugadas en total: 42
Partidas jugadas en modo normal: 28
Partidas jugadas en modo un sólo jugador: 14

Resultados obtenidos:

Partidas ganadas (se acertaron las 3 rondas): 28
Partidas perdidas (al menos una ronda fue fallida): 14
Promedio de etiquetas por clip: 6
Total de etiquetas: 640
Total de etiquetas únicas: 109
Etiqueta más utilizada por los jugadores: "hombre" 39 veces ingresada.
Etiquetas únicas involucradas en al menos una partida ganada: 99.
Etiquetas únicas involucradas en al menos dos partidas ganadas: 53.
Los principales *tags* verificados en rondas de partidas ganadoras fueron:

```
"hombre": 32 rondas.
"opera": 28 rondas.
"tranquilo": 26 rondas.
"coro": 24 rondas.
"mujer": 23 rondas.
"agua": 21 rondas.
"rock": 21 rondas.
"guitarra": 17 rondas.
"arabe": 16 rondas.
"ambiental": 16 rondas.
"electronica": 16 rondas.
"ingles": 15 rondas.
```

Rondas Bonus:

Total de bonus jugados: 17
Total de bonus ganados: 11
Total de bonus perdidos: 6

Uno de los resultados observados fue que dos clips específicos, siempre que aparecían juntos, eran marcados como similares. Se compararon las etiquetas de cada uno de ellos y se obtuvo una coincidencia de 3 sobre un promedio de 6 etiquetas.

Optimización de datos

Los datos recolectados de las diferentes partidas jugadas por los usuarios de **mGWAP** no siempre contienen la calidad deseada. En determinados casos los jugadores pueden, intencionalmente o no, ingresar etiquetas mal escritas o etiquetas que no están relacionadas con el clips, sino que intentan, por ejemplo, comunicarse con el compañero de juego. Es por esto, que es necesario antes de utilizar el conjunto de datos recolectados, aplicar sobre los mismos algoritmos que permitan subsanar de manera parcial al menos la situación mencionada anteriormente.

Algoritmos de optimización de datos

Los algoritmos propuestos permiten en una primera instancia, normalizar las etiquetas, esto es, pasarlas a letra minúscula quitando caracteres especiales y letras con acentos. Por último se buscan las palabras que están incluidas dentro del conjunto definido como "lista negra" y se eliminan. Dicha lista contiene las palabras que comúnmente son ingresadas por los usuarios para intentar comunicarse entre sí, como por ejemplo "iguales", "son diferentes". Una vez que el conjunto de datos original fue sometido a los procesamientos de optimización, es posible ejecutar operaciones de búsqueda optimizadas.

El algoritmo de optimización de datos y normalización de las etiquetas:

- Convertir a minúsculas y reemplazar las letras acentuadas por la misma letra sin acento. Este procesamiento se realiza sobre cada etiqueta recolectada.

- Eliminar, de las etiquetas, todos aquellos caracteres que no pertenezcan al alfabeto o al conjunto de números. Para las etiquetas de más de una palabra, se chequea que no existan múltiples espacios en blanco consecutivos, de ser así solo se deja uno.

El algoritmo de optimización de datos y eliminación de etiquetas inválidas, consiste en:

- Chequear que las etiquetas recolectadas, no se encuentren dentro del conjunto denominado “lista negra”.

- Eliminar las etiquetas que se encuentran dentro de la lista negra, se la elimina.

Al finalizar la optimización de los datos, opcionalmente se podría verificar manualmente el conjunto de etiquetas en busca de errores ortográficos, corregirlos y volver a aplicar la optimización de datos. Esto último puede resultar laborioso según el tamaño del conjunto de etiquetas recolectadas, pero es un paso muy eficaz para el mejoramiento de la calidad del conjunto.

Cabe destacar que por cada fragmento (clip) de audio existirán eventualmente etiquetas repetidas, esto es, etiquetas iguales ingresadas por diferentes usuarios. Esta situación enriquece la información obtenida por **mGWAP**, ya que una etiqueta ingresada varias veces por diferentes usuarios tendrá más valor que una etiqueta que fue ingresada sólo una vez para un clip determinado, dado que en el primer caso la etiqueta fue confirmada por más de una persona.

Procesamiento de datos

En esta sección del capítulo se describirán algunas maneras de procesar los datos para obtener información útil. Las que se describen no son todas las opciones, pero ilustran algunas de las posibles opciones.

Búsqueda de canciones a partir de una etiqueta

El algoritmo más sencillo podría ser el siguiente: buscar y recuperar todas las canciones que contengan una etiqueta determinada. Para esto debemos recuperar todos los fragmentos que contienen la etiqueta en cuestión. A partir del conjunto de fragmentos obtenidos podemos saber cuáles son las canciones que contienen dicha etiqueta dado que las canciones están compuestas, como ya vimos, por fragmentos o clips.

Búsqueda de canciones a partir de varias etiquetas

De la consulta anterior se desprende una un poco más amplia que permita buscar y recuperar canciones que cumplan con al menos una etiqueta del conjunto. Simplemente se debe tener en cuenta en lugar de una sola etiqueta, varias. Esto es, buscar los fragmentos que cumplan con al menos un elemento del conjunto de etiquetas en cuestión.

Búsqueda de canciones que contengan todas las etiquetas de un conjunto dado

Para obtener todas las canciones que contengan un conjunto determinado de etiquetas se deben buscar todos los fragmentos que contienen al menos una etiqueta del conjunto de etiquetas. Luego, agrupar los fragmentos utilizando como criterio de agrupación la canción a la que pertenecen. Una vez agrupados los fragmentos se debe contabilizar con cuántas etiquetas del conjunto coincide cada grupo de fragmentos. Las canciones de cada grupo que contenga todas las etiquetas del conjunto, serán las que cumplan con el criterio de selección.

Búsqueda de canciones que contengan una cantidad determinada de etiquetas

Este algoritmo es una variante del anterior, el cual permite obtener todas las canciones que cumplen con cierto número de etiquetas del conjunto. Para esto se debe seguir el procedimiento anterior, pero en lugar de buscar grupos que contengan todas las etiquetas, se deben buscar grupos que contengan al menos una cantidad determinada de etiquetas del conjunto.

Obtener canciones, indicando el momento de la etiqueta dentro de la canción

Supongamos ahora se quiere saber qué canciones cumplen con una determinada etiqueta, pero con la diferencia que se desea especificar en qué momento de la canción sucede. Para eso se utiliza la información registrada con los fragmentos. Una posible solución sería buscar todos los clips que contienen la etiqueta especificada, posteriormente listar todas las etiquetas junto con el tiempo de inicio del fragmento asociado más el instante en el cual se ingresó la etiqueta. De esta manera se obtendría para canción que contenga un fragmento con dicha etiqueta asociado, en qué momento de la canción el usuario ingresó la etiqueta.

Sugerencias de canciones a partir de una dada

Como primera opción lo que fácilmente se puede resolver es, dado una canción, sugerir otras similares. La primera consulta que podría realizarse para obtener las canciones similares es mediante la utilización de la información generada por el bonus. Para obtener dicho resultado se deben buscar todos los bonus en los cuales participó algún fragmento o clip de la canción y no fue elegido como el más diferente. Una vez obtenido el conjunto de partidas bonus, obtener las canciones a las cuales pertenecen los fragmentos que participaron en el bonus y quedaron emparejados como similares con los fragmentos de la canción dada a la consulta. De esta manera podemos obtener una aproximación de canciones similares entre sí.

El algoritmo anterior puede ser mejorado de forma que devuelva resultados más exactos. Una buena manera de hacer esto es buscar las coincidencias en las etiquetas de los fragmentos de la canción. Para conseguir eso vamos a utilizar la última consulta del punto anterior, que nos permite indicar la cantidad de etiquetas que deben coincidir. La consulta se aplica sobre los clips que cumplieron con la consulta del bonus y el conjunto de etiquetas será el que tiene asignado el clip original.

Autores que compusieron canciones similares a una canción determinada

Obtener autores que compusieron canciones similares a una canción determinada es un buen sistema de recomendación. Esta búsqueda podría implementarse utilizando el algoritmo anterior, “Sugerencias de canciones a partir de una dada” y a partir del resultado obtenido recuperar los autores de dichas canciones.

Prototipo Web para la consulta de canciones a partir de una etiqueta

Con el objetivo de ver los resultados obtenidos a partir de las diferentes partidas realizadas en el juego, se desarrolló un prototipo web que permite ingresar una etiqueta y obtener como respuesta una lista de canciones que fueron previamente etiquetadas con dicha etiqueta. Las canciones se muestran junto con su autor y con un botón que permite su reproducción. La Figura 7.1 muestra la pantalla del prototipo del buscador con un ejemplo resultados de la búsqueda.

Figura 7.1 - Prototipo del buscador de canciones a partir de una etiqueta determinada

El algoritmo utilizado en el prototipo recolecta todos los clips que fueron al menos una vez etiquetados con la etiqueta ingresada. A partir del conjunto de clips de resultado, se recuperan las canciones a las cuales pertenecen los clips.

CAPÍTULO 8

Conclusiones y líneas futuras de trabajo

Conclusiones

Habiendo culminado el proyecto, se puede llegar a dos conclusiones importantes. La primera es que, aún queda mucho por seguir investigando y desarrollando, ya que son muchas las ramas que se desprenden del tema en cuestión, tanto en el área de desarrollo del juego en sí, como en el área de implementación de los algoritmos de procesamiento de la información generada por **mGWAP**. El juego aún tiene mucho por mejorar en cuanto a usabilidad y prestaciones, como así también en lo referente a la utilización de la información generada, que al final de cuentas es la razón por la cual existe **mGWAP**.

La segunda conclusión, no menos importante, a la que se llega luego de haber estudiado un concepto apenas difundido como **Gwap**, es que presenta un gran abanico de aplicaciones posibles, lo cual permite sacar provecho de la inteligencia humana para entrenar sistemas computacionales, siendo voluntario el aporte de las personas involucradas.

Otro punto importante a tener en cuenta es la innovación que este tema produce en la región, no existen hasta el momento desarrollos que exploren temas como GWAP o Human Computation. Al no haber desarrollos similares en la región, mGwap proporciona la oportunidad de generar etiquetas para repertorios de música autóctona y utilizando terminología propia de la región. A partir de esto, resulta interesante pensar en diferentes aplicaciones de mGWAP en distintos ámbitos, como por ejemplo, en el ámbito educativo. Con los nuevos planes nacionales para la integración de los chicos a internet, por ejemplo "Conectar Igualdad[13]", se puede adaptar mGwap para ser utilizado desde un navegador de internet y proporcionar a los alumnos un juego que les permita escuchar y aprender de la música autóctona.

Experiencia adquirida

Desde el punto de vista teórico, hemos incorporado nuevos conocimientos del área de Human Computation, como así también en el área de **Gwap**. Y desde el punto de vista

[13] http://conectarigualdad.gob.ar/

práctico, los conocimientos adquiridos fueron variados. Por un lado fue necesario capacitarse en el desarrollo de aplicaciones Android, como así también estudiar el protocolo Bayeux y la utilización de la librería CometD, la cual implementa dicho protocolo. En cuanto a Java, ya teníamos adquiridos durante la carrera.

Líneas futuras de trabajo

A continuación se plantean algunas de las líneas futuras de trabajo que se podrían desarrollar, tanto por los autores del trabajo como por terceros. Algunos puntos tratan sobre mejoras a la implementación realizada, mientras que otras son futuras líneas de desarrollo de algoritmos de procesamiento de la información generada.

Mejora de la interface de usuario

Actualmente **mGWAP** funciona sin inconvenientes sobre dispositivos con teclado físico, pero en dispositivos con teclado virtual la interface de **mGWAP** puede no mostrarse apropiadamente debido al espacio que dicho teclado ocupa en pantalla. Una posible mejora sería poder dar soporte a dispositivos con teclados virtuales. Para esto, sería necesario rediseñar parte de la pantalla de juego para permitir que el teclado virtual se muestre sin perjudicar el juego.

Una alternativa posible para el rediseño de la interface que permita la utilización de teclados virtuales sería mostrar las etiquetas tanto del usuario como de su pareja en forma de chat, esto es intercalando las etiquetas que cada uno ingresa, en orden de llegada. Esto permitiría en un espacio más reducido mostrar las etiquetas, dando lugar en la pantalla al teclado virtual.

Otra posible mejora en cuanto a la interface sería el agregado de botones para controlar el volumen y pasar a la siguiente ronda sin escoger ninguna opción.

Optimización del modo un sólo jugador

El modo un sólo jugador permite que un usuario siempre pueda formar pareja para comenzar a jugar. Éste modo es implementado por un “bot” que imita el papel de un usuario determinado en una partida jugada anteriormente y almacenada por **mGWAP**.

La manera en que el “bot” selecciona la opción “Iguales” o “Diferentes” es a partir de estadísticas de etiquetas ingresadas. Aunque este mecanismo cumple con las expectativas, sería interesante intentar aplicar una mejora analizando los resultados obtenidos a través de un gran número de partidas realizadas con esta modalidad, con la intención de mejorar la selección que hace el “bot” a partir de la información existente.

Pruebas con un conjunto mayor de usuarios

Hasta el momento el juego ha sido probado sólo por los autores del trabajo, quienes realizaron sucesivas partidas con el objetivo de obtener información estadística. Una posible línea de desarrollo sería la utilización del juego por parte de un número mayor de personas, obteniendo de esta manera un escenario más apropiado para el análisis de la información generada.

Despliegue y *testing* del juego sobre dispositivos móviles reales

Al momento del desarrollo sólo pudo ser posible realizar las pruebas sobre el emulador de la plataforma Android proporcionado por el SDK de Android, siendo el mismo de gran utilidad para el desarrollo. Un avance importante sería instalar y realizar diferentes pruebas sobre dispositivos móviles reales que utilicen el sistema operativo Android.

Implementación del juego en otras plataformas móviles

A pesar de estar desarrollado en Java, **mGWAP** funciona exclusivamente en aquellos dispositivos móviles que utilicen la plataforma Android. Pero no existe ninguna restricción al desarrollo de **mGWAP** para otros dispositivos móviles con otros sistemas operativos, como pueden ser, iPhone OS, Symbian, Windows Mobile y demás.

Para cada caso sería necesario realizar una evaluación previa tanto de las características de la plataforma como de los dispositivos que la utilizan, para determinar si cumple con todos los requisitos necesarios para la implementación de **mGWAP**, como por ejemplo, que las dimensiones de la pantalla sean apropiadas.

Referencias bibliográficas

[1] Human Computation.

http://www.youtube.com/watch?v=qlzM3zcd-lk

[2] GWAP (Games With A Purpose).
http://www.cs.cmu.edu/~biglou/ieee-gwap.pdf

[3] Luis von Ahn.
http://www.cs.cmu.edu/~biglou/

[4] Google Image Labeler.
http://images.google.com/imagelabeler/

[5] Android.
http://developer.android.com

[6] Java Standard Edition.
http://java.com/es/

[7] Eclipse IDE.
http://www.eclipse.org/

[8] PostgreSQL.
http://www.postgresql.org/

[9] Apache Tomcat.
http://tomcat.apache.org/

[10] Eclipse ADT.
http://developer.android.com/guide/developing/eclipse-adt.html

[11] Open Handset Alliance.
http://www.openhandsetalliance.com/

[12] Android SDK.
http://developer.android.com/sdk/

[13] TagATune.
http://tagatune.org

[14] Creative Commons License.
http://creativecommons.org/licenses/by-nc-sa/2.5/es/

[15] CAPTCHA.
http://www.captcha.net

[16] reCAPTCHA.
http://recaptcha.net

[17] Input-agreement.
http://www.cs.cmu.edu/~elaw/papers/tagatune.pdf

[18] Output-agreement.
http://www.cs.cmu.edu/~biglou/GWAP_CACM.pdf

[19] ESP Game.
http://www.espgame.org

[20] Peekaboom.
http://www.cs.cmu.edu/~biglou/Peekaboom.pdf

[21] Listen Game.
http://ismir2007.ismir.net/proceedings/ISMIR2007_p535_turnbull.pdf

[22] MajorMiner.
http://majorminer.org

[23] Matchin.
http://www.cs.cmu.edu/~shacker/matchin.pdf

[24] Smartphones.
http://es.wikipedia.org/wiki/Smartphone

[25] Windows Mobile.
http://www.microsoft.com/windowsmobile/

[26] iPhone OS.
http://es.wikipedia.org/wiki/IPhone_OS

[27] Palm OS.
http://es.wikipedia.org/wiki/Palm_OS

[28] Symbian OS.
http://www.todosymbian.com/

[29] Historia de Java.
http://es.wikipedia.org/wiki/Lenguaje_de_programacion_Java

[30] El lenguaje Java.
http://java.sun.com/docs/white/langenv/

[31] Tutoriales Java.
http://java.sun.com/docs/books/tutorial/

[32] Comet.

http://alex.dojotoolkit.org/2006/03/comet-low-latency-data-for-the-browser/

[33] Bayeux.
http://svn.cometd.com/trunk/bayeux/bayeux.html

[34] CometD.
http://cometd.org/

[35] Jetty Continuations.
http://communitymapbuilder.org/display/JETTY/Continuations

[36] Hibernate.
http://www.hibernate.org/

[37] Java Persistence API.
http://java.sun.com/javaee/technologies/persistence.jsp

[38] JSON.
http://www.json.org

[39] Tomcat.
http://tomcat.apache.org/

[40] Java Community Process
http://jcp.org

[41] Proyecto PostGIS
http://postgis.refractions.net/

[42] Dojo Foundation
http://www.dojofoundation.org/

[43] "Mobile Design and Development". Brian Fling. O´Really. ISBN: 978-0-596-15544-5

[44] "Unlocking Android. A Developer´s Guide". W. Frank Ableson. Charly Collins. Robin Sen. Editorial: Manning

[45] “Hello, Android. Introducing Google´s Mobile Development Platform”, 3rd Edition. Ed Burnette. Editorial: The Prgamatic Bookshelf. ISBN-10: 1-934356-56-5

[46] “Human Computation”. Luis von Ahn, School of Computer Science, Carnegie Mellon University, 2005.