laSalle

UNIVERSITAT RAMON LLULL

Escola Tècnica Superior d'Enginyeria La Salle

Treball Final de Màster

Màster Universitari en Creació, Disseny i Enginyeria Multimèdia

iD-Stress: Diseño e implementación de un mini juego y elementos interactivos

Alumne
Sofia Swidarowicz Andrade

Professor Ponent
Oscar García Pañella

ACTA DE L'EXAMEN
DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Sofia Swidarowicz Andrade

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

iD-Stress: Diseño e implementación de un mini juego y elementos interactivos

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

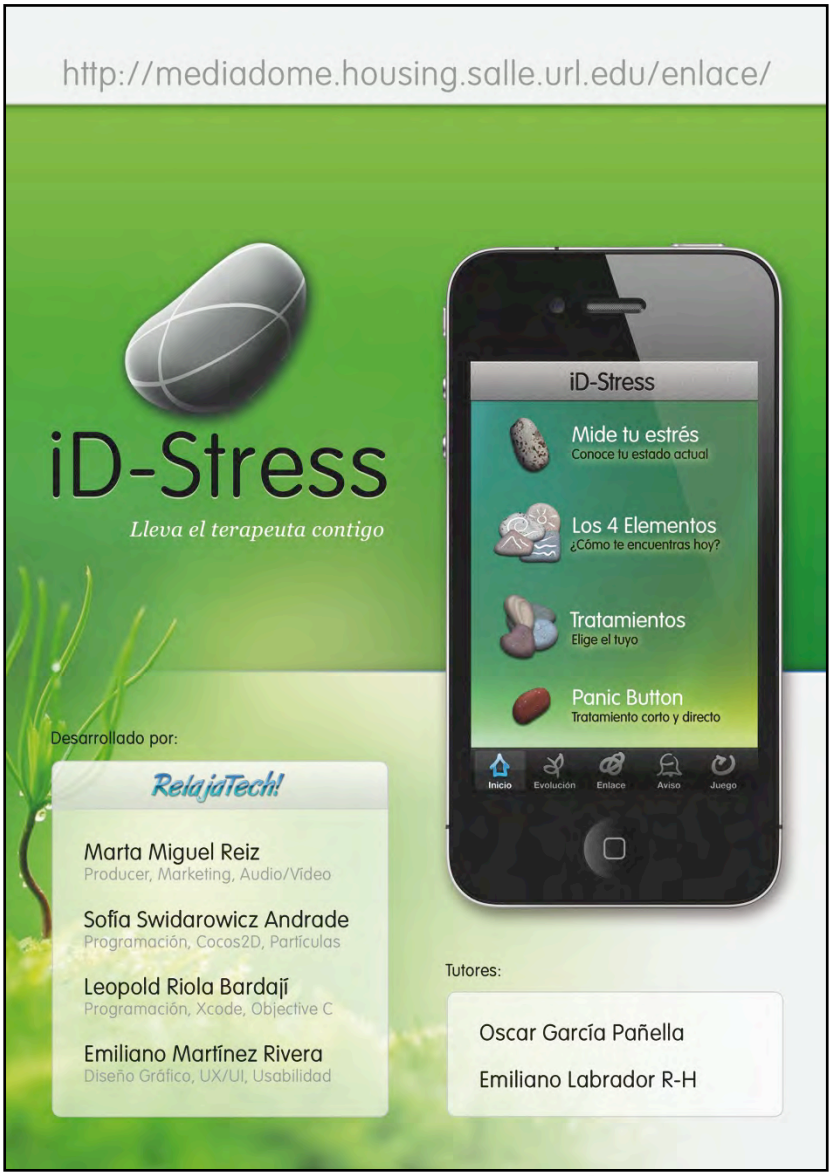
Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

iD-Stress:
Diseño e Implementación de un Mini Juego y Elementos Interactivos

Sofía Elizabeth Swidarowicz Andrade (ls23974)
Trabajo Final de Máster – Profesores Ponentes:
Óscar García Panella y Emiliano Labrador Ruiz de la Hermosa
La Salle – Universitat Ramon Llull
Máster en Creación, Diseño e Ingeniería Multimedia – Junio 2011



RelajaTech!

Abstract

El objetivo de este proyecto ha sido diseñar y desarrollar una aplicación para iPhone que ayude a gestionar el estrés en colaboración con el Centro Enlace. Se realizó en un equipo de cuatro integrantes, por lo que este documento presenta sólo una parte del total.

En esta memoria se hace especial mención al proceso del diseño de un mini juego y elementos interactivos como complementos al resto de la aplicación que proporcionarán la sensación lúdica al usuario.

Abstract

L'objectiu d'aquest projecte és dissenyar i desenvolupar una aplicació per telèfon móvil, específicament iPhone, que faciliti a gestionar l'estrès en col·laboració amb el Centro Enlace. Es va realitzar en un equip de quatre integrants, de manera que aquest document presenta només una part del total.

En aquesta memòria es fa especial menció al procés de disseny d'un mini joc i als elements interactius com a complements per la resta de l'aplicació que proporcionaran la una sensació lúdica a l'usuari.

Abstract

The purpose of this project has been to design and develop an application for mobile phones, specifically the iPhone, an application that helps manage stress in collaboration with Centro Enlace. A team of four members was formed, and this document presents only one part of the total.

This report makes special mention to the design process of complementary mini-game and interactive elements to the rest of the application that provides a lucid sensation to the user.

Índice

Índice de Figuras

CAPITULO I

INTRODUCCIÓN

1. Introducción

1.1 Resumen

La memoria que se presenta a continuación, representa el último peldaño para la titulación del Master de Creación, Diseño e Ingeniería Multimedia de BES La Salle URL, en ella se pretende explicar el proceso técnico de elaboración de una aplicación para iPhone -solicitada por una empresa colaboradora- cuyo objetivo principal es el de gestionar el estrés de tal manera que el usuario no requiera un especialista presente para llevarlos a cabo.

El proyecto fue realizado por dos técnicos programadores (entre ellos mi persona), un diseñador gráfico y un producer; conformando así un equipo multidisciplinar. Cada integrante del equipo ha tenido que desarrollar una memoria, planteando, explicando y describiendo los procesos de desarrollo del área particular que le competen.

De este modo, éste documento estará orientado principalmente a plantear y explicar el aspecto interactivo de la aplicación, y de como se pudo desarrollar un mini juego y diversos efectos gráficos empleando un framework [1.1] con diversas librerías de desarrollo para aplicativos interactivos y juegos en dos dimensiones, denominado Cocos2D y que, unido al API [1.2] propio de desarrollo de sistemas para iPhone, denominado Xcode, se pudo crear la aplicación bautizada como ID-Stress.

Primero se describirá la información teórica necesaria para el desarrollo de la aplicación y luego la parte práctica del desarrollo y retos presentados durante la implementación.

1.2 Marco Teórico

En la actualidad la telefonía móvil representa la forma de comunicación más importante en el mundo. Se ha desarrollado en un periodo de tiempo relativamente corto gracias a la necesidades generadas por la globalización, por lo que debe mantenerse en constante evolución y desarrollo. Ya no sólo encontramos dispositivos capaces de establecer llamadas telefónicas sino también aquellos que además integran un sin número de tareas y aplicaciones que facilitan las labores de oficina (hojas de cálculo, toma de notas, correo electrónico, navegación web) y permiten al mismo tiempo el esparcimiento en momentos de ocio (juegos, música, imágenes, navegación web); son conocidos como Smartphones o teléfonos inteligentes y falta muy poco para que sean considerados un PC (Personal Computer) en toda regla.

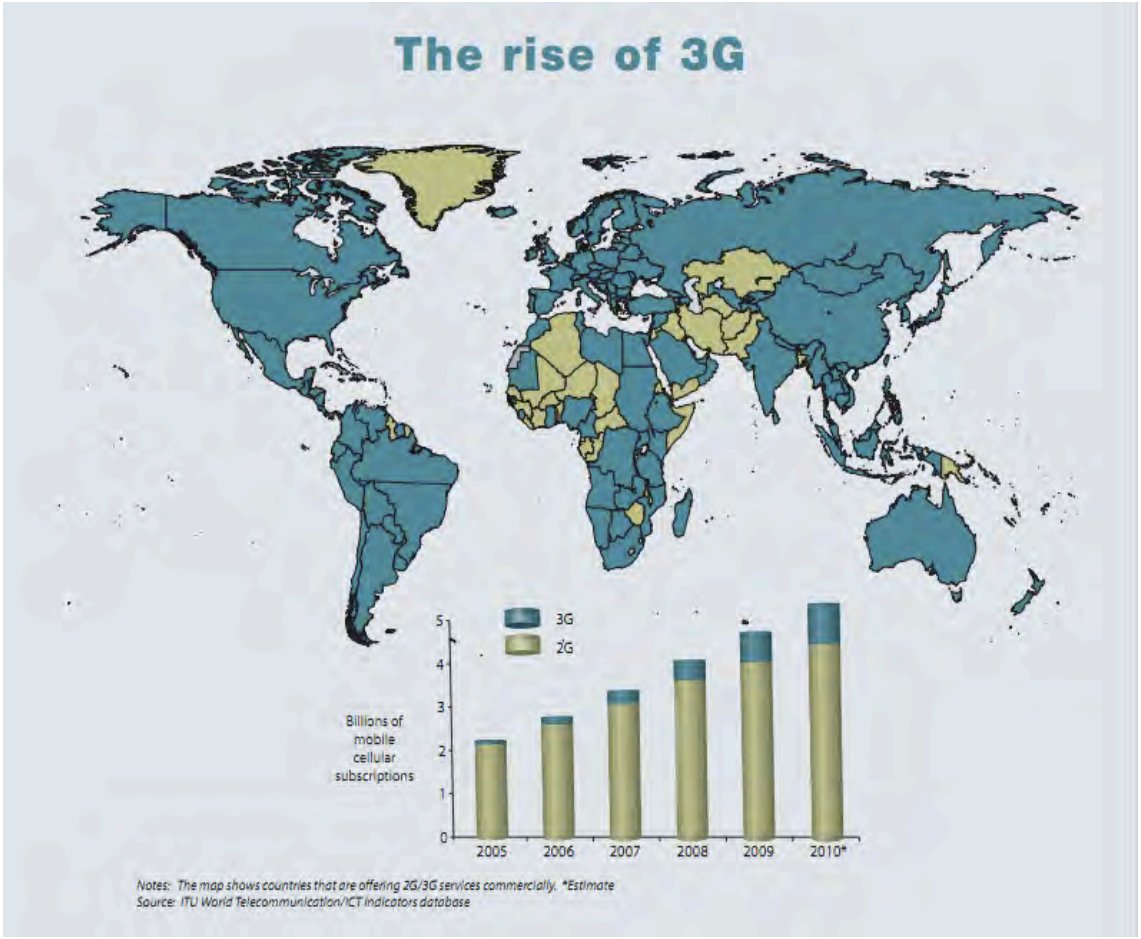


Figura 1 Países que ofrecen conexión 2G/3G hasta el año 2010.

El auge de los Smartphone [1.3] en la última década, ha propiciado oportunidades de negocio que son reflejadas en aplicaciones de diversa índole, que pueden adquirirse a precios relativamente modestos en tiendas virtuales, generando ganancias para el inversor y para el usuario final dependiendo del volumen de la descarga.
Las ventas de smartphones aumenta cada día a un ritmo vertiginoso. MobileMix [1] es una firma de publicidad que se encarga de analizar los movimientos y las tendencias del mercado

móvil ofreciendo estadísticas sobre los resultados mensuales y anuales que obtienen sobre el tema. Para el año 2010, las mayores ganancias del mercado en la venta de aplicaciones móviles en Estados Unidos de América, fue para iPhone de la compañía Apple y Android de la archiconocida marca Google, ambos en partes iguales.

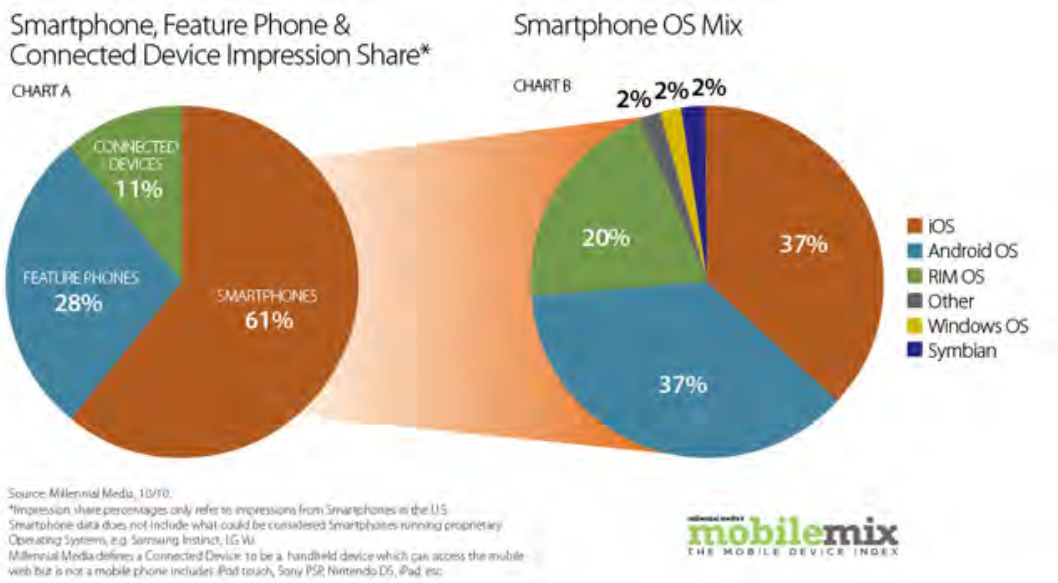


Figura 2 Porcentaje del mercado que posee cada Smartphone en USA

Algo similar sucede en Europa, de acuerdo a comScore MobiLens en un estudio realizado en cinco países europeos (Francia, Reino Unido, Alemania, España e Italia) Apple le gana por ahora a los dispositivo móviles con Android con un poco más de la mitad de subscriptores.

Apple iOS vs. Google Android Across Media Devices
3 Mo. Avg. Ending February 2011
Total EU5 Mobile Subscribers, Age 13+
Source: comScore MobiLens

	Total Installed Base (000)	Share (%) of Mobile Subscribers
Total Subscribers	**233,500**	**100.0%**
Apple iOS	28,873	12.4%
Google Android	13,368	5.7%

Figura 3. Volumen de ventas de dispositivos Smartphones en Europa

Los datos de ComScore sobre España son muy buenos y se pueden resumir así de Diciembre de 2009 a Diciembre de 2010 [2]

- más de **13 millones de usuario**s de Smartphones (por delante de Francia)
- **37% de penetración** de smartphones (10% incremento respecto al año anterior)
- Número de líneas móviles respecto al nº de habitantes: **116%**

Las categorías más populares vendidas en el App Store de Apple son:

1 - Books (57390 active)
2 - Games (51987 active)
3 - Entertainment (39411 active)
4 - Education (29529 active)
5 - Lifestyle (25791 active)

La firma consultora Research2Guidance, especializada en investigación de tecnologías móviles, muestra un informe detallado sobre el mercado de las aplicaciones móviles para salud y proporciona datos, cifras clave de mercado, tendencias tecnológicas y sociales que se han obtenido analizando y encuestando a los principales actores de este mercado y revisando las cifras de un sector que se espera crezca un 807% hasta 2013, mueva 17.5 billones de dólares en 2012 y llegue a 500 millones de personas usando aplicaciones médicas en dispositivos móviles para 2015 [3].

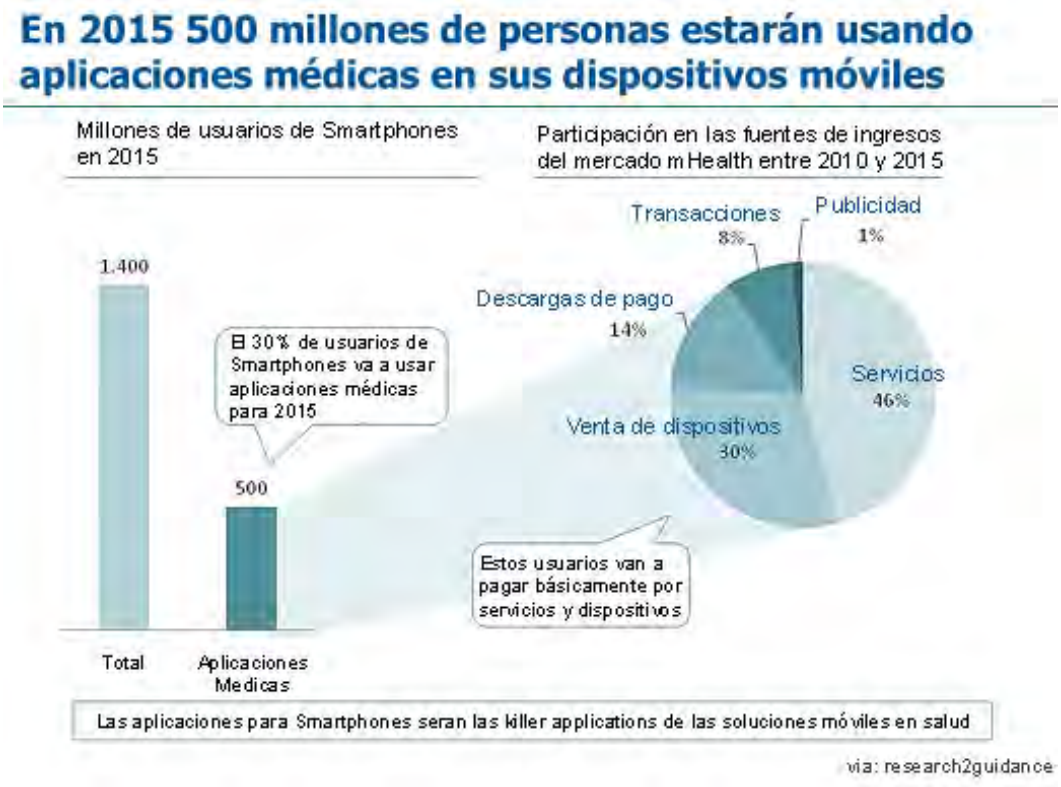


Figura 4. Porcentaje de aplicaciones de smartphones en el área de la salud

Por ello, no es de extrañar que diversas compañías quieran invertir en el desarrollo de aplicaciones para este sistema operativo móvil generando también, la utilización de herramientas tecnológicas diversas que aprovechan las prestaciones del dispositivo para facilitar la creación de interactivos, juegos, audio, video y que puedan embeberse en conjunto en una sola aplicación.

1.3 Empresa Colaboradora

Como proyecto final de Master, se ha concertado con una empresa, el desarrollo de una aplicación que servirá para su beneficio, tanto para propósitos de investigación como para la obtención de una versión lo suficientemente avanzada para ser vendida a un público objetivo, lo que nos dará experiencia en torno a la producción de un proyecto multimedia, a controlar y manejar el tiempo de desarrollo, a lidiar con los clientes, a utilizar diversas herramientas tecnológicas para solventar el problema propuesto por éste y por último poder añadirlo a una tienda virtual móvil.

Así pues, nuestra empresa colaboradora está constituida por una clínica franco-española de medicina tradicional y alternativa, dirigida por Ana Lombard, especialista en sofrología[1.4] y denominada Enlace, que se encuentra ubicada en el centro de la ciudad de Barcelona, España. Se caracteriza por agrupar distintos profesionales que mediante el análisis de los problemas del ser humano en la sociedad actual pueden adaptarse y elaborar técnicas en la optimización de las patologías de sus pacientes. [4]

El centro representa hoy un espacio para el tratamiento equilibrado del cuerpo y la consciencia, todos los profesionales trabajan en estrecha colaboración y con una única filosofía: "*Ser complementarios para ayudar y acompañar a sus pacientes en el camino del bienestar físico y psíquico".* [4]

Algunas de las patologías más frecuentes entre los pacientes que asisten a consulta en Enlace, son las producidas por el estrés como el insomnio, falta de concentración, ansiedad y que deben ser tratadas con tratamientos que Ana Lombard ha desarrollado en el transcurso del tiempo.

1.4 Descripción del Problema

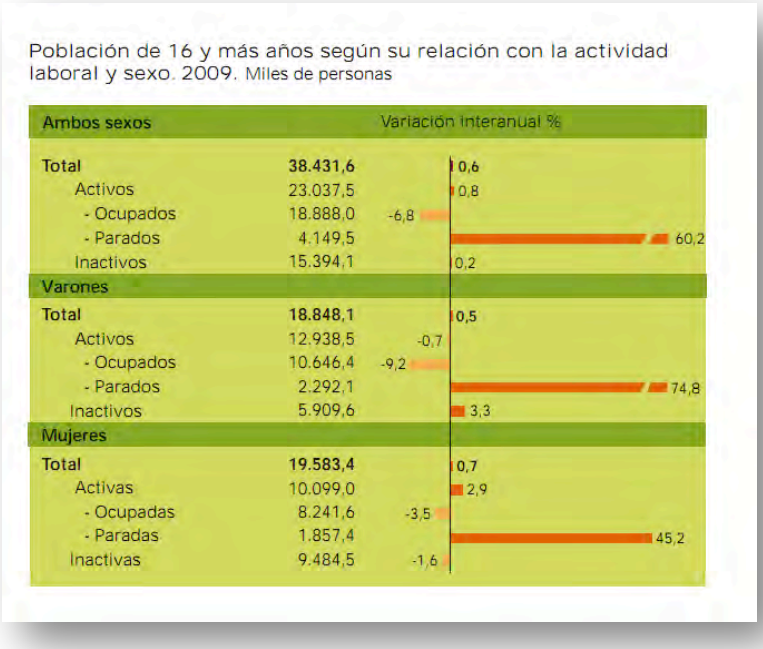
En la actualidad, el consumidor español, a nivel demográfico, según las estadísticas del INE de 2008, España posee una mayoría de población de edad comprendida entre los 20 y 44 años de edad. Según el informe "**España en cifras 2010"** del INE, el gasto medio por persona es de 11.801 euros al año. El mayor valor se registra en hogares unipersonales, con 21.596 euros, cuando la persona es menor de 65 años y 15.182 euros, cuando tiene 65 años o más. [5]

La figura cinco (5) muestra cómo, según la Encuesta de Población Activa (EPA), en 2009 el número de activos se sitúa en algo más de 23 millones de personas. La tasa de actividad alcanza así el 59,9% de la población de 16 y más años, siendo la femenina el 51,6% y la masculina el 68,6%. El número de ocupados desciende un 6,8% respecto a 2008.

Por sectores económicos, la construcción es el más afectado, un 23,0% menos de ocupados, seguido de la industria (13,3%). La tasa de paro en el último trimestre de 2010 en un 20,33%, y la tasa de población activa se sitúa en un 59,99%. [5]

En cuanto a la salud, según datos avance de la reciente Encuesta Europea de Salud en España, en 2009, siete de cada 10 personas de 16 años o más afirman que su estado de salud en los últimos 12 meses ha sido bueno o muy bueno. El 27,3% de la población ocupada ha faltado al trabajo por un problema de salud en ese mismo periodo.

Figura 5. Gasto medio por persona 2008



Entre los problemas crónicos más diagnosticados figura el dolor de espalda (24,9%) y la hipertensión (19,7%). El tabaco repunta ligeramente su consumo (31,5%) respecto de la última cifra registrada [5].

Dentro de los riesgos laborales de carácter psicosocial, el estrés laboral y el síndrome de quemarse por el trabajo (burnout) ocupan un lugar destacado, pues son una de las principales causas del deterioro de las condiciones de trabajo, y fuente de accidentabilidad y absentismo. Una sentencia del Tribunal Supremo de Octubre del 2000, ratificando la sentencia de 2 de noviembre de 1999 dictada por la Sala de lo Social del Tribunal Superior de Justicia de la Comunidad Autónoma del País Vasco, respalda legalmente esa importancia al reconocer el

síndrome de quemarse por el trabajo como una dolencia psíquica causante de periodos de incapacidad temporal, y como un accidente laboral. [6]

De acuerdo a un estudio de la Fundación Europea para la Mejora de las Condiciones de Vida y Trabajo (1999) el 28% de los trabajadores europeos padece estrés [6]

- El 20% padece burnout.
- Más de la mitad de los 147 millones de trabajadores afirman que trabajan a altas velocidades y con plazos ajustados.
- Más de un tercio no pueden ejercer ninguna influencia en la ordenación de las tareas
- Más de un cuarto no puede decidir sobre su ritmo de trabajo.
- Un 45% afirma realizar tareas monótonas.
- Para un 44% no hay posibilidad de rotación.
- El 50% realiza tareas cortas repetitivas.
- Se piensa que estos «estresores» relacionados con el trabajo han contribuido a importantes manifestaciones de enfermedad:
    - un 13% de los trabajadores se quejan de dolores de cabeza
    - un 17% de dolores musculares
    - un 20% de fatiga
    - un 28% de «estrés»
    - un 30% de dolor de espalda
    - muchos otros, de enfermedades que pueden poner en peligro la vida
- Una estimación moderada de los costes que origina el estrés relacionado con el trabajo apunta a unos 20 000 millones de euros anuales.

Más de un 15% de personas (según los estudios más recientes, esta cifra podría alcanzar incluso el 25%) a lo largo de su vida sufrirán algún trastorno de ansiedad, como por ejemplo ataques de pánico y agorafobia, que lo padecen entre un 1,5 y un 3,5% de personas. El estrés laboral puede ser un factor de vulnerabilidad para llegar a sufrir este tipo de trastornos, aunque no es el único [7]. Las personas estresadas acuden generalmente en primera instancia al médico. Sus quejas más habituales suelen ser ansiedad, dolor y depresión. Pues bien, en el año 2001, si se tiene en cuenta los datos "sólo" de la receta médica oficial, en España se consumieron casi 35 millones de envases de fármacos de tipo ansiolítico o tranquilizante. Casi un envase por habitante.

En cuanto a los depresivos, de los tipos que hoy en día más se consumen, en el mismo año se recetaron oficialmente más de 14 millones de envases. De las personas que acuden al médico de atención primaria, el 21% de los pacientes consume ansiolíticos y/o antidepresivos. [7]

El mundo laboral ha experimentado una transformación importante en las últimas décadas en nuestro contexto sociocultural. Así, las nuevas exigencias del trabajo y el desajuste entre los requisitos del puesto de trabajo en las organizaciones y las posibilidades de rendimiento de cada sujeto han originado la aparición de nuevos riesgos denominados psicosociales, entre ellos el síndrome de burnout (agotamiento emocional, despersonalización o deshumanización y falta de realización personal en el trabajo), cuya prevalencia se ha ido incrementando y que ha venido a constituirse en un problema social y de salud pública que conlleva, por tanto, un gran coste económico y social [8].

Figura 6. Demanda vs Control del estrés

En la figura seis (6) se puede observar cómo a mayor demanda y menor control, el nivel de estrés crece considerablemente. Los factores ambientales que provocan el estrés laboral, pueden clasificarse, según la investigación de MAN-WAN LO [9] en dos grupos: los macro-estresores y los micro – estresores, como indica el siguiente cuadro:

Los macro- estresores son factores estresantes que están relacionados con el trabajo general de una organización o empresa, e incluyen:

- Política: el poder contribuye a aumentar las ventajas personales.
- Desarrollo profesional: Dificultad y falta de oportunidades para conseguir un training adecuado
- Recompensas: Sistemas injustos de recompensa donde éstas no están basadas en el desempeño.
- Participación: Falta de oportunidades para participar en la toma de deciones.
- Uso de pocas habilidades: Falta de retos e imposibilidad de utilizar el talento.
- Estilo Supervisor: No considera las necesidades de los subordinados
- Estructura de la organización: políticas restrictivas, cadena de mando y estructura confusa

WORK ENVIRONMENT STRESSORS	STRESS INDIVIDUAL FACTORS	STRESS OUTCOMES
MACROSTRESSOR - politics career development rewards participation underutilization supervisory style organization	BEHAVIOR PATTERN - Type A/Type B (decisiveness, self-confidence, need levels, locus of control, tolerance of ambiguity)	PSYCHOLOGICAL - satisfaction commitment tension tension discharge
MICROSTRESSOR - role ambiguity role conflict workload career progress responsibility time pressure communication work relationship change job scope	DEMOGRAPHIC DIFFERENCE - position held years in field sex age	PHYSICAL/ BEHAVIORAL - blood pressure cholesterol doctor visits smoking drinking
		ORGANIZATIONAL - absenteeism turnover quantity quality cost

Figure 1. Stress model for information systems professionals (derived from Ivancevich [6; 7], Saleh [9], Sauter [10], and Weiss [14]).

Los micro-estresores son factores que están relacionados con el trabajo individual:

- **Ambigüedad de roles**: Falta de objetivos definidos, autoridad, expectativas y alcance de las responsabilidades
- **Conflicto de rol**: Recibir peticiones incompatibles con la posición.
- **Sobrecarga cuantitativa**: tener mucho trabajo por hacer
- **Sobrecarga cualitativa**: obtener requerimientos demasiado complejos como para hacerlos bien.
- **Progreso de la carrera profesional**: No tener suficientes oportunidades
- **Responsabilidad sobre otras personas**: Ser responsable del trabajo de esas personas y no ser capaz de ayudarles.
- **Presión de tiempo**: Entregas demasiado ajustadas
- **Comunicación**: no estar al corriente de "lo que pasa" o "no ser informado"
- **Relaciones personales**: Conflictos entre el personal
- **Cambios**: Cambios en las actividades del día a día
- **Alcance del trabajo**: No obtener feedback, falta de variedad en los deberes a desarrollar.

Así pues, la empresa colaboradora, la Clínica Enlace, nos propuso un proyecto de desarrollo de una aplicación móvil de ejercicios interactivos con el objetivo de enfrentarse y regular el estrés del usuario hasta eliminarlo, respondiendo al **problema social** descrito anteriormente. La idea nació de la experiencia de Ana Lombard (www.enlacebcn.com) como experta en terapias de tratamiento de gestión del estrés, concentración, relajación, insomnio, entre otros tratamientos. Tras muchos años de estudios y trabajo, ideó un sistema funcional para ayudar a que las personas puedan "auto curarse".

Se trata de que el paciente aprenda a no depender del terapeuta y empiece a realizar diversos ejercicios por su cuenta, de tal forma que aprenda y logre enfrentarse a sus problemas y dificultades mediante técnicas que establecen diversos niveles de compromiso entre las disciplinas de la relajación y la sofrología, siempre teniéndose en cuenta las particularidades de las diferentes culturas del mundo.

1.5 Solución Propuesta

Nos proponemos acercar los **tratamientos** de **Ana Lombard** del Centro Enlace a las personas interesadas sin necesidad de la presencia de un terapeuta. Queremos satisfacer al cliente externo y a los futuros usuarios, por lo que ha sido necesaria una profunda investigación para la creación de un producto innovador. De esta forma, nos planteamos el objetivo de ofrecer una producto destinado a las personas que sufren de estrés laboral, bien como complemento para los usuarios que ya siguen algún tratamiento; bien como alternativa, para las personas que no quieren o no pueden acudir a un especialista.

La mejor forma de llevar a cabo dicha labor es mediante una aplicación para móvil o smartphone, ya que un alto porcentaje de los Españoles posee uno de acuerdo a los datos de ComScore [2]:

- más de **13 millones de usuario**s de Smartphones (por delante de Francia)
- **37% de penetración** de smartphones (10% incremento respecto al año anterior)
- Número de líneas móviles respecto al nº de habitantes: **116%**

Así pues, la aplicación propuesta pondría al servicio de todos el acceso a estas terapias.

La plataforma seleccionada para implementar dicho proyecto, por ofrecer una gran popularidad en el público objetivo, ha sido el iPhone, en cualquiera de sus versiones; el nombre que se ha dado a la aplicación es **ID-Stress**, propuesto por la empresa colaboradora. De esta forma la aplicación se descompondrá en varios módulos, partiendo desde los tratamientos propuestos por la Clínica Enlace -que son locuciones que dan las instrucciones necesarias para inducir a los pacientes a un estado de relajación que ayude a establecer el equilibrio entre cuerpo y mente- y siguiendo con los complementos a éstas locuciones que ofrecerán una experiencia interactiva mucho más memorable.

Por su plataforma de desarrollo y la temática que abarca, esta aplicación va dirigida a un público adulto, de edades comprendidas entre 25 a 50 años, que se encuentren en constante movimiento o estrés laboral y que estén interesados en mejorar su bienestar y gestionar dicho estrés.

Hay que considerar que las aplicaciones que se desarrollan para iPhone deben cumplir reglas y requisitos, por lo que se han de estudiar los elementos pertenecientes a su SDK (Software Developer Kit) conformado por el XCode[1.5] y el Interface Builder[1.6], y que se ajusten a los requerimientos tanto de Apple como los de nuestro cliente Enlace. Posteriormente se deberán hacer testeos de usabilidad con potenciales usuarios, que nos aportarán la información necesaria para mejorar el producto final.

A nivel particular, el desafío con este proyecto se presenta en el desarrollo de los aplicativos: "Cuatro Elementos" y el "Mini Juego" que serán realizados con una herramienta denominada **Cocos2D para iPhone** que involucra una curva de aprendizaje elevada pero plausible y que deberá realizarse en paralelo con el resto de la estructura de la aplicación.

Todas las características mencionadas anteriormente cumplen las expectativas de un proyecto multimedia, que involucra los parámetros de innovación, creatividad, tecnología y diseño; uniendo las habilidades de diferentes perfiles profesionales, por lo que la aplicación se desarrollará bajo estas premisas.

1.6 Estado del Arte

A continuación se analizarán las principales aplicaciones que se encuentran disponibles en la Apple Store, aunque también consideramos aplicaciones que se encuentran en plataformas diferentes al iPhone, como en Web o en el sistema operativo Android que abordan la temática de nuestro proyecto, en la categoría "Salud y Bienestar" tienen como objetivo principal la

realización de tratamientos de relajación, hipnosis, ayuda al insomnio, gestión del estrés, ejecución de sonidos y manipulación de imágenes.

1. **Hästens MindSpa iMeditation.**

Desarrollador: Neurotech INC
Precio: 3,99€
Valoraciones: 27 valoraciones
"What a waste of time. It's Awful" por el usuario Grinder70000007
Publicado: 15/01/11
Categoría: Salud y Forma Física
Tamaño: 98.8 Mb
Versión 1.0
Idioma: Inglés
Sitio web del autor: http://www.ibandler.com/

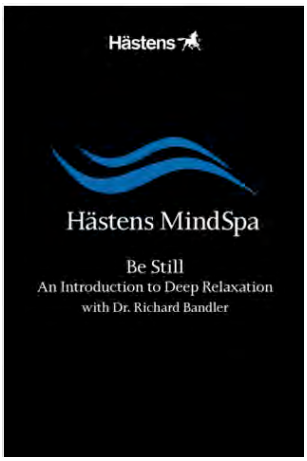


Figura 7. MindSpa iMeditation

Descripción: Utiliza la estimulación Auditorio-Visual (AVS). El audio y las imágenes están sincronizados de manera que el parpadeo de la luz y el sonido inducen a un estado de relajación. Según sus creadores "La luz y el sonido influyen enormemente en la actividad de las ondas cerebrales. De manera natural, la mente encuentra modelos de luz parpadeante relajantes". La estimulación actúa sobre las ondas cerebrales beta, alfa, theta y delta. Esta app se encuentra avalada por la disciplina del Dr. Richard Bandler, autor del libro "Get the life you want" y co-creador de la Programación Neurolingüística.

Hallazgo positivo: Se trata de una aplicación interesante, a pesar de que no tiene ninguna opción configurable ni posibilidad de entender muy bien lo que el usuario está haciendo, aunque esto quizá también forme parte del programa de meditación que ofrece.

Hallazgo negativo: Tras probar la aplicación hemos visto que, durante los ejercicios, no hay posibilidad

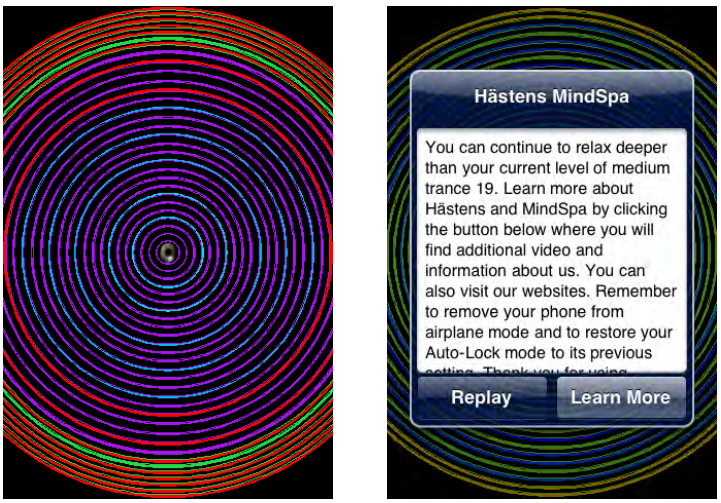


Figura 8 MinSpa Meditation

de detenerlos ni de obtener información acerca de lo que estamos haciendo. Tras unos 10 minutos de locución acompañada de la voz del Dr. Bandler, la aplicación se detiene y nos muestra una advertencia en forma de pop-up y nos ofrece la posibilidad de visitar la web para información sobre el autor.

2. Stress Free, de Louise L. Hay

Desarrollador: OceanHouse Media, Inc.
Precio: 4,99€
Valoraciones: 1 valoración
"I adore it!" por el usuario Vicvin el 04/11/10
Actualizado: 23/09/10
Categoría: Salud y Forma Física
Tamaño: 48.2 Mb
Versión 1.03
Idioma: Inglés
Enlace: http://www.oceanhousemedia.com/products/

Figura 9. Stress Free

Descripción: Se trata de una colección de 60 autoafirmaciones diseñada para aliviar el estrés y motivar cambios positivos en la vida de los usuarios. Ofrece la posibilidad de seleccionar una autoafirmación al azar o seleccionar una lista de locuciones.
Estas autoafirmaciones duran entre unos 20 segundos y están acompañadas de música. Incluye un audio de 30 minutos con música y afirmaciones subliminales (el usuario sólo escucha una melodía).

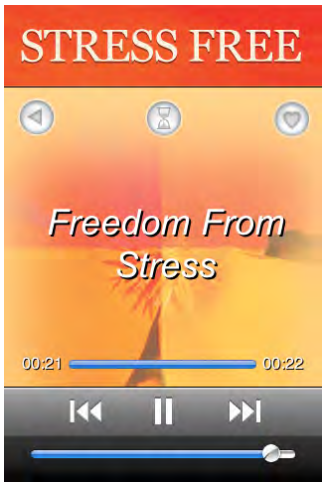


Figura 10. Stress Free

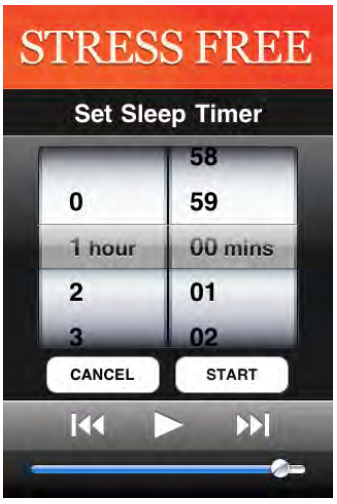


Figura 11. Stress Free

Hallazgo positivo: Instrucciones: Aquí nos ofrece información sobre las diferentes playlists y sobre cómo activar el temporizador para dormir. El temporizador es muy útil y sencillo de activar.

Hallazgo negativo: Durante la locución no se muestra más que el reproductor en la pantalla.

3. Natural Sounds For Me

Licencia: Creative Commons
Desarrollador: The Sound Waves of Nature.
Precio: ninguno
Idioma: Inglés, Polaco y Lativo
Enlace: http://naturesoundsfor.me/

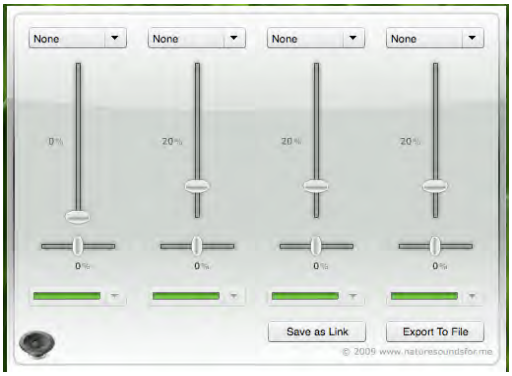


Figura 12. Natural Songs For Me

Descripción: *Natural Sounds for me* es una aplicación Web que funciona como un sencillo gestor de sonidos para ser utilizado cuando el usuario trabaja o estudia en el ordenador. Se aprecian sonidos de la naturaleza como: tormentas, olas, abejas, ríos, fuego, lluvia, entre otros. Incluso puedes reproducir hasta cuatro sonidos a la vez y luego salvarlos en tu ordenador para escucharlos offline.

Hallazgo positivo: Lo interesante de este producto es que intenta ser un banco de sonidos creado por los propios usuarios, es decir, cada usuario puede añadir sonidos que le resulten relajantes y de esta forma aumentar el número de sonidos disponibles.

Hallazgo negativo: Solo está disponible en la Web, no hay forma de descargar el aplicativo a tu ordenador.

4. iSleep

Desarrollador: Alejandro Luengo Gómez
Precio: 1,59€
Valoraciones: 8 valoraciones
"Está bien. La uso todas las noches y me va bien" por el usuario Fernando J. Expósito el 16/03/11
Actualizado: 25/01/11
Categoría: Salud y Forma Física
Tamaño: 133 Mb
Versión 1.1
Idioma: Inglés y Español
Enlace: http://www.atomstudios.es/web/

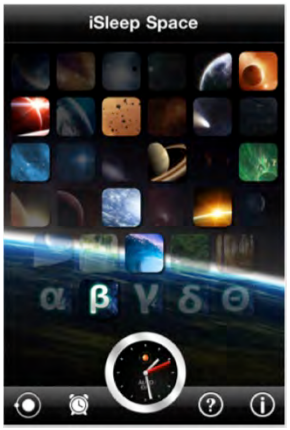


Figura 13. iSleep

Descripción: iSleep es una aplicación que se basa en una intensa investigación de neurociencia. Utiliza tonos binaurales, que combinados con sonidos naturales y melodías, ayudan a dormir mejor, concentrarse y relajarse. iSleep Space está formado por 24 melodías o sonidos inspirados en el espacio, 5 sonidos naturales y 5 ondas cerebrales.

El usuario puede escoger varios de esos sonidos y mezclarlos a su gusto, para crear un sonido agradable. La aplicación dispone de un texto explicativo que detalla las propiedades de cada onda cerebral, para que el usuario las utilice según su necesidad. iSleep Space tiene muy buena crítica en la AppStore y es una de las aplicaciones más descargada en este ámbito.

Hallazgo positivo: iSleep tiene la opción de activar un "Modo automático" que modifica las combinaciones de sonidos aleatoriamente para crear ambientes totalmente distintos. También hay la opción de poner una cuenta atrás para desactivar la aplicación según el usuario estime que va a necesitarla.

Hallazgo negativo: Los sonidos son monótonos y se repiten continuamente. El modo aleatorio genera sonidos poco acertados.

5. SYMBIOLINE

Desarrollador: SIMBIOFI.
Precio: 399€
Categoría: Salud y Forma Física
Idioma: Inglés/francés
Plataforma: Windows XP y Windows Vista
Contenido: Modulo experto de variabilidad de los latidos del corazón, ejercicios 3D.
E-Learning: Guía interactiva del protocolo ECG Bundle.
Enlace: http://www.symbiofi.com/

Figura 14. Symbioline

Descripción: Es una aplicación multimedia para ordenador, diseñado para la educación psicológca del estrés y las terapias de autoayuda. Ofrece ejercicios básicos de relajación.

Hallazgo positivo: Permite al usuario controlar su estrés y ansiedad a través de un asistente de biofeedback cardíaco.

Hallazgo negativo: Es extremadamente cara.

6. SymbioFeel

Desarrollador: Mental Workout
Precio: 7,99€
Valoraciones: No tiene
Actualizado: 15/12/09
Categoría: Salud y Forma Física
Tamaño: 82.4 Mb
Versión 1.1
Idioma: Francés
Enlace:
http://www.symbiofi.com/fr/produits/symbiofeel

Enlace del desarrollador:
http://www.mentalworkout.com/

Figura 15. SymbioFeel

Descripción: Esta aplicación es la adaptación a iPhone de la aplicación web que hemos detallado anteriormente. SymbioFeel es un conjunto de aplicaciones en francés respaldada por el doctor Dominique Servant. La aplicación analizada es la de relajación, pero todas tienen el mismo tipo de funciones.

La aplicación dispone de cuatro apartados:

- En primer lugar, hay videos del doctor Dominique Servant, explicando una breve introducción sobre el tema y dando la bienvenida.
- El segundo apartado son consejos en formato texto sobre la meditación y relajación.
- En tercer lugar se puede encontrar el apartado de ejercicios.
- En el último apartado hay un test de autoevaluación de la relajación, que contiene 10 preguntas y muestra un resultado final y las puntuaciones anteriores.

Hallazgo positivo: Los ejercicios son instrucciones en formato voz. Ésta puede ser mezclada con una música de fondo ambiental y con un tema de imágenes (agua, naturaleza, cielo…), que serán mezclados con la voz durante el ejercicio. MentalWorkout, la empresa desarrolladora, ha ganado premios por sus aplicaciones en éste ámbito, colaborando con varios expertos y organizaciones especializadas.

Hallazgo negativo: Los vídeos tienen mala calidad. Tarda mucho en cargar el sonido de fondo.

7. MyCalmBeat

Desarrollador: e-Faces & Names
Precio: Gratuita
Valoraciones: 4/5 estrellas
Actualizado: 21/07/10
Categoría: Salud y Forma Física
Tamaño: 1,8 Mb
Versión 1.1
Sistema: Android
Idioma: Inglés
Instalaciones: 10.000 – 50.000
Web:
https://www.mybrainsolutions.com/mycalmbeat

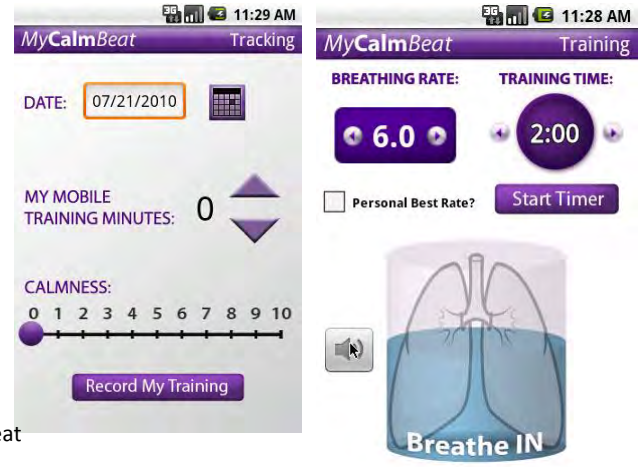


Figura 16. MyCalmBeat

Descripción: Es una aplicación móvil para Android que monitoriza la respiración del usuario. Muestra los dato de la tasa de respiración, y el tiempo en que se debe realizar.

Hallazgo positivo: Facilita guardar cada entrenamiento para ser continuado posteriormente. Está avalada científicamente.
Hallazgo negativo: No ofrece ningún extra, por lo que, tras un uso continuado, los usuarios pueden perder el interés.

8. Zen Relax

Desarrollador: Avant Web Solutions S.L.
Precio: 2,39€
Valoraciones: Sin valoraciones
Publicado: 19/04/10
Categoría: Estilo de vida
Tamaño: 11.7 Mb
Versión 1.0
Idioma: Español
Enlace:
http://www.avantwebsolutions.com/mobileapps/zenrelax.html

Figura 17. Zen Relax

Descripción: ZEN RELAX es una terapia de relajación guiada por la voz de un maestro Zen, que durante unos 20 minuots te transportará a un nivel de relajación absoluta. Tu mente recorrerá cada una de las partes de tu cuerpo.
La voz del maestro y el sonido de los elementos naturales que la acompañan, te ayudarán a reducir y eliminar tus tensiones, dolores o estrés.

Con Zen Relax descubrirás el poder de tu mente sobre tu propio cuerpo, llevándote a una dimensión hasta ahora desconocida.

Hallazgos Positivos: Antes de comenzar la escucha recomienda acomodarse y cerrar los ojos. La voz es cercana y agradable, natural, creíble.

Hallazgos Negativos: El audio de la voz no suena muy bien, no tiene buena calidad, se notan distorsiones y ecos, la acústica es bastante regular, es un audio como muy comprimido, sucio. Los sonidos ambientales de fondo no son controlables. Pueden hacerse molestos los trinares de los pájaros tropicales. Solo tiene una terapia en forma de audio que es esencialmente una terapia de relajación que pretende funcionar para eliminar tensión, dolor y estrés. Al terminar vuelve a comenzar.

9. iGuides – Eliminate Stress

Desarrollador: Artic Gerbil Creations
Precio: 5,49€
Valoraciones: Sin valoraciones
Publicado: 04/05/09
Categoría: Educación
Tamaño: 0.4 Mb
Versión 1.0
Idioma: Inglés

Figura 18. iGuides

Enlace: http://www.techrepublic.com/software/iguides-eliminate-stress-10-mobile/1352319

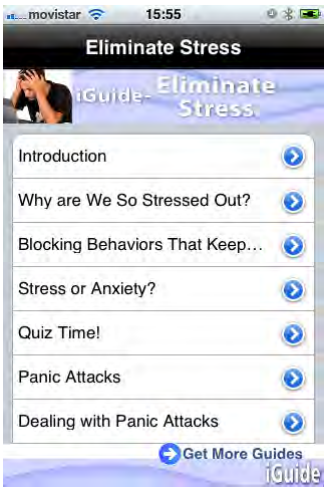


Figura 19. iGuide

Descripción: Pareciera que lo escuchamos de cada persona que conocemos, todos dicen "¡estoy tan estresado!" Se trata de la presión que nos rodea en el mundo de hoy. Esas presiones nos causan estrés y ansiedad.
Uno de cada 8 norte-americanos entre los 18 y los 54 años sufren de desorden de ansiedad. Eso es un total de más de 19 millones de personas.

Características:

- Introducciones que ayudan a aprender
- Visualización por capítulos para encontrar rápidamente
- Instrucciones paso a paso fáciles de seguir
- Paso entre capítulos con botones de anterior y próximo

Diferencias

- Precio bajo
- Vínculos fáciles de acceso directo para todas las iGuides
- Fuente de información más completa.

Hallazgos Positivos: Información muy completa.

Hallazgos Negativos: Solo texto. Podría estar descrito antes de comprar que solo se trata de texto, una especie de libro o memoria sobre el estrés.

1.6.1 Conclusiones del Estado del Arte

Revisadas dichas aplicaciones, y analizando los aspectos positivos y negativos de cada una de ellas, llegamos a la conclusión de que no existe o no encontramos ninguna aplicación que ofrezca en conjunto lo que pretende ofrecer ID-Stress. Es decir, tratamientos de audio avalados por una especialista en el área de la sofrología; cuestionarios para determinar el estrés; aplicaciones interactivas para realizarse antes, después u otro momento; alarma para recordar cuando debe hacerse un tratamiento; reproducción de audio que pueda pausarse en el momento deseado; ofrecer información acerca de cada ítem de la aplicación; un sistema que muestre el resultado de la interacción con los tratamientos de esta forma mantenemos el interés del usuario.

Estos aspectos, son los que consideramos factores claves de éxito para la aplicación y que hacen distinguirla del resto de aplicaciones en el mercado.

1.7 Perspectiva General del Proyecto

Capítulo I – Introducción

Se establece la descripción del problema y la solución al mismo así como también el estado del arte y la perspectiva general del proyecto.

Capítulo II – Fundamentos Teóricos

En este subcapítulo se explica la metodología de trabajo de proyectos SCRUM, la técnica de propuesta de ideas grupal, los perfiles del equipo de desarrollo del proyecto multimedia, los prototipos y flujo de datos que se aplican en la implementación. Así como lo esencial para la compilación y ejecución de la aplicación en el ordenador, las herramientas necesarias para crear las interfaces de la aplicación, el sistema operativo móvil de Apple y los lenguajes de programación utilizados para el desarrollo de la aplicación.

Se expone, además las características, tipologías de jugadores-juegos involucradas en la realización del mini juego de la aplicación

Capítulo III – Desarrollo de la Aplicación

Se explica la forma en que se trabajó con Cocos2D, la inclusión del código en el IDE Xcode y se hace una introducción a las librerías empleadas en el desarrollo de la aplicación para el sistema de partículas, para el mini juego y los elementos de Fuego, Aire, Tierra y Agua

Capítulo IV – Conclusiones y líneas de futuro

Comenta las conclusiones y cita las posibles líneas de futuro

Capítulo V – Glosario y Referencias

Se explican palabras técnicas que se pueden encontrar en esta memoria y se muestran los enlaces a las referencias empleadas para el desarrollo de esta memoria

CAPITULO II
FUNDAMENTOS TEÓRICOS

2. Fundamentos teóricos

2.1 Metodología de Trabajo

Para la realización de cualquier proyecto es indispensable que exista una metodología de trabajo que pueda garantizar la culminación exitosa del mismo, y para este trabajo final se empleó una metodología particular que se explicará a continuación.

2.1.1 SCRUM

Antes de comenzar a desarrollar el proyecto, era necesario organizar la forma de trabajo, establecer los plazos de entregas, y coordinar el tiempo que cada integrante del equipo debería dedicarle a los aspecto de desarrollo.

Así pues, se empleó el sistema SCRUM, *que es un proceso en el que se aplica de manera regular un conjunto de mejores prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos* [10].

El sistema de SCRUM ayuda a que trabajen todos los integrantes del equipo de forma conjunta en la misma dirección ya que se definen los objetivos al empezar el proceso, cerciorándose antes de que éstos sean claros y entendidos por todos. Durante el proceso de SCRUM se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto ya que se puede medir de forma clara el avance de las tareas realizadas y las que aún quedan por realizar.

Este proceso, se emplea cuando un proyecto es muy complejo, cuando se necesita obtener resultados de forma muy rápida y los requisitos sean muy heterogéneos y cambiantes durante el transcurso de desarrollo.

Figura 20 SCRUM Sprint Board

Proceso

En SCRUM un proyecto se ejecuta en segmentos temporales cortos y fijos -denominados Springs- usualmente no exceden un mes y en muchas ocasiones se utilizan dos semanas para cada iteración. Cada iteración, tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite. [10]

Este proceso parte de una lista de objetivos priorizados –denominado product backlog- que actúa como plan del proyecto. De esta forma, el cliente puede regular los objetivos para maximizar la utilidad de lo que se desarrolla y el retorno de inversión mediante la replanificación de objetivos al inicio de cada iteración. [10]

En este proyecto, se realizaron Springs de entre 1 y 2 semanas, armonizando el tiempo de desarrollo del proyecto con el necesario para el resto de tareas de la vida cotidiana, por lo que se podían cumplir los plazos planteados para la realización de cada objetivo.

Participantes

Para la realización del SCRUM, se emplean diversos roles que serán asumidos por los integrantes del equipo, los cuales deberán ser multidisciplinares, es decir, cada uno de ellos poseerá un perfil determinado para realizar una tarea específica. Se divide de la siguiente manera:

- **Product Owner o Producer:** Es la persona que tiene la autoridad suficiente para tomar decisiones de peso dentro del proyecto. Representa a todas las personas involucradas con el cliente y se encarga de definir los objetivos, calendarios y planificar el trabajo dentro del equipo.
- **Scrum Master:** Es la persona que asegura el cumplimiento de la metodología de trabajo, sirve de guía y asesoramiento al equipo ante cualquier problema que pueda surgir.
- **Scrum Team**: Corresponde al resto del equipo. Deben ser estables durante el tiempo de desarrollo del proyecto, tener perfiles multidisciplinares e independientes del exterior. Son los responsables de implementar las funcionalidades establecidas por el Producto Owner o Producer.
- **Clientes**: Son los beneficiarios finales del producto. Estos pueden ver el progreso del proyecto y aportar ideas, sugerencias, cambios y necesidades.

Acciones

- **Product Backlog:** Corresponde a una lista de tareas, requerimientos o funcionalidades que debe realizar el equipo.
- **Sprint Backlog:** Es el agrupamiento de tareas del product backlog que se deberá cumplir en un tiempo estipulado por cada miembro del equipo. No se deberán mover tareas asignadas durante un spring, si alguna de ellas no es
- **Daily Scrum Meeting: Es una** tarea iterativa que se debería realizar todos los días con un máximo de 30 minutos. Se trata de una reunión informal y ágil donde se hacen las siguientes preguntas a cada integrante del equipo:

1. *¿Qué tareas ha realizado desde la última reunión (**que he hecho**)?.*
2. *¿Sobre qué va a trabajar en el día actual (**que voy a hacer hoy**)?.*
3. *Identificación de obstáculos o riesgos que impiden o pueden impedir el normal avance (**que ayuda necesito**). El Scrum Master, debe eliminar aquí cualquier obstáculo que encuentre.* [11]

Algunas de las herramientas existentes para gestionar el proceso de SCRUM van desde post-its hasta software especializado. Cada grupo puede emplear la herramienta que mejor le convenga.

Figura 21 Spring Taskboard

2.1.2 El Equipo

A continuación, se ofrece una descripción de los perfiles que han participado en el proyecto iD-Stress y que cumplen con los parámetros SCRUM.

EMILIANO MARTINEZ RIVERA – Diseñador

Actualmente dedicado al servicio de Diseño Gráfico por cuenta propia, ha vivido la evolución de la comunicación gráfica tal y como la conocemos hoy desde los últimos 20 años. Formado como Diseñador Gráfico en el Instituto de Diseño de Caracas y con estudios de Diseño Industrial y Mecánica, comenzó su especialización en Interactividad y Experiencia del Usuario desde 1998, habiendo adquirido experiencia en diferentes empresas y agencias de publicidad de Caracas, Miami y Barcelona.

Su labor en el proyecto consistió en la creación de la imagen del producto, el diseño de las interfaces gráficas de la aplicación y el testeo de usabilidad de la misma con diferentes usuarios.

LEOPOLD RIOLA – Técnico

Estudiante de Ingeniería Técnica de Gestión en la Universidad Rovira i Virgili de Tarragona. Se encuentra realizando de forma individual una aplicación para iPhone para jugadores de póker profesionales, profesión con la que se está costeando los estudios.

Se encargó de estructurar la aplicación, la creación de la interfaces, programación de la alarma, de redes sociales, de la reproducción del audio y el resto de los aspectos funcionales de la aplicación.

SOFIA SWIDAROWICZ ANDRADE – Técnico

Graduada como licenciada en Informática en la Universidad de Oriente en Venezuela en el año 2008, ha trabajado en diferentes proyectos de software en la empresa Sigma Dental Venezuela ocupando el cargo de analista programador Jr. y analista de Software Multimedia y en 2010 ocupó paralelamente el puesto de líder de proyectos de software en la fundación
Operación Sonrisa Venezuela (Operation Smile).

Mi trabajo consistió en el desarrollo del mini juego y el desarrollo de los sistemas de partículas representadas por los cuatro elementos dentro de la aplicación.

MARTA MIGUEL REIZ – Producer

Licenciada en Comunicación Audiovisual por la Universidad Complutense de Madrid. Posee amplios conocimientos de grabación de vídeo y audio, así como de edición y postproducción. Actualmente compatibiliza los estudios con su trabajo como Agente de Ventas en Orange España. Con una extensa experiencia profesional en el sector comercial y ventas, muestra una clara orientación al cliente y a la consecución de objetivos.

Fungió como líder organizacional, se comunicaba con los clientes, fijaba fechas de entregas, planeaba el aspecto de marketing y gestión de proyecto.

2.2 Brainstorming

También llamada lluvia de ideas, es una herramienta de trabajo grupal que facilita el surgimiento de nuevas ideas sobre un tema o problema determinado en un ambiente relajado. [12].

Puesto que nuestro proyecto final de Master consistiría en realizar una aplicación móvil para un cliente real, Enlace, utilizando tratamientos de audio los cuales son inamovibles, es decir que siempre deben estar presentes por exigencia del cliente, el proceso de creatividad tuvo que ser dirigido hacia el momento anterior y posterior a los tratamientos para darle aún más valor a la aplicación móvil.

Para determinar como, y con que elementos debíamos trabajar para darle dicha forma y estructura a la aplicación, realizamos una serie de brainstormings o lluvia de ideas, como se aprecia en las figuras 20,21 y 22 durante un período comprendido de 3 semanas del mes de febrero. Así pues, en el primer brainstoming se planteó una posible arquitectura de la aplicación de forma global, junto con ideas que pudieran resultar interactivas:

- Los tratamientos auditivos deberían ser **Multidioma**, puesto que el cliente es franco-español y su target objetivo son principalmente sus clientes tanto franceses como españoles.
- **Visual**: concordamos que deberíamos usar animaciones en 2D, animación en 3D, videos y fotos.
- **Social**: que el usuario pudiera compartir con amigos, en redes sociales de internet que está empleando la aplicación.
- **Calendario**: que se pudieran configurar alarmas que le indicara al usuario cuando debe
- **Juegos**: caleidoscopio, mándalas, construcciones, pinturas, jardín.

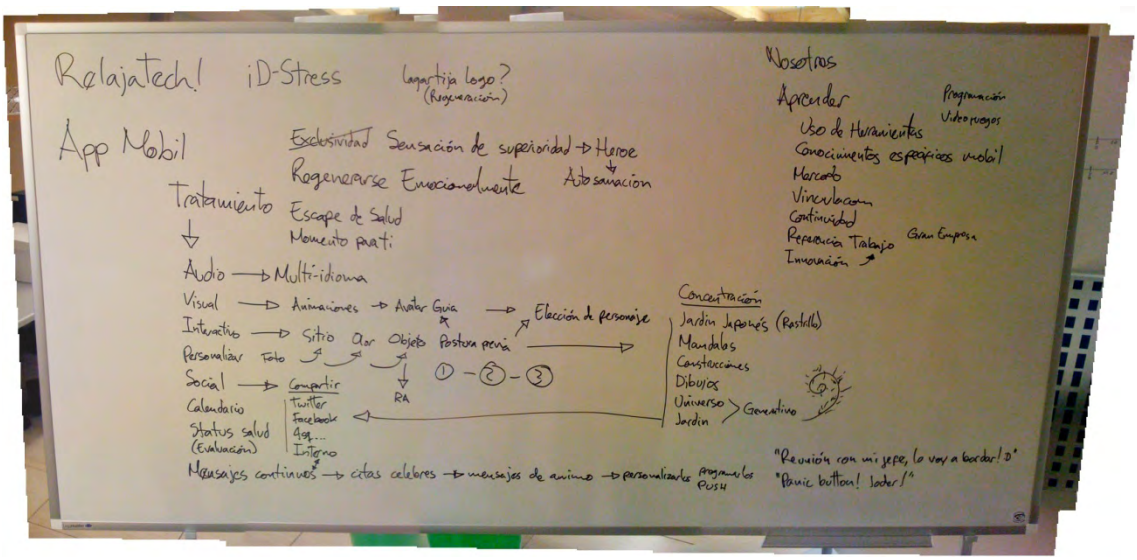


Figura 22. Brainstorming 1

Este primer brainstorming fue compartido posteriormente con el cliente, para que nos orientara con respecto a su idea inicial. Una vez se aclararon dudas y se comentaron aspectos sobre nuestras ideas, realizamos un segundo brainstorming considerando las sugerencias y cambios que el cliente nos ofreció.

En el nos enfocamos principalmente en las actividades interactivas para los tratamientos:

- Emplear efectos de luminiscencia en determinados momentos
- Mini Juego
- Consejos
- Fotos para cargar antes del tratamiento
- Animación explicativa sobre la forma de realizar correctamente un tratamiento

El segundo brainstorming se realizó para definir los interactivos de la aplicación. Se concluyó que deberíamos emplear una asociación con los elementos de la naturaleza para las actividades pre tratamiento, como un sistema de partículas de agua, tierra, fuego y aire. También que deberíamos realizar un minijuego para la reactivación después de cada tratamiento, en forma de sugerencia. Igualmente añadir un sistema simbólico que le indicara al usuario la forma en que ha interactuado con la aplicación, que sería algo como una proyección entre el resultado de sus tratamientos (evaluados por él mismo) y la sensación que estos le han producido.

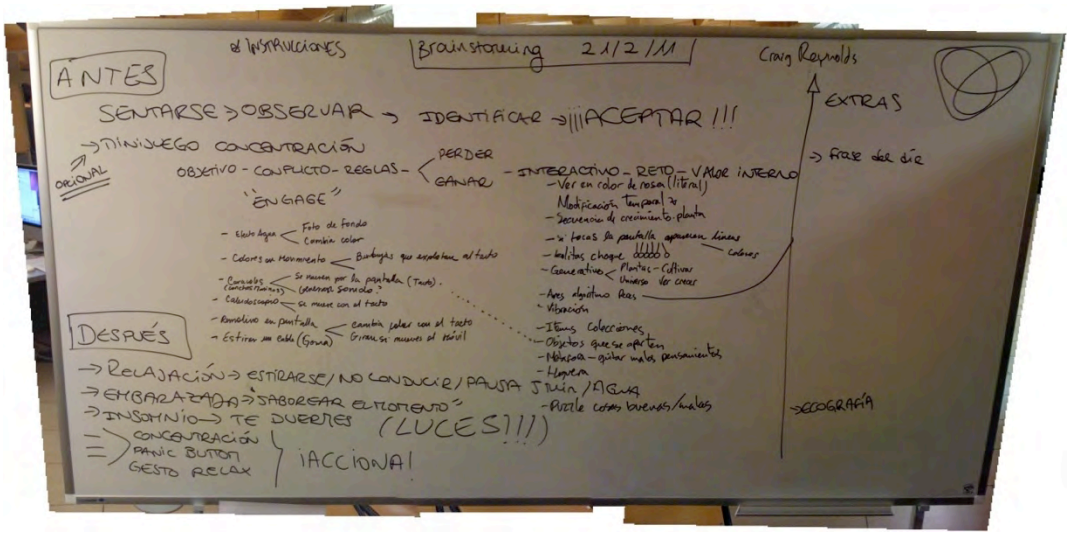


Figura 23. Brainstorming 2

Una vez definidos los tratamientos finales que irían en la aplicación y tomada la decisión de emplear los interactivos antes y después de cada tratamiento, se realizó un tercer brainstorming para determinar la posible estructura de la aplicación para cada ítem, de esta forma nos orientaríamos mejor al realizar los mockups [2.1].

Igualmente se realizó un brainstorming para determinar el tipo de juego que debería llevar la aplicación

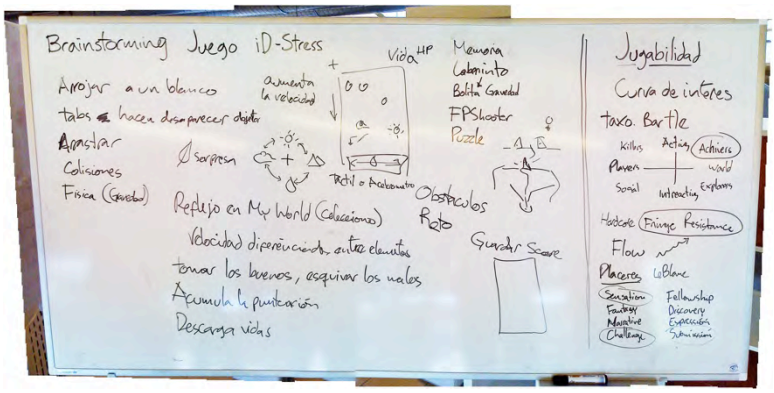


Figura 24. Brainstorming Mini Juego

Finalmente, la estructura de la aplicación debería conformarse de la siguiente forma:

- **Alarmas**: Con la posibilidad de configurar alarmas para recordarle al usuario escuchar el tratamiento que el decida antes de un evento, una exposición o una reunión importante que requiera concentración, relajación o disminución del estrés.
- **Cuatro Elementos**: interactivo que se propone como preparativo para la realización de un tratamiento, y que está basado en los cuatro elementos de la naturaleza, agua, fuego, aire, tierra.
- **Mini Juego**: Que servirá para reactivar al usuario una vez haya terminado un tratamiento, pues requerirá volver a un estado de atención para continuar con sus labores diarias. Será un juego breve, y sencillo.
- **Avatares Guía**: Son personajes que se encontrarán en determinadas áreas del aplicativo y servirán de guía, informándole al usuario cualquier dato o dándole respuesta a preguntas sobre la aplicación.
- **Cuestionario:** Medirá el nivel de estrés en el que se encuentra el usuario. Podrá realizarlo cada vez que quiera y determinar su mejora.
- **Mundo Evolutivo**: En esta sección, los usuarios podrán ver su nivel inicial de estrés (determinado por el cuestionario) así como su evolución dentro de la aplicación, relacionándolo con la valoración que éste haga de cada tratamiento realizado (Ej.: Hago un tratamiento de relajación y lo valoro de manera positiva, esto hará crecer Mi Mundo).

Figura 25. Brainstorming 3

2.2.1 Diagrama de Flujo de Datos

Es un diagrama que muestra la interacción entre el sistema de información y las entidades externas. Es empleado para diseñar los procedimientos o sentencias con coherencia lógica y que representan la solución al problema planteado en forma gráfica.

Para este proyecto se realizó un diagrama de flujo de los datos que deberían interactuar en la aplicación, de tal forma se podría visualizar un panorama general sobre los distintos procesos que intervendrían en el sistema y que luego servirían de apoyo al momento de realizar los Mockups.

Los rombos representan las acciones que podrán ejecutarse en la aplicación: Terapias, Cuestionario, Opc Configurables, Elementos y cada uno de estos realizará una acción determinada. Cuando haya acabado dirigirá al menú principal. Los círculos representa el proceso de extracción del área de almacenamiento interna de la aplicación.

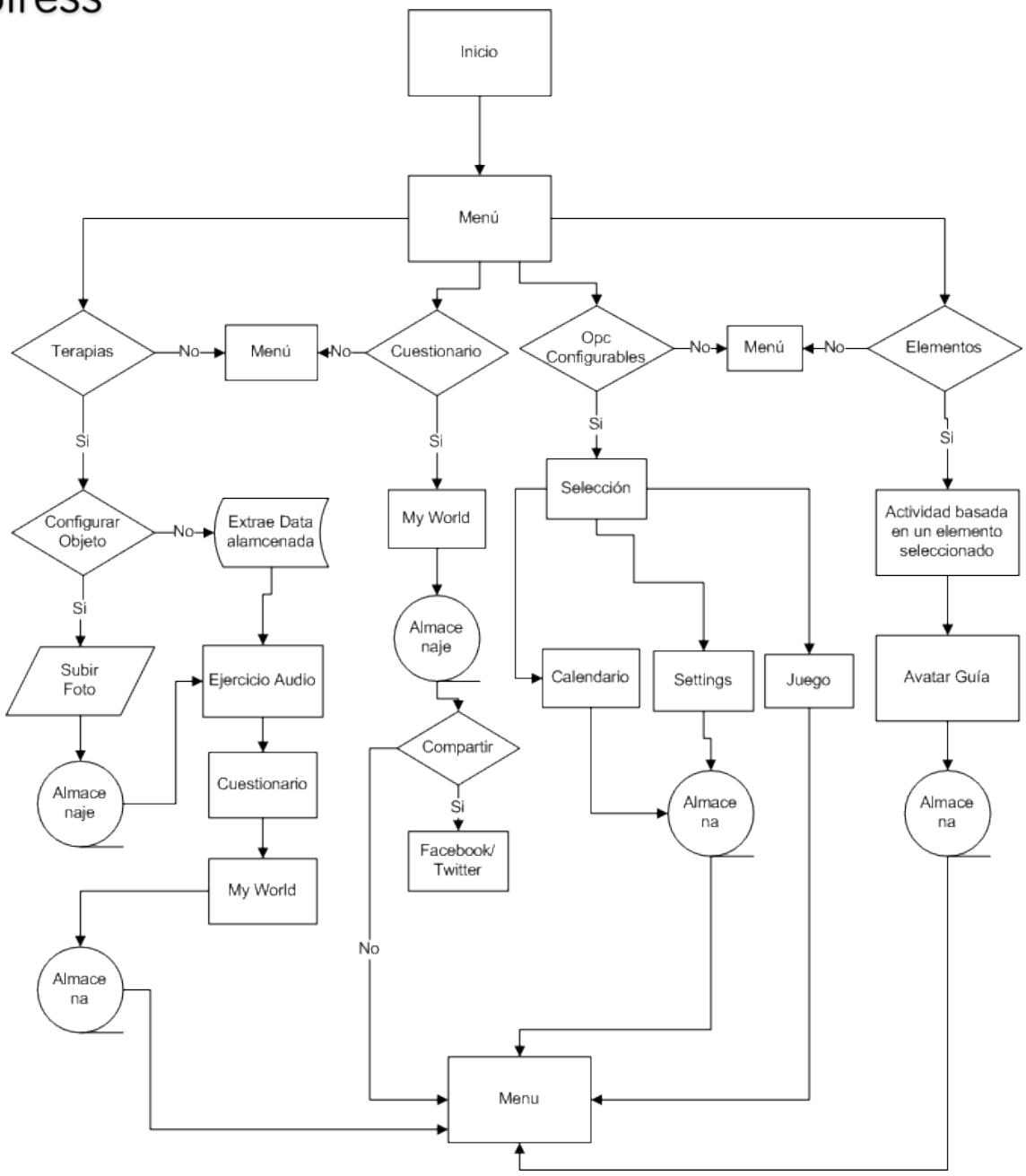


Figura 26. Diagrama de Flujo de Datos

2.2.2 Mockups

Los muckups son la representación gráfica de las posibles interfaces de la aplicación. Se realizan para determinar si el flujo de datos es el correcto o si hay algo que modificar y/o añadir en los diferentes procesos. Se pueden emplear diversas herramientas, desde software especializado hasta papel y lápiz. Lo importante es que se representen todos los aspectos concebidos y los resultados de cada acción.

Ya que la empresa colaboradora tomo la decisión de utilizar el sistema iOS [2.2] de Apple, es decir que el sistema se desarrollara en un iPhone, utilizamos la herramienta de mockup denominada "**Mockapp**" que genera plantillas con el formato de iPhone que pueden observarse en las figuras 26,27,28 y 29.

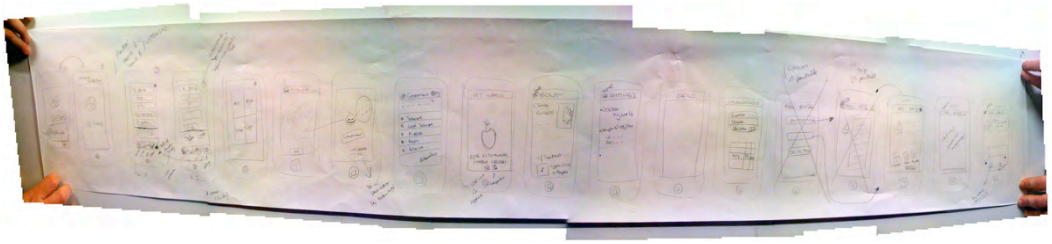


Figura 27. Mockup hecho con papel y lápiz.

Figura 28. Mockup de inicio versión 1

Figura 29. Mockup de Menú inicio versión 2

Figura 30. Mockup de Menú inicio Versión 3

Figura 31. Mockup de Menú inicio versión Final

Como se puede apreciar, los mockups [2.1] funcionan para advertir como puede quedar una aplicación antes de programarla, lo que resulta en ahorro de tiempo para los programadores o técnicos. Además, se pueden realizar pruebas con usuarios para determinar si la aplicación es entendida correctamente o no.

Así pues, partiendo de la idea planteada en los brainstormings, la aplicación final consistirá de 8 apartados básicos:

- Tratamientos: listados de los diversos tratamientos
- Panic Button: enlace directo al tratamiento de emergencia
- Cuestionario: preguntas para evaluar el nivel de estrés
- 4 elementos:
- Mini Juego: interactivo que se proporciona para luego de un tratamiento
- Evolución: como va el progreso del usuario en relación a su interacción con la aplicación.
- Alerta: una alarma de alerta que se configurará a gusto del usuario y se accionará en el momento indicado.
- Información sobre el centro Enlace: una especie de About Us sobre la empresa colaboradora.

Para determinar la forma en que éstos apartados deberían colocarse dentro de la aplicación, se realizaron tres tipos de mockups que fueron testeados con usuarios externos para determinar cual resultaba más fácil e intuitivo de emplear y que puede apreciarse en profundidad de detalles en la memoria de Emiliano Martínez [12]. Luego de un testeo preliminar se descarto uno de los modelos, reduciéndose así a dos tipos de mockups que

variaban en cuanto a acceso en los menú de inicio. El primer prototipo llevaba un menú inicio, el cual resulta atípico para las aplicaciones iPhone y otra sin dicho menú.

En la primera versión con **Menú de Inicio** se encuentran los siguientes apartados en la lista principal (la que se observa en medio de la pantalla):

Tratamientos, Cuestionario, 4 Elementos y Panic Button (uno de los tratamientos que, a petición del cliente, tenía que ser fácilmente accesible).
Así, quedan **Evolución**, **Alerta**, **Juego** y **Enlace** (una especie de "Sobre Nosotros") en la barra inferior o *Tab Bar* [2.3], junto al ítem Inicio, que devuelve a la pantalla con la lista principal.

Figura 32. Mockup versión 1

La segunda versión contiene de forma visible por defecto, en su *Tab Bar [2.3]*:

Cuestionario, Tratamientos, 4 elementos y Juego.
Los otros tres apartados se encuentran en la pestaña **More** (ver la figura en la parte inferior derecha), aunque esta distribución podrá ser reordenada por el usuario en todo momento.

Figura 33. Mockup versión 2

¿Qué ofrece cada apartado del mockup?

Cuestionario y Juego
Se inician con una ventana previa mediante el botón empezar. En esta sección aparte de ofrecer el cuestionario y el juego, se ofrece además, información básica acerca de cada actividad. Algo similar sucede en el apartado de los 4 elementos, con la diferencia que en éste el usuario puede escoger con cuál de los cuatro elementos quiere interactuar.

Tratamientos
El apartado de los tratamientos se inicia con una lista (llamada table view) de todos los tratamientos. En ésta aparecen el nombre, descripción, icono y duración de cada tratamiento. Una vez el usuario ha seleccionado alguno, se muestra una ventana parecida a la explicada anteriormente, con un botón empezar y una breve explicación del tratamiento.

Botones
Los botones "Empezar" hacen que se ejecute la función propia en cocos2D, el cual se comentará en el capítulo III Fundamentos Prácticos siguiente, excepto en el cuestionario que está realizado con *UIKit* [2.4].

Cuando la acción propia de cada apartado ha terminado, se muestra una ventana de cierre en la que el usuario puede: valorar el tratamiento en el caso de los tratamientos, leer los resultados y recibir sugerencias en el caso del cuestionario y recibir un mensaje de confirmación en los 4 elementos y el juego.

Alerta
El apartado Alerta está basado en la estructura de las alarmas de iPhone, mientras que los de Evolución y Enlace son ventanas individuales que contienen la información del apartado propio, sin botones.

Todos los apartados son *Navigation Views* [2.5], lo que significa que el usuario puede volver atrás en todo momento utilizando la barra superior de la pantalla. Ésta contiene además un botón de información o ayuda que sirve para orientar y dar información extra a los usuarios acerca de las operaciones que estén realizando.

Finalmente, después de la realización de los testing con diferentes usuarios decidimos emplear la versión uno (1) de los prototipos mencionados anteriormente como estructura final de la aplicación iD-Stress.

2.3 Herramientas, Lenguajes de Programación y Tecnología utilizadas

2.3.1 Herramientas

Una vez definido el sistema operativo en que la empresa colaboradora deseaba desarrollar la aplicación, y pensadas las posibles estructuras de flujo de datos elaboradas en el mockup, lo siguiente era comenzar a desarrollar y codificar la aplicación empleando las herramientas tecnológicas necesarias para dicho fin.

En éste caso se ha empleado el kit de desarrollo iPhone SDK, que es proporcionado por Apple, que provee de todos los recursos necesarios para programar una aplicación nativa para un dispositivo iPhone OS. Este kit ofrece todas las funcionalidades que tiene el dispositivo, como acelerómetro, interfaz Multi-Touch, sistema de sonido, sistema de video, etc.

2.3.1.1 Xcode

Es un entorno de desarrollo o IDE (siglas en inglés), para Mac OS X que es utilizado para el desarrollo de aplicaciones de dicho sistema operativo y para el sistema operativo móvil iOS, trabaja conjuntamente con Interface Builder y el iPhone Simulator que genera las interfaces gráficas de usuario y permite trabajar con proyectos escritos en C, C++ , Java y Applescript [13].

Características más relevantes:

- Crear y administrar proyectos, incluyendo plataforma, dependencias y requisitos del dispositivo.
- Escritura de código en un editor con opciones para la sintaxis e identación.

- Depuración del proyecto, bien sea en el simulador, en el sistema operativo o en el dispositivo móvil.
- Es el IDE oficial de Apple para programación de aplicaciones Mac OS X e iOS.

Gracias a estas características, este ha sido el IDE seleccionado para desarrollar nuestra aplicación para iPhone.

Figura 34. IDE Xcode

2.3.1.2 iPhone Simulator

El simulador de iPhone replica el sistema operativo de iPhone en el ordenador MAC y nos permite compilar y ejecutar nuestras aplicaciones sin necesidad de volcarla directamente en el iPhone [14].

Además, el simulador ofrece algunas características propias del dispositivo físico, como la simulación del modo panorámico (la ventana se rota a la izquierda o derecha para cambiar el tipo de vista de la aplicación), el bloqueo del dispositivo, el teclado en pantalla, entre otros. Sin embargo, no se puede confiar exclusivamente en una aplicación probada únicamente en el simulador, pues éste al ser un programa ejecutado en un sistema operativo de escritorio no posee las restricciones de procesador y memoria que tiene el dispositivo físico; a pesar de esta diferencia sí agiliza mucho la programación de aplicaciones, sobretodo cuando el proyecto se encuentra en el proceso de depuración, ya que no es necesario compilarlo y ejecutarlo en el iPhone.

La versión oficial del iPhone Simulator viene en el kit SDK de desarrollo para iPhone y ésta puede conseguirse en la página oficial de desarrolladores de Apple.

Figura 35. Simulador de iPhone

2.3.1.3 Interface Builder

Es el software para el desarrollo visual de la interfaz gráfica de las aplicaciones, tanto para el sistema Mac OSX como para iPhone OS. Esta herramienta ofrece una colección de objetos y componentes para crear la interfaz de la aplicación mediante "drag and drop" (arrastrar y soltar), como campos de texto, botones, tablas de datos, menús pop-up, etc. Además permite indicar la relación entre ellos y las acciones que deben ejecutarse cuando por ejemplo son presionados por el tacto.

Esta herramienta proporciona básicamente cuatro elementos principales:

- **Ficheros xib**: es un directorio que contiene todos los elementos y/u objetos (con sus respectivas características: tamaño, posición y conexión entre ellos) que aparecen en la interfaz gráfica de nuestra aplicación. Este fichero se almacena en un directorio localizado dentro de un proyecto Cocoa Touch.
- **Librería de objetos**: contiene los objetos y elementos que pueden ser añadidos a la interfaz gráfica. Podemos encontrar objetos típicos como botones, texto, ventanas, menús, controladores de objetos, vistas creadas por el programador, frameworks de objetos específicos... Una vez añades un objeto a la interfaz, Interface Builder instancia el objeto con la posibilidad de modificar los atributos del objeto (tamaño, posición etc). [13]
- I**nspector**: Para realizar las modificaciones de los objetos de la interfaz, luego de ser instanciados, se utiliza la ventana Inspector, que permite cambiar las características propias del objeto como: el tamaño o la posición, se pueden añadir animaciones y fundamentalmente asociar las acciones de los objetos.

- **Panel de conexiones**: En este panel se realizan las conexiones entre elementos de la interfaz como la indicación de los outlets, de la vista asociada, acciones que maneja, entre otros.

Interface Builder está integrado a Xcode y los cambios efectuados en las clases creadas por el desarrollador son visibles desde Interface Builder.

2.3.1.4 iPhone OS

También llamado iOS, corresponde a las iniciales en inglés de iPhone Operative System o Sistema Operativo iPhone, que es la plataforma empleada para desarrollar los sistemas iPhone. Fue creado originalmente para los dispositivos móviles iPhone pero luego se empleó como sistema operativo de los iPod Touch, iPads y Apple TV.

Este sistema operativo se divide en cuatro capas de abstracción [2.6]:

✓ **Capa Cocoa Touch:** provee el entorno de ejecución para las aplicaciones que funcionan en el sistema operativo iPhone OS. Se distinguen dos frameworks principales: *UIKit Framework* y *Foundation Framework*. Estos, reflejan la división de clases que poseen una interfaz gráfica (UIKit) de las que no la poseen (Foundation).

- **UIKit**: se encarga de proveer y controlar la interfaz del iPhone OS y define la estructura de comportamiento de la aplicación, incluyendo el manejo de eventos y la construcción de la interfaz. [13]
- **Foundation**: define la capa base de las clases de Objective-C, establece el comportamiento básico de los objetos, los mecanismos para su manejo, y provee objetos para tipos de datos primitivos, como cadenas, colecciones y servicios del sistema operativo. [13]

- ✓ **Capa Media:** Es la que proporciona y se encarga de los servicios multimedia que pueden ser empleados en el dispositivo móvil, como gráficos y sonidos.

  Se pueden distinguir las siguientes tecnologías:

  - **Gráficos**: En cuanto a gráficas se refiere, ofrece algunos frameworks para poder desarrollar tanto a alto como bajo nivel. Los más destacados son: **OpenGLES** (utilizado para la realización de la aplicación iD-Stress) que provee herramientas para trabajar en 2D y 3D y es el framework principal para la interfaz gráfica de la aplicación; y **EAGL**, que es el encargado de comunicar los objetos en la interfaz de la aplicación y de dibujarlos [13].
  - **Audio**: Permite añadir a las aplicaciones tratamiento nativo del sonido, así como la manipulación (reproducción, grabación, mezcla) del audio. Algunos de los frameworks incluidos en estas tecnologías son **AudioToolbox** y **CoreAudio** que proveen los tipos de archivos de sonido con los que trabaja la plataforma iPhone OS [13].
  - **Video**: A través del framework Media Player, iPhone OS provee la posibilidad de reproducción a pantalla completa de video, de archivos .mov, .mp4, .m4v y .3gp, además de soporte de códecs de compresión tanto de audio como de video. [13].

- ✓ **Capa de Servicios Principales (Core Service):** Provee una abstracción de los servicios principales que ofrece el sistema y que no están involucrados con la interfaz gráfica, tal como el manejo de strings, colecciones, procesos, threads (hilos), recursos, memoria e interacción con el sistema de archivos. Incluye los Carbon Manager, Core Foundation, Apple Events y OpenTransport. [15]

- ✓ **Núcleo del Sistema Operativo (Core OS):** Posee el núcleo del sistema, siendo éste el responsable de la administración de la memoria virtual, de los hilos, comunicación entre los procesos, archivos del sistema, drivers e interfaces básicas del sistema operativo. También posee las tecnologías de frameworks tales como: Accelerate Framework, External Accessoryy Framewok y Security Framework

2.3.2 Lenguajes de Programación

Un lenguaje de programación es el idioma diseñado para la comunicación entre la máquina y el ser humano. Son herramientas que nos permiten crear programas y software, éstos pueden ser de alto o bajo nivel.

2.3.2.1 Objective-C

Es un lenguaje de programación orientado a objetos, cuya característica fundamental consiste en su funcionamiento como un pequeño pero potente conjunto de extensiones para el lenguaje estándar C - comparte muchas características con ese lenguaje y por ello puede ser utilizado en conjunto con Objective C sin problemas-. Así, el desarrollador puede elegir en cualquier momento hacer una determinada tarea siguiendo las pautas de la programación orientada a objetos (definir una clase, por ejemplo), o mediante técnicas de programación

funcional (definir una estructura y varias funciones, en lugar de una clase). Fue utilizado como lenguaje principal por la compañía NeXT, fundada por Steve Jobs, y actualmente es el lenguaje de programación utilizado en Mac OS X y iOS [13].

Alguna de las características que se pueden encontrar en este lenguaje son:

- Las interfaces y la implementación deben hacerse en bloques de códigos separados. Tenemos entonces que la interfaz se coloca en un archivo cabecera (header) cuyo sufijo es .h mientras que la implementación (clase) se coloca en un archivo código con sufijo .m

Implementación:
En el código siguiente, se observa como se declara la implementación de la clase y los diferentes métodos declarados en el cuerpo.

```
@implementation nombreClase
+metododelaClase {
// implementación }
-instanciaMetodo {
 // implementación }
@end
```

Interfaz:
Se observa como solo se declaran los métodos ha desarrollarse en la implementación.

```
@interface pruebagame : CCLayer<AudioVisualizationProtocol>
{
   //declaración de variables y objetos
   CCMenu *backMenu;
   int score;
   int time;
}

@property (readwrite,retain) CCMenu *backMenu;

- (void)sleep; //llamada al método public sleep
- (void)wake;   // llamada al método púbico wake
+ (id) scene;    // llamada al método privado scene
```

- Se deben utilizar signos en la declaración de los métodos: positivo (+) se refiere a los métodos de una clase (privados) o métodos que puede ser llamados sin ser instanciados en una clase. Y negativo (-) cuando se denota la instanciación de los métodos que solo pueden ser llamados entre una instancia particular de una clase (públicos).
- Se distingue un archivo objective C de uno escrito en C por la extensión del fichero (.mm).
- Muchas decisiones que en otros lenguajes se toman en tiempo de compilación, Objective-C las deja para tiempo de ejecución.

- Las variables siempre se crean en memoria dinámica, evitando así el desbordamiento de pila.
- Se distingue el Objective C porque se hace una llamada a la librería o clase cabecera mediante la instrucción "#import", a diferencia del Lenguaje C (que es análogo a Objective C pues es un subconjunto de éste) que se escribe "#include", además de otras características semánticas particulares de las que se diferencian son en general muy similares, permitiendo su compatibilidad.

```
#import <stdio.h>

int main( int argc, const char *argv[] ) {
    NSLog( @"Hola Mundo\n" );
    return 0;
}
```

2.3.2.2 Lenguaje C

La base del C proviene del BCPL Basic Combined Programming Language (Lenguaje de Programación Básico Combinado), escrito por Martin Richards, y del B escrito por Ken Thompson en 1970 para el primer sistema UNIX en un DEC PDP-7 (un microcomputador). [16]

C proporciona varios tipos de datos que son directamente tratables por el hardware de la mayoría de las computadoras actuales y que ofrece, los típicos números en coma flotantes, números enteros, caracteres, arreglos, listas, punteros, registros y uniones, pero no proporciona el tratamiento de datos que no sean los básicos, sino que deben ser desarrollados por el programador, garantizando de este modo la eficiencia del lenguaje. Sin embargo estas carencias se solucionan con el empleo de librerías que normalmente dependen del sistema operativo. [16]

Su propósito general es el de ofrecer economía sintáctica, control de flujo, estructuras sencillas y un buen conjunto de operadores, por lo que no es un lenguaje de muy alto nivel y no está ligado a ningún sistema operativo ni a ninguna máquina concreta a pesar de que se le suele llamar lenguaje de programación de sistemas debido a su utilidad para escribir compiladores y a que fue desarrollado en conjunto con el sistema operativo UNIX [16].

En el ejemplo a continuación se puede apreciar la sintaxis del lenguaje:

```
#include <stdio.h>
main(){
  float cels, farh;

  fahr = 35.0;
  cels= 5.0 * (fahr -32.0)/9.0;
  printf("->%f F son %f C\n", farh, cels);
}
```

A través de los años han aparecido variantes de este lenguaje como:

- ✓ **C++** fue el segundo intento por difundir un lenguaje orientado a objetos y es la más difundida de estas variantes hoy día.
- ✓ **Objective C** aportó la programación orientada a objetos en C [16]

Otros lenguajes inspirados en C pero que no son compatibles con el:

- ✓ **Java**, inspirado en la sintaxis de C++ con orientación a objetos más parecido a la de Smalltalk y Objective C
- ✓ **JavaScript**, lenguaje inspirado en Java para ser utilizado en las páginas Web
- ✓ **C#,** es un derivado de C++ y Java desarrollado por Microsoft

2.3.3 Tecnologías

A parte de emplear el iPhone como sistema elemental de implementación para la aplicación iD-Stress, como un pedido de nuestra empresa colaboradora también se ha tomado en consideración emplear otras plataformas Apple como el iPod Touch y el iPad 2, por lo que serán analizadas también junto con el iPhone en el segmento presentado a continuación.

2.3.3.1 Aplicaciones Móviles

Las aplicaciones móviles constituyen la mejor forma de mostrar las prestaciones que una compañía puede ofrecer a sus clientes. De acuerdo a Emilio Avilés Santiago, especialista en el área de los servicios integrales de movilidad, "Si hace dos años todas las empresas querían tener presencia en Internet con una web, ahora todas quieren una aplicación. Hay mil millones de smartphones en el mundo y esta cifra se duplicará dentro de poco. Las aplicaciones ofrecen la oportunidad de estar presente en cualquier lado con una inversión relativamente pequeña; por menos de 10.000 euros una pequeña empresa puede tener su propia solución de movilidad" [17]

Las aplicaciones móviles pueden ser de dos tipos:

- *Web*: diseñadas para aprovechar las prestaciones del browser (GLOSARIO) del dispositivo móvil, adaptándose a los requisitos del mismo, por lo que no necesitan ser instaladas o compiladas en él. Utilizan XHTML, CSS y JavaScript y pueden proveer la sensación de una aplicación mientras son ejecutadas, no utilizan el sistema de refrescamiento del contenido como las páginas Web, permiten al usuario interactuar con el contenido en tiempo real y puede ser usados por varios dispositivos móviles. Por ejemplo, la aplicación móvil de Facebook o Twitter [18]

- *Nativas*: También denominadas "aplicaciones de plataforma", son aquellas que deben ser desarrolladas y compiladas para cada plataforma móvil. Por lo que se utilizan API's especificas para su desarrollo. Pueden funcionar bien online u offline y deben ser certificadas por las app stores [2.7] antes de ser puestas a la venta o para la descarga [19].

Una aplicación móvil de cualquiera de estas tipologías constituye una excelente plataforma de comunicación, ya que el costo de desarrollo es relativamente moderado y de un rápido retorno de inversión.

Figura 36. Ilustración aplicaciones iPhone

2.3.3.2 iPhone

El iPhone es un teléfono móvil con multifunciones, que proporciona acceso a servicios como Internet, correo electrónico, navegación Web, mapas, GPS, cámara de fotos, reproductor de audio, búsquedas. Posee una pantalla táctil y un teclado que se muestra en pantalla cuando es activado.

Lo que lo hace diferente del resto de smartphones es su interfaz impecable, su carcaza con bordes redondos, el sistema táctil "Multi-touch", la facilidad de sincronización con el ordenador y los sensores de movimiento e inclinación y la rigurosidad en que son analizadas las aplicaciones que son puestas en la tienda virtual de Apple que lo han convertido en el Smartphone más deseado del mercado.

Figura 37. Aplicaciones iphone

2.3.3.3 iPod Touch

El iPod Touch es un reproductor multimedia diseñado y distribuido por Apple, y es la tercera generación de reproductores multimedia iPod, fue introducido al mercado en septiembre de 2007.

Básicamente, el iPod empezó siendo un simple reproductor mp3, y ha ido evolucionando hasta tener las siguientes características [13]:

- Interfaz multi-táctil.
- Pantalla de 3.5 pulgadas con una resolución de 480x320 pixeles.
- Sistema operativo iPhone OS.
- Unidad flash para almacenamiento de 8, 32 y 64 Gigabytes de capacidad.
- Conexión Wi-Fi con compatibilidad para el estándar 802.11 b/g. y Bluetooth
- Procesador Apple ARM a 412 MHz (1G), Apple ARM a 632 MHz (2G), ARM Cortex-A8 a 800 MHz(3G).
- Sistema operativo para sincronización: Mac OS X 10.4.10 para Mac o Windows XP SP2/Vista/7 para PC, con iTunes 7.6 o posterior

El iPod Touch no dista mucho de ser un iPhone, ciertamente la diferencia más grande es que no proporciona un servicio de llamada pero sí permite llamadas VoIP a través de Skype, lo que lo convierte en un gran dispositivo multimedia.

2.3.3.4 iPad 2

El iPad es un dispositivo electrónico multimedia tipo Tablet PC desarrollado por Apple. Salió al mercado a principios del año 2010 y se ha convertido en otro de los productos claves de Apple INC.

El iPad funciona sobre NUI (Interfaz Natural de Usuario) en una versión adaptada del sistema operativo iOS. Posee las siguientes características:

- pantalla con retroiluminación LED y capacidades multitáctiles de 9,7 pulgadas (24,638 cm)
- Espacio en memoria flash 16 a 64 gigabytes (GB)
- Bluetooth integrado
- Un conector dock de 30 pines que permite la sincronización con el software iTunes, además de proporcionar conexión para diversos accesorios.
- Existen dos modelos: uno con conectividad a redes inalámbricas Wi-Fi 802.11n y otro con capacidades adicionales de GPS y soporte a redes 3G (puede conectarse a redes de telefonía celularHSDPA).

En el iPad pueden ejecutarse videos, audio, ebooks [2.8], visualización de fotografías, reproducción de audiolibros, editores de texto. Posee como navegador predeterminado Safari,

por lo que se puede navegar por internet, visualizar videos de youtube, tomar fotografías y editarlas, acceso a la App Store y iTunes Store, entre otra serie de prestaciones.

Para 2010 se presentó el iPad 2, más ligero que su predecesor, con un procesador más potente Apple A5 Dual Core chip de 900MHz que ofrece el doble de rendimiento y gráficos hasta 9 veces más rápidos que la versión anterior.

En este caso incluyeron:

- 2 cámaras, una en la parte frontal y otra en la parte trasera
- Un giroscopio y Acelerometro.
- Salida de video de Alta Definición (HD)

2.3.3.5 iMac

Para el desarrollo de la aplicación iD-Stress se empleó una iMac de 64 bits, pues los diversos frameworks y herramientas de desarrollo que proporciona Apple sólo funcionan bajo el sistema operativo Mac OS X

Este ordenador se caracteriza por tener el monitor y la CPU integrado en el mismo dispositivo. Anteriormente las iMacs funcionaban bajo la arquitectura PowerPC de IBM, pero hasta hace unos años adoptó la arquitectura Intel con procesadores de doble núcleo. Algunas características del modelo utilizado para el proyecto son:

- Procesador Intel Core 2 Duo a 2 Ghz, doble núcleo de 64 bits.
- Pantalla de 20 pulgadas.
- Sistema Operativo Mac OS 10.5, o también llamado Leopard.
- Permite la virtualización de otros sistemas operativos.

2.3.3.6 Apple App Store

Las tiendas virtuales de aplicaciones móviles constituyen el espacio donde los desarrolladores o empresas, colocan sus productos para ser vendidos o repartidos de forma gratuita por los usuarios.

Apple ofrece su propio servicio de tienda de aplicaciones, para iPhone, iPad, iPods y que permiten descargar cientos de aplicaciones. Sin embargo, subir aplicaciones se deben cumplir una serie de requisitos: ser desarrolladas con el iPhone SDK (Software Developer Kit) y cumplir con el Human Interface Guide (HIG).

Cuando vendes una aplicación en su tienda, Apple se queda con el 30% de la ganancia y el desarrollador o empresa dueña del producto con el 70% restante.

2.4 Framework para el desarrollo de Juegos

En esta sección se hará referencia a los distintos frameworks y engines gráficas que fueron analizados, y que se ajustaban al perfil de herramienta para el desarrollo de los interactivos tales como el Mini Juego y los sistemas de partículas. Con esta investigación se pudo

seleccionar la herramienta más apropiada para cumplir con los objetivos propuestos y que aprovechara las potencialidades interactivas que ofrece el iPhone.

2.4.1 Cocos2D for iPhone

Es un framework creado para diseñar y construir aplicaciones interactivas gráficas, demos y juegos 2D para iPhone y está basado en la librería Cocoa de Apple. Fue desarrollado por Ricardo Quesada basándose en el diseño Cocos2D de python para ofrecer una herramienta con las siguientes características [20]:

- **Fácil uso:** pues utiliza una API (Aplication Programming Interface) familiar (Xcode) y posee una gran cantidad de ejemplos que resultan de gran utilidad para principiantes.
- **Rápida:** pues emplea OpenGL ES (variante de la API OpenGL) que permite la generación y control de gráficos en dispositivos móviles y estructuras de datos optimizadas.
- **Flexible:** permite ser integrada fácilmente en librerías de terceros.
- **Gratis:** es una herramienta open source, compatible tanto con fuentes de juegos abiertos o cerrados.
- **Comunidad:** de gran tamaño, amistosa y activa que darán soporte a las dudas de programación que puedan surgir.
- **Aprobada por la AppStore:** el punto más importante de todos pues existen actualmente 2500 juegos en la AppStore que han sido desarrolladoss con Cocos2D for iPhone.

Entre las prestaciones que encontramos en Cocos2D tenemos:

- Transiciones de escenas
- Sprites (imágenes)
- Efectos (Ripple, Lens, Waves, etc)
- Sistema de Partículas
- Sistemas físicos (gravedad, fricción, rebotes)
- Renderizado de texto
- Soporte de Audio
- Soporte Touch/Acelerómetro – Teclado/Mouse

Puesto que utiliza como lenguaje de programación Objective C se pueden utilizar muchas librerías ya desarrolladas en este lenguaje o importar librerías implementadas en lenguaje C, ya que también es soportado por el IDE Xcode.

Figura 38. Logotipo Cocos2D for iPhone

Puntos Positivos:

- Tiene muchos ejemplos publicados en la red.
- Existen muchos tutoriales y libros sobre Cocos2D y como crear juegos para iPhone.
- Posee una comunidad muy activa y dispuesta a colaborar.

- Es Open Source y gratuita.
- La semántica es sencilla.

Puntos Negativos:

- Hay que codificar todos los elementos de interacción: botones, hipervínculos, animaciones, efectos, etc.

2.4.2 ShiVa 3D

Es un Game Engine [2.9] multiplataforma de desarrollo de juegos y aplicaciones 3D en tiempo real para los sistemas operativos Windows, Mac Os, Linux, iPhone, Android, Palm, Wii y iPad. Se caracteriza por poseer un editor 3D WYSIWYG (What You See Is What You Get) y un MMO (Massive Multiplayer Online) Server [21]

Figura 39. Editor gráfico ShiVa 3D

Presenta las siguientes características:

- Creación de sistema de partículas
- Estelas
- Animaciones
- Diseñar HUDs (Head Up Display) que es la información que se muestra en todo momento en la pantalla durante la partida bien sea en forma de íconos y números.
- Diseña materiales, terrenos, océanos, etc.
- Script AIs (Artificial Inteligence) codificados en Lua
- Librerías de sonido

- Sistemas de sombreado
- Engines Físicos

ShiVa está basado en una versión optimizada de Lua (lenguaje de programación) empleado en la actualidad para la creación de video juegos como World of Warcraft, permitiendo crear comportamientos complejos. Es compatible con C, C++ y Objective-C.

Puntos Positivos

- Desarrollo de juegos en 3D
- Modelado de personajes, entorno, etc
- Librerías de sonido muy amplia
- Empleado en plataforma iOS
- Es un entorno de edición de video juegos muy completo

Puntos Negativos

- No es un framework gratuito, la licencia cuesta 500€ aunque existe una versión para estudiantes, sin embargo no permite la publicación de los desarrollos, ya que es sólo para propósitos académicos.
- La curva de aprendizaje es muy grande, hay que modelar objetos en 3D y programar las animaciones y el engine del juego.
- Existe poca documentación, ejemplos y tutoriales disponibles en la red.

2.4.3 iProcessing

Es un framework open source cuyo propósito principal es facilitar el desarrollo de aplicaciones nativas que requieran el empleo de imágenes, animaciones e interacción para iPhone empleando lenguaje Processing. Específicamente es una integración de la librería Processing.js y un framework de aplicaciones en Javascript [22]

Figura 40. iProcessing para iPhone

Este framework se encuentra en una versión beta, por lo que no garantizan la inclusión en la App Store de las aplicaciones desarrolladas en este framework, por ende lo hemos descartado sin necesidad de analizar mucho más allá sus beneficios.

2.4.4 Unity3

Es un editor o motor gráfico para desarrollar juegos de video 3D u otro contenido interactivo como visualizaciones arquitectónicas o animaciones reales en 3D. Unity es similar a

Director, Blender, Virtools, Torque Game Builder o Gamestudio en el sentido de que proporciona como método principal de desarrollo un ambiente gráfico integrado.

Funciona para los sistemas operativos Windows, Max OS X, para los sistemas móviles Android y iOS para las plataformas Nintendo Wii, XBOX 360, PS3 (Play Station 3), y para los sistemas

Web, por lo que se pueden desarrollar increíble contenido multimedia para dichas plataformas.

Unity 3 es la nueva versión de este motor gráfico y posee las siguientes características:

- Renderizado optimizado para ser veloz y mantener la calidad
- Iluminación precisa sobre el ambiente dentro de tu juego
- Terrenos, lo que correspondería a backgrounds para la ambientación del juego
- Librerías Físicas
- Librerías y controles de audio
- Utiliza la plataforma de programación denominada Mono. Basado en el modelo .net de Microsoft.

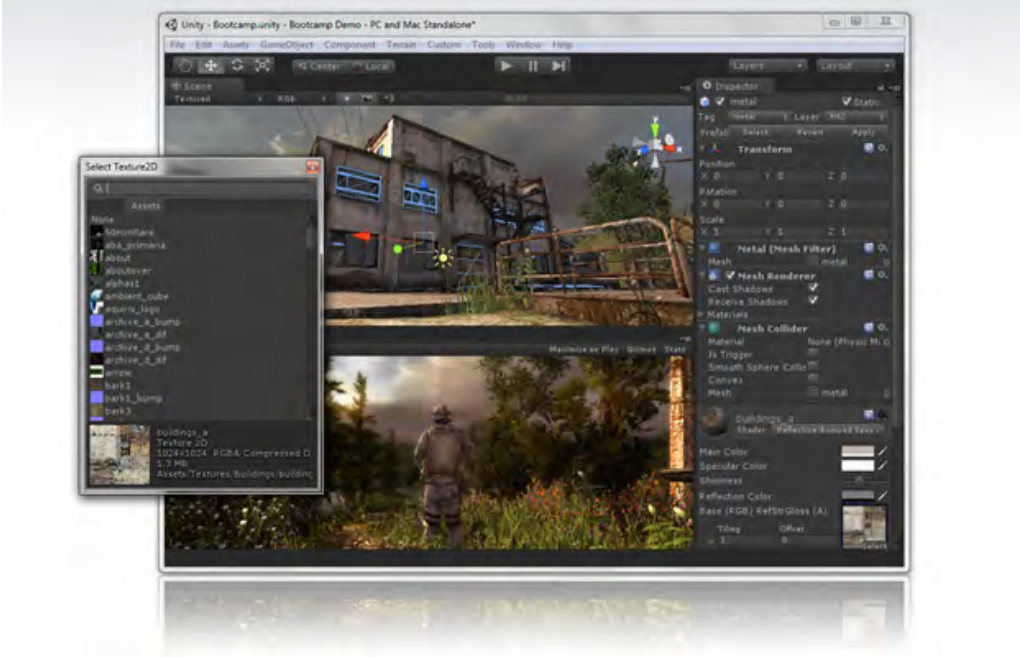


Figura 41. Entorno gráfico de Unity 3

Puntos Negativos

- Requiere de mucho tiempo para modelar los objetos, sujetos, exteriores.
- La curva de aprendizaje sería muy elevada para el tiempo que se dispone.
- A pesar de que ofrecen ejemplos básicos y sencillos para aprender los conceptos primordiales, no se consiguen tutoriales para aquellos que no han experimentado antes con un engine gráfico como es mi caso.

Puntos Positivos

- Es una herramienta de desarrollo gráfico de video juegos y es gratuito
- Ofrece una gran librería de modelos, audio, y gráficos.
- Puede ser desarrollado para casi cualquier plataforma en el mercado.
- Proporciona una amplia documentación
- Existen juegos creados por Unity que están a la venta en la App Store de Apple.

2.4.5 Adobe Air

Es un entorno de ejecución multiplataforma para la construcción de aplicaciones RIA (*Rich Internet Applications*) utilizando Adobe Flash, Adobe Flex, HTML y AJAX, las cuales pueden usarse como aplicaciones de escritorio. [23]

AIR fue creado como un entorno de ejecución versátil que permite usar código Flash, Actionscript, HTML o JavaScript para crear aplicaciones para internet con muchas características de los programas tradicionales de escritorio. Adobe lo define como un entorno de ejecución que no necesita navegador, para poder portar RIAs (aplicaciones de internet enriquecidas) al escritorio; más que como un framework de aplicaciones corriente.

Se consideró esta herramienta como posible candidata para el desarrollo de nuestra aplicación y sobretodo para los interactivos y mini juego, ya que se ejecuta como aplicación web y no como nativa en el móvil utilizando tecnología Flash, que es una plataforma que no necesitaría mayor curva de aprendizaje pues ya se tiene nociones sobre ella. Sin embargo fue descartada ya que Apple a finales del año 2010 ya no ofrecía soporte a las aplicaciones desarrolladas con flash, rechazando cualquier integración en su App Store.

2.4.6 Conclusiones de los Frameworks Analizados

Luego de analizar cada herramienta, de instalar algunos frameworks y probar su funcionalidades con ejemplos y algunos tutoriales, llegamos a la conclusión de que la herramienta que ofrecía los mayores puntos positivos y beneficios para desarrollar una aplicación como ID-Stress, que debe contener actividades interactivas, audio, gráficos e imágenes, era Cocos2D.

Su comunidad está muy extendida y se encuentran en la red ejemplos y ejercicios que facilitarían el aprendizaje de la herramienta en un período de tiempo más reducido que el resto de herramientas, además no hay que diseñar ni estructurar modelos en 3D, sino que se pueden emplear hojas de estilos llamadas "Sprite Sheets" que consiguen ser diseñadas fácilmente en cualquier editor de imágenes y gráficos en dos dimensiones.

Considerando también que son los tratamientos de audio lo fundamental en la aplicación y tanto el mini juego como el sistema de partículas deberían ser sólo una forma de darle a iD-Stress la sensación lúdica no hay necesidad de hacer algo extremadamente llamativo ni complejo.

2.5 Game Design

Como se ha comentado en el capítulo II de fundamentos teóricos, para esta aplicación se realizará un juego que formará parte de los extras de la aplicación y que será recomendado al final del tratamiento como forma de reactivación. Es decir, cada vez que un usuario realice un tratamiento que lo relaje, podrá, si así lo desea reactivar su estado de atención mediante la utilización del mini juego.

Tipo Jugadores

Antes de desarrollar el mini juego, y tener ideas de qué deberíamos crear era indispensable conocer al tipo del jugador al que iba dirigido, de esta forma garantizamos que el público objetivo de nuestra aplicación se sienta atraída e identificada con el mismo:

- **Demografía**

  De acuerdo a Jesse Schell, diseñador de juegos, y profesor en la Universidad Carnegie Mellon y autor del libro "The Art of Game Design" [24] plantea que no todas las personas son iguales, pero cuando creamos algo que está destinado a un numeroso grupo de personas es indispensable agruparlos de tal forma que podamos considerarlos iguales. Esta forma de agrupación se denomina demografía o en términos de marketing ***segmentación de mercado.*** [24]

  Puesto que nuestro target objetivo son personas de entre 25 y 56 años, entre hombres y mujeres, tal como se explicó en el apartado 1.4 Descripción del Problema, podemos entonces considerar lo siguiente:

  > **25-35**: Veinte y treinta. A esta edad, el tiempo comienza a ser más preciado. Es la edad en que las responsabilidades de la adultez comienzan a ser más evidentes, la mayoría de los adultos en esta edad son **jugadores casuales**, juegan para entretención ocasional, o juegan con sus hijos pequeños. Por otro lado también se encuentran los jugadores "hardcore" —aquellos que consideran su hobby principal jugar— y son un target importante en el mercado puesto que compran muchos juegos, y suelen expresar lo que no les gusta, lo que hace, y son los que influencian las potenciales compras en su red social.
  >
  > **35**-**50**: Treinta y Cincuenta. A veces se refiere a esta etapa como a "maduración familiar", la mayoría de los adultos en este bloque están muy enfocados en su carrera y responsabilidades familiares por lo que son **jugadores casuales**. A medida que sus hijos se hacen mayores, los adultos en este segmento son los que toman las decisiones en las compras de juegos, y cuando es posible, buscan juegos que involucren la participación de toda la familia. [24]

Tal como se explica en el párrafo anterior, nuestros jugadores serán en su mayoría considerados "jugadores ocasionales". Además si tomamos en cuenta que nuestro juego no debe ser el atractivo principal de iD-Stress, sino sólo un añadido que aporte riqueza a la aplicación, resulta apropiado realizar un juego para esta clase de jugadores.

➢ **Taxonomía de los Jugadores según Bartle**

También hay que considerar, a parte de la demografía que son factores externos y físicos, los factores psicológicos que definen qué actividad le gusta más a una persona y a otra cuando juega. Bartle segmentó en cuatro categorías [24]:

♦ **Achievers**: les gusta lograr objetivos y cumplir las metas del juego. Su placer principal es el reto.

♠ **Explorers**: quieren conocer la amplitud del juego. Su placer primario es el Descubrimiento.
♥ **Socializers**: Están interesados en las relaciones con otras personas. Ellos buscan principalmente placeres que involucren a una comunidad.
♣ **Killers**: Están interesados en competir y derrotar a los otros jugadores. Poseen una mezcla de placeres, entre la competición y la destrucción.

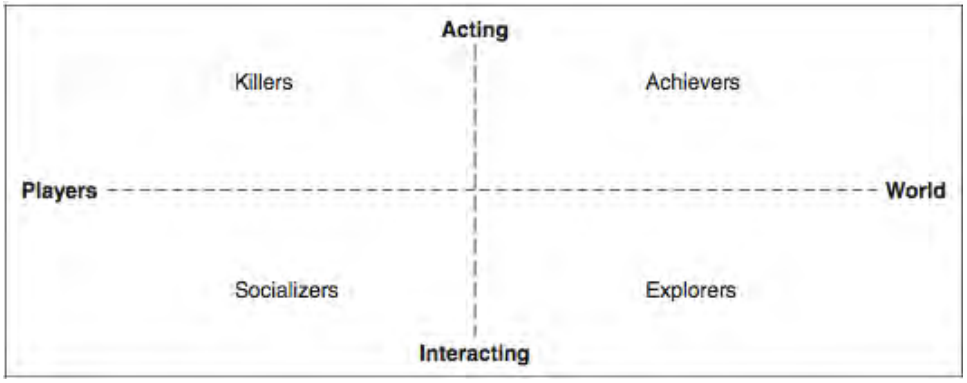


Figura 42 Taxonomia de los Jugadores

Conociendo de antemano estas características en los jugadores, analizamos la clasificación de los placeres en los juegos según Marc Leblanc [25]

- **Sensación**: Cualquier cosa que implique la emoción de experimentar con los sentidos, tiene que ver mucho con la estética del juego. **Fantasía**: Es el placer que ofrecen los mundos imaginarios y sumergirse en ellos.
- **Narrativa**: El placer que involucra experimentar el desarrollo de los acontecimientos.
- **Reto**: Tiene que ver con el placer de resolver problemas en un juego.
- **Comunidad**: Involucra la amistad, cooperación y comunidad dentro del juego.
- **Descubrimiento**: Buscar y encontrar algo nuevo.
- **Expresión**: Expresarse y crear cosas (juegos que permiten diseñar los personajes, etc.).
- **Sumisión**: Aquellos que se dejan ser arrastrados por las normas y las experiencias del juego.

De este modo, teniendo las características del jugador así como la de los placeres, realizamos la debida selección:

- Tomando como referencia la clasificación de Bartle, el perfil de nuestros jugadores serán Achievers, les gusta lograr objetivos y cumplir las metas del juego, pues se adapta al tipo de jugador ocasional que será nuestro objetivo principal.
- Y considerando la clasificación de Marc Leblanc, debería orientarse hacia el Reto, es decir que el juego ofrezca problemas a ser resueltos coincidiendo con la clasificación de Bartle.

Así pues, ya obtenidas dichas características decidimos realizar el brainstorming para nuestro juego en el que obtuvimos lo siguiente:

Qué tipo de juego podría ser:

- **Puzzle**: Algún rompecabeza, crucigramas o acertijos lógicos.
- **Laberinto**: ofrecer calles y encrucijadas en el que se debe descubrir el camino correcto a la meta.
- **Memoria:** Utilizar la agilidad mental y el recuerdo para descubrir fichas o cartas que sean iguales de un set determinado.
- **Shooter:** Juegos donde se debe disparar a los enemigos, ya sean objetos o individuos, por ejemplo Angry Birds [2.1.1] o Space Invaders [2.1.2].
- **Tetris:** Objetos que caen de la parte superior de la pantalla y que deben coincidir, bien sea por color, forma, etc, con otros objetos que ya se encuentren en la parte inferior o suelo del juego.

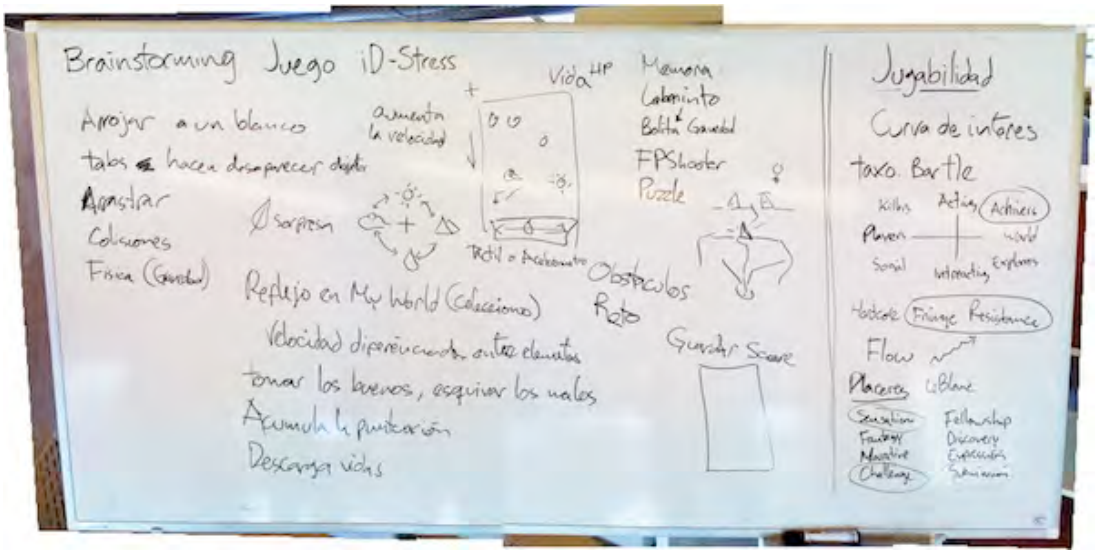


Figura 43. Brainstorming del mini Juego

Al mismo tiempo se consideró aprovechar las funcionalidades del iPhone:

- Colisiones entre objetos
- Arrastrar objetos
- Hacer desaparecer objetos
- Arrojar a un blanco
- Obstáculos en el camino

- Sonido al desaparecer los objetos
- Acelerómetro para mover los objetos
- Temporizador de tiempo
- Contador de puntos

Finalmente, y teniendo todos los aspectos mencionados anteriormente presentes, decidimos implementar el concepto de un Tetris [2.1.3] (aunque aplicando cambios para hacerlo más propio), ya que a es sencillo de entender, suelen jugarlo personas que

quieren pasar sólo un corto periodo de tiempo jugando y además se ajusta a los parámetros de nuestro tipo de jugador y de los placeres que se han descrito anteriormente. De este modo, realizamos otro brainstorming para determinar como debía comportarse el mini juego.

Posibles mecanismos o reglas del juego

- Velocidad diferenciada entre los diferentes elementos que caen de la parte superior.
- Esquivar objetos que podrían ser "malos" ya que restarían puntos o vidas y tomar los "buenos" para acumular puntos.
- Acumular puntos como objetivo principal del juego.

Una vez definidos estos aspectos, se prosigue a explicar como se conforma nuestro juego de acuerdo a las especificaciones de Jesse Schell:

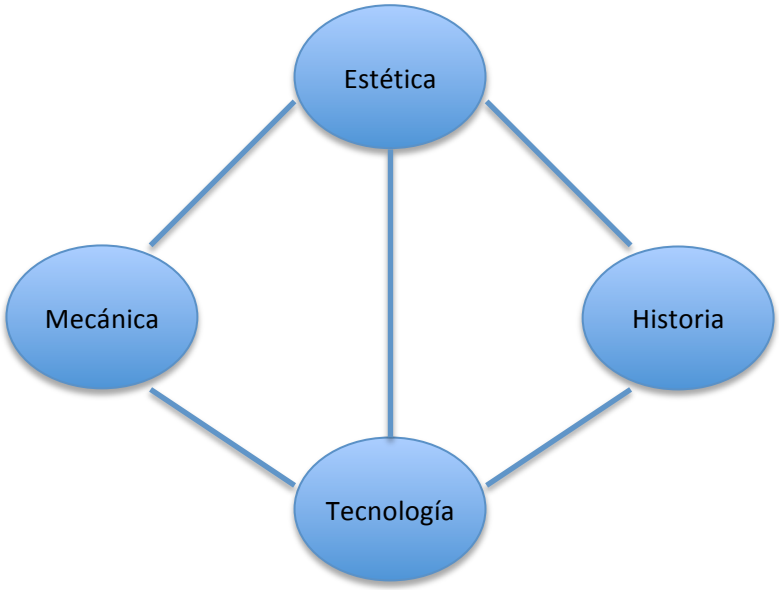


Figura 44. Elementos de un juego según Jesse Schell

Reglas: son todos los procedimientos y reglas del juego. La mecánica describe el objetivo, como los jugadores pueden o no alcanzar el objetivo, y que pasa cuando lo intentan.

En nuestro caso, la mecánica del juego es bastante sencilla aunque esto no significa que el juego sea simple. Como se comentó anteriormente, tomamos como referencia el juego Tetris, con la gran diferencia de que los objetos no tienen que hacer parejas para ser eliminados, sino que éstos caerán desde la parte superior de forma aleatoria y sólo un tipo de ellos deberá ser eliminado en un tiempo determinado.

El jugador ganará si logra eliminar un número determinado de objetos específicos, definidos al iniciar el juego, en dicho tiempo, sino deberá repetir la operación hasta lograr el objetivo.

Historia: es la secuencia de eventos que se desarrollan en el juego. Puede ser lineal o tener un guión; puede tener ramificaciones o ser emergente. La historia depende de la estética y la tecnología ya que fortalecen la historia.

La historia de nuestro juego, es lineal, hay que recolectar objetos para vencer a la máquina en un tiempo determinado. La acción no se sitúa en un ambiente específico, ni posee un personaje definido.

Estética: es como suena, se ve, se siente el juego y es la parte fundamental del mismo ya que es el elemento que entabla directa relación con el jugador.

La estética del juego coincidirá con el espíritu de la aplicación en general, ofreciendo calidez y frescura. Para este momento, las gráficas no están completamente terminadas, sin embargo, se muestra la primera versión. Cada objeto poseerá una imagen con colores brillantes entre verde, amarillo, rosa y azul, debiendo eliminar un determinado color cada vez.

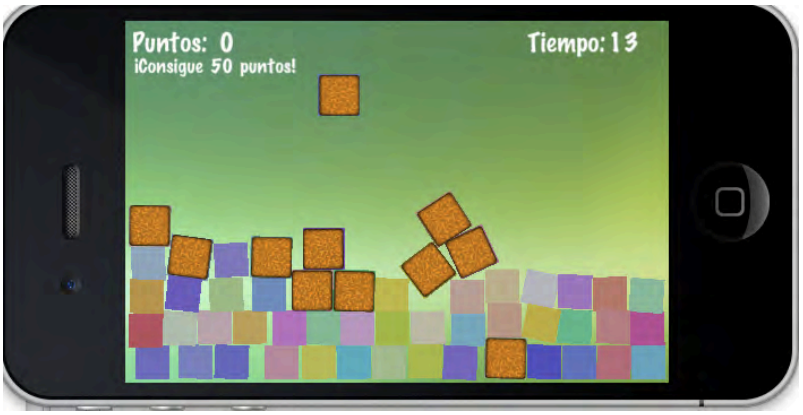


Figura 45. Estética principal de la aplicación

Tecnología: son todos los elementos e interacciones que harán posible el juego. Desde papel y lápiz, figurines de plástico a poderosos rayos láseres virtuales.

Este punto ha sido clave, ya que los objetos tendrán gravedad y rebotarán al ponerse en contacto con otros, ofreciendo elasticidad y fricción, esta característica le da un toque de

innovación al juego. Se logrará gracias al empleo del framework Cocos2D [2.1.4], orientado a la creación de juegos 2D para iPhone y será comentado técnicamente en el siguiente capítulo Desarrollo de la Aplicación de esta memoria.

También hemos tomado en consideración el balance que explica Jesse Schell [24] que se dividen de la siguiente manera.

- **Fairness**: tiene que ver con la equidad entre jugadores en el juego. Existen los juegos simétricos y los asimétricos, y hay que saber cual de las dos características se aplicará en el juego. En nuestro caso los jugadores deben tener cierta agilidad para hacer desaparecer los objetos con el tacto y con rapidez, lo que podría suponer una ventaja (no muy grande) a las personas que tienen un poco más de experiencia con esta clase de juego que a los jugadores noveles.

- **Challenge vs. Success**: Tiene que ver con el flow del jugador y el juego, si éste es muy retador (challenge) el jugador se frustrará, pero si el jugador gana muy rápido y frecuentemente se aburrirá. Este punto ha sido crítico, porque en determinados casos consideramos muy fácil de ganar, por lo que el temporizador es el punto clave del sistema, si se hace en un tiempo muy corto, no dará tiempo suficiente a eliminar objetos, pero si es muy largo será exactamente lo opuesto.
- **Meaningful Choices** : tiene que ver con las decisiones que toma el jugador dentro del juego y sobretodo decisiones significativas que le permitan hacerse preguntas sobre como debe abordar el juego o como ganar. ¿Cómo podrían ganar nuestros jugadores utilizando alguna estrategia?. Los objetos caen de forma aleatoria, por lo que podría suceder que el jugador decidiera eliminar no los objetos que caen, sin los que ya están asentados en la parte inferior, por ejemplo, aumentando su puntuación.

- **Skill vs. Chance** : ofrecer muchas oportunidades al jugador inhibe los efectos de las habilidades del jugador y viceversa. En el caso de nuestro juego las rapidez podría interpretarse como una habilidad, eliminar los objetos rápidamente, por ejemplo podría ser una habilidad que no todos los jugadores poseen, sin embargo también se ofrecen oportunidades que favorecen su puntuación, en determinados momentos se aumentará al doble la puntuación si elimina un objeto comodín que aparecerá en un momento determinado.

- **Head vs. Hands** : ¿qué tanto se emplean las acciones físicas del jugador (apretar botones) y qué tanto necesita pensar para lograr un objetivo?. En este caso, utilizar las manos es lo que tiene mayor peso, ciertamente hay que tener habilidad al tocar la pantalla, pero el propósito inicial para el concepto del juego ha sido ese, que el jugador no tenga que pensar demasiado en lo que debe hacer o en lo que está haciendo, más que dejarse llevar. El juego tiene un reto claro, pero ganar dependerá más de la habilidad con las manos que de la intuición y/o la lógica.

- **Short vs. Long** : Si el juego es muy largo el jugador podría aburrirse, y si es muy corto podría no tener la oportunidad de desarrollar y ejecutar estrategias significativas. En nuestro caso el juego es considerado como corto, de 1 minuto

máximo aproximadamente, que es el tiempo que ofrecemos en el temporizador, puesto que como objetivo principal comentado con anterioridad es que este se desarrolle rápidamente. Sin embargo la estrategia que el jugador podría desarrollar es la rapidez con que elimina los objetos.

- **Rewards :** Las recompensas son un factor fundamental en el sistema de los juegos, los jugadores juegan para obtener recompensas y es la forma en que se le indica al jugador que lo ha hecho bien. Existen diferentes formas de recompensas: A través de alabanzas, por la puntuación, juegos prolongados sin que agotes los recursos, hacerse más poderos. En nuestro mini juego, la recompensa consiste en acumular puntos hasta alcanzar la victoria y además se ofrecen algunos objetos que aumentan la puntuación obtenida a favor del jugador en determinado momento del juego.

- **Punishment**: Jesse Schell comenta que también debe existir un equilibrio entre los premios y los castigos dentro del juego, no hay que ser excesivos con ninguno porque puede causar aburrimiento en el primer caso y frustración en el segundo. Sin embargo, al igual que las recompensas, los castigos son parte importante de la experiencia del juego y lo hacen más excitante. Algunos de los castigos que se suelen aplicar pueden ser la pérdida de puntos, la disminución del tiempo, perder vidas, retorno al punto de inicio cuando se elimina a tu personaje, etc.
  En nuestro juego consideramos varios castigos, pero se decidió la pérdida de puntuación cuando se selecciona un objeto determinado para ello, para darle más equilibrio al juego y que no sólo existan recompensas.

- **Simple vs. Complex** : ambos conceptos parecen paradójicos. Llamar a un juego "simple" puede verse como crítica: "es tan simple que aburre" o como un halago: "simple y elegante" y la complejidad puede ser: "Muy complejo y confuso" a "Intrínsecamente complejo y muy rico". Para determinar qué tan complejo o simple es el juego existen las siguientes características:

  - Complejidad innata: Cuando las reglas del juego se tornan complejas, pueden ser difíciles de dominar pero a ciertas personas les gusta controlar y aprender esas series de reglas.
  - Complejidad emergente: Tienen reglas sencillas pero generan una complejidad a medida que se desarrolla el juego, éstas son las más preciadas por los jugadores.

  En nuestro juego, las reglas fueron creadas para se fáciles de aprender, sin embargo no consideramos que el juego sea simple (que aburre), pues como se explica anteriormente, existen recompensas y castigos, se debe emplear la habilidad de la velocidad y al mismo tiempo se debe pensar en alguna estrategia para eliminar objetos que sea favorables mientras se esquivan los castigos. Por lo que la complejidad emergente es lo que caracteriza al juego.

2.6 Los Cuatro Elementos

A parte de desarrollar un mini juego que funcionara como reactivador del estado de alerta del usuario al finalizar un tratamiento, también se presentaba la opción de realizar otros interactivos que actuaran de forma opuesta al mini juego, es decir que prepararan al usuario para iniciar un tratamiento en el que su estado de concentración y relajación debieran estar muy elevados.

Esto nos hizo pensar en los cuatro elementos de la naturaleza -agua, fuego, aire y tierra-, los cuales eran considerados en la antigua Grecia como los ladrillos de la construcción del

Universo, y parte de una estructura básica propia [26] y de cómo, a pesar de que hoy día nos encontramos rodeados de edificaciones, seguimos queriendo estar en contacto con la naturaleza. Tranquiliza caminar por la playa o el campo, y escuchar sus sonidos relajan el estado anímico de un individuo o le permiten conciliar el sueño.

Partiendo de este concepto, decidimos desarrollar entonces interactivos que simularan estos cuatro elementos, que pudieran realizarse ya sea como preparación a un tratamiento o simplemente para interactuar con ellos de forma casual.

Para desarrollarlos fue necesario emplear al igual que con el mini juego el framework Cocos2D, puesto que ofrece librerías de partículas, y librerías físicas que se adaptarían perfectamente al concepto que queremos transmitir con cada elemento. De este modo definimos los diferentes elementos y cuales deberían ser sus características:

Figura 46. Elemento de Fuego

Figura 47. Elemento de Agua

- En el caso del fuego, el usuario podrá interactuar con la llama de forma táctil.
- En el caso del agua podrá generar ondas e interactuar con ellas.

- En el caso del aire utilizando la vibraciones auditivas captadas por el micrófono se generaran partículas simulando hojas que vuelan.
- En el caso de la tierra se generarán partículas que mediante el acelerómetro del iPhone o el tacto permitirá al usuario su interacción con ellas.

Figura 48. Elemento de Aire.

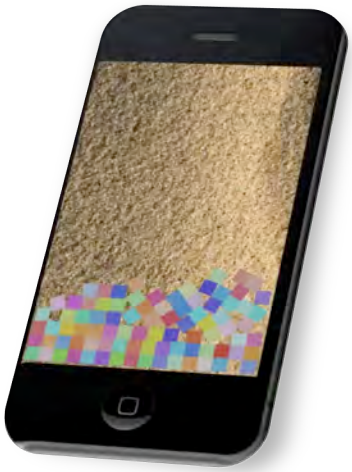


Figura 49. Elemento de Tierra.

Todo los aspectos técnicos de dichos elementos serán explicados en el capítulo III Desarrollo de la Aplicación.

2.5 Human Interface Guidelines (HIG)

Uno de los requisitos de la empresa colaboradora consistía en la publicación de la aplicación en la App Store de Apple [2.7], por lo que es necesario que ésta cubra los requisitos mínimos establecidos por dicho servicio, y que son obligatorios.

Entre uno de los más importantes se encuentra el Human Interface Guideline (HIG) que describe los principios estéticos y de interacción hombre-máquina que debe poseer cualquier aplicación para iPhone que quiera ser publicada en su App Store. Por ejemplo, el siguiente cuadro muestra el tamaño en píxeles que debe poseer una aplicación para que se ajuste al tamaño de la pantalla.

Sistema	Portrait	Landscape
iPhone 4	640 x 960 pixels	960 x 640 pixels

iPad	768 x 1024 pixels	1024 x 768 pixels
Other iPhone and iPod touch devices	320 x 480 pixels	480 x 320 pixels

Figura 50. Tamaño de la pantalla de los dispositivos de Apple

Todas sus aplicaciones deben tener consistencia, feedback, control del usuario sobre la aplicación, integridad estética, y requisitos programáticos que han de cumplirse al pie de la letra.

Por lo que hemos de tener cuidado y estar muy pendientes sobre qué elementos incorporar y cuales dejar a un lado.

CAPÍTULO III

DESARROLLO DE LA APLICACIÓN

3. Desarrollo de la Aplicación

Este capítulo constituye la sección más importante de la memoria. Se puede dividir en dos partes, la que explica con detalles como se estructuró el sistema con los elementos que componen la aplicación nativa, y la sección interactiva que consiste en el juego y los cuatro elementos; ésta parte será la que se explicará de forma más extensa, profunda y detallada a continuación.

3.1 Integración COCOS2D

Como se ha explicado anteriormente, para el desarrollo de estos interactivos era necesario utilizar una herramienta que facilitara la implementación de un sistemas de partículas y de otros recursos, como librerías físicas, para el desarrollo de un mini juego. Al seleccionar el framework Cocos2D [2.1.4] para ello, debimos aprender la forma de utilizarlo, puesto que era la primera vez que trabajamos con esta herramienta.

Antes de explicar como se realiza la integración de Cocos2D, es necesario entender primero como funciona esta librería. En primer lugar, ésta se divide en diferentes áreas de trabajo, secciones y componentes que hacen posible su funcionamiento.

- **Escena:** una escena es una pieza independiente del flujo de trabajo (workflow en inglés) de la aplicación, se implementa con el objeto **CCScene**. Algunas personas le llaman "Pantalla" o "Escenario". Una aplicación puede contener tantas escenas como se requieran, pero sólo una de ellas estará activa en un momento determinado.

  Por ejemplo, se podría tener un juego con las siguientes escenas: Intro, Menu, Level 1, Cutscene 1, Level 2, Winning cutscene, losing cutscene, High scores screen [24]. Cada una de estas escenas se pueden definir como aplicaciones separadas; pero entre ellas existe un componente lógico que las conecta. La escena de intro lleva al menú cuando es interrumpido o ha acabado el juego, el Level 1 puede llevarte a cutscene 1 si terminó ese nivel o a Losing cutscene si ha perdido, etc. Ver figura 51.

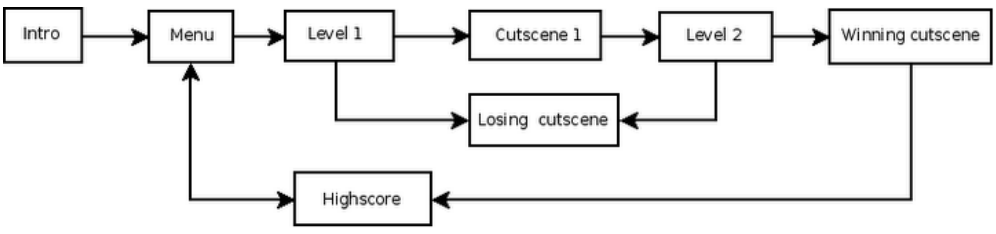


Figura 51. Esquema del flujo lógico de Cocos2D

Existen también subclases de **CCScene** como **CCTransitionSceneobject** empleado para realizar transiciones entre escenas (aparecer/desaparecer, aparecer deslizando de un lado, etc).

La implementación de la escena en nuestro programa se puede observar en el código siguiente:

```
//Método privado para instanciar la escena principal de la clase

+(id) scene
{
//'scene' es un objeto del tipo autorelease
CCScene *scene = [CCScene node];

//'layer' es un objeto del tipo autorelease
EfectoAire *layer = [EfectoAire node];
layer.isTouchEnabled=YES;

//se añade "layer" como hija de la escena
[scene addChild: layer];

//regresa el objeto de tipo escena
return scene;
```

Puesto que las **escenas** son subclases de CCNode (la clase base para renderizar todos los objetos en Cocos2D, como las escenas –CCScene-, las capas –CCLayer-, los sprites-CCSprites- y menú-CCMenu-), éstas pueden ser transformadas manualmente o mediante el empleo de **acciones**.

- **Acciones**: Las acciones son ordenes que recibe cualquier objeto CCNode. Estas acciones usualmente modifican los atributos de los objetos, como su posición en pantalla, rotación, escala, etc.

```
#Mueve un sprite 50 pixeles a la derecha, y 10 pixeles al top por 2 segundos.
[sprite runAction: [CCMoveBy actionWithDuration:2 position:ccp(50,10)]];
```

- **Director:** La clase CCDirector crea y controla la ventana principal y gestiona como y cuando ejecutar las escenas. CCDirector es el que en realidad cambia el objeto CCScene luego que el objeto CCLayer ha solicitado que reemplace, comience o termine la escena actual.
  En el código siguiente se puede observar como CCDirector toma el control para detener la animación cuando el background de la aplicación aparece.

```
-(void)
applicationDidEnterBackground:(UIApplication*)application {
        [[CCDirector sharedDirector] stopAnimation];
 }
```

- **Layer**: Son las capas o subconjunto de capas dentro de una escena y corresponde a todo el área dibujable, que define la apariencia y comportamiento de la aplicación, y es donde se deben desarrollar los métodos. Se realiza mediante la utilización de la clase CCLayer, sublclase de CCNode, ésta puede ser semitransparente (puede poseer hoyos y/o transparencia parcial en algunos lugares) permitiéndole "ver" otras capas ubicadas detrás de la principal.

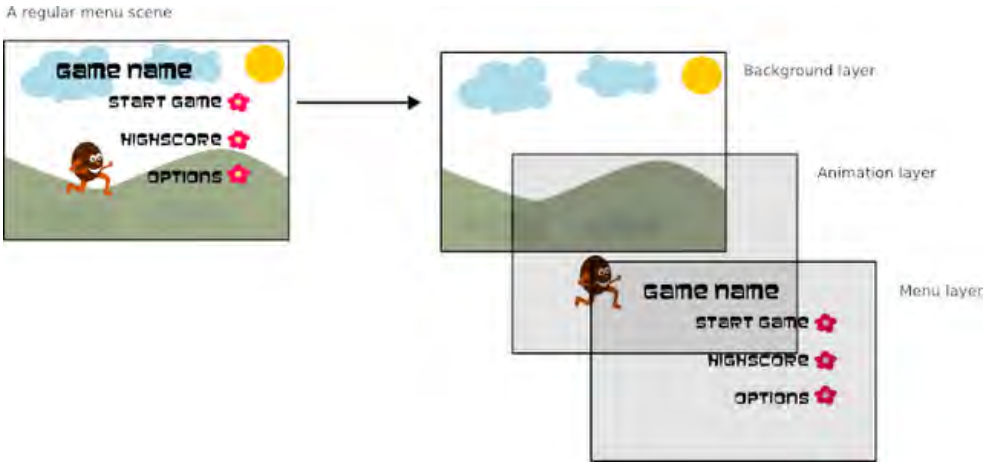


Figura 52. Diversas capas (Layers) dentro de una escena

Otros elementos que se encuentran en Cocos2D son:

1. **Sprites**: Imágenes 2D empleadas en la aplicación.

2. **Eventos**: Acciones que se ejecutan dentro de la aplicación y le indican qué hacer en todo momento.

3. **Flow Control**: Controla el flujo de control entre las diferentes escenas.

Teniendo claro el funcionamiento de la estructura lógica de Cocos2D, se puede proceder a realizar la integración de dicho framework con el framework **UIkit** que provee las clases necesarias (como los objetos de la aplicación, manipulación de eventos, vistas de ventanas y controles diseñados específicamente para una interface táctil) para construir y controlar la interfaz de usuario del sistema iOS. Normalmente la estructura base se desarrolla en UIKit, pero dado que cocos2D es una plataforma pensada para ser la base de las aplicaciones que lo implementen, se ha tenido que ejecutar la aplicación como si ésta fuera la principal. Una vez ejecutada la aplicación, ésta esconde directamente la capa de cocos2D para mostrar la vista general, realizada en UIKit.

En el primer cuadro se observa la instanciación de las propiedades y eventos propios del UIKit así como el mensaje que se envía al sistema para ejecutar primero Cocos2D:

```
//Método que ejecuta el mensaje que inciará Cocos2D para luego darle
// paso  a la ventana UIKit

- (void)applicationDidFinishLaunching:(UIApplication *)application
   {
      //Envía el mensaje y verifica la notificación
      self.notifyCenter = [NSNotificationCenter defaultCenter];
      [notifyCenter addObserver:self selector:@selector(trackNotifications:)
                    name:nil object:nil];

      //Se instancia el objeto ventana principal
      UIWindow *window = [[UIWindow alloc]
                          initWithFrame:[[UIScreen mainScreen] bounds]];

      [window setUserInteractionEnabled:YES];
      [window setMultipleTouchEnabled:YES];
```

En el siguiente cuadro se observa la instanciación de **CCDirector** en el método para ejecutar la escena principal para luego cargar los controles UIView pertenecientes al UIKit.

```
    CCDirector *director = [CCDirector sharedDirector];
    EAGLView *glView = [EAGLView viewWithFrame:[window bounds]
                        pixelFormat:kEAGLColorFormatRGB565 depthFormat:0];

    [glView setMultipleTouchEnabled:YES];

      // Se adhiere openglView al director
      [director setOpenGLView:glView]
      // hace OpenGLView hijo de la ventana principal
      [window addSubview:glView];
      // hace la ventana principal visible
      [window makeKeyAndVisible];

      //se ejecuta la escena principal de Cocos2D
      [[CCDirector sharedDirector] runWithScene:gs];

      General *high= [[General alloc] init];
      self.general = high;
      [high release];

      // instancia los controles del UIView
      [self showUIViewController:general];
}
```

Cocos2D seguirá en segundo plano hasta que la aplicación mande un mensaje de tipo NSNotification al **appDelegate** (clase principal de toda aplicación) como se muestra en el código siguiente y éste haga aparecer una escena de cocos2D según el mensaje recibido. Esto se aplicará, de forma inversa, cuando se quiera volver a mostrar la vista general.

```
// Método que ejecuta la escena de Cocos2D especificada
- (void) trackNotifications: (NSNotification *) notification
{
      id nname = [notification name];
      //Ejecuta la escena del tratamiento Panic Button
      if([nname isEqual:@"empezarPanicButton"])
      {
        [self hideUIViewController:general];
        [[CCDirector       sharedDirector]       pushScene:       [CCTransitionMoveInB
transitionWithDuration:2.0f scene:[HelloWorld scene]]];
       }
}
```

3.2 Librerías

Las librerías son un conjunto de subprogramas que se desarrollaron para resolver un problema determinado en un sistema de software y que son compartidos en dichos sistemas.

3.2.1 Sistema de Partículas

Antes de explicar la programación del sistema de partículas, es necesario conocer los principios físicos que la rigen. Así pues, una partícula es un cuerpo cuyas dimensiones son despreciables pero posee una masa definida, una posición, velocidad y responde a las fuerzas [27].

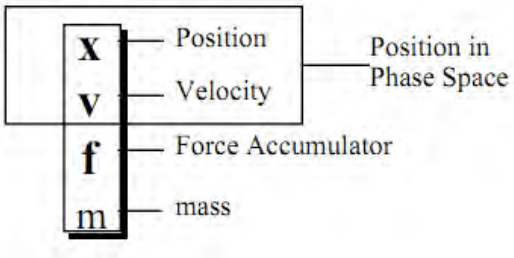


Figura 53. Elementos necesarios para la creación de partículas

El movimiento de un partícula Newtoniana está gobernada por la clásica fórmula: $\boldsymbol{F} = \boldsymbol{m}.\boldsymbol{a}$ lo que será escrito como $\ddot{\boldsymbol{x}} = \boldsymbol{f}/\boldsymbol{m}$. Esta ecuación resulta en una ecuación de segundo grado, pero es necesario llevarla a una de primer grado introduciendo variables extra para manejarla. Por lo que se introduce una variable **v** que representa a la velocidad, resultando en un par de ecuaciones de primer grado:

$$\ddot{\boldsymbol{x}} = \boldsymbol{f}/\boldsymbol{m}$$
$$\dot{v} = f/m \qquad \dot{x} = v$$

Pero también las partículas son influenciadas por la fuerza de gravedad: $\boldsymbol{F} = \boldsymbol{m}.\boldsymbol{g}$ donde la gravedad es un vector constante apuntando hacia abajo. Sin embargo en los sistemas

informáticos simular la gravedad como resulta en la realidad de 9.81 m/s^2 resulta muy lenta para nuestros sentidos, por lo que se suele utilizar un valor muy inferior.

También se considera el tiempo de inicio de la partícula para obtener el siguiente resultado:

$$t = t_o$$
$$x(t_o)$$
$$v(t_o)$$

Así se obtiene la ecuaciones siguientes que corresponden al motor de inferencia numérica o Solver del tipo Euler:

$$xn + 1 = xn + \Delta\, t.vn + 1$$
$$xn + 1 = vn + \Delta\, t.\mathcal{E}/m$$

Es de tipo explícito y su utilización ofrece pros:

- Es rápido
- Es simple porque se hace una evaluación de derivada por iteración

y contras:

- No es robusto como código
- No es muy estable ni preciso
- Depende de $\Delta\, t$

Aunque se terminó empleando las librerías de partículas de Cocos2D, fue importante saber de dónde salían dichos cálculos y cómo se conseguían los valores que la librería utiliza para generar partículas. Luego, se podría modificar y adaptar más fácilmente dicho código puesto que ya se conoce la lógica base del sistema.

El sistema de partículas fue realizado empleando como base la librería de partículas que ofrece Cocos2D, presentadas a continuación:

- **CCParticleExamples**: Ofrece ejemplos en código fuente de partículas para poder modificarlos cuando sea necesario o para observar como funciona el código sin necesidad de profundizar demasiado en la lógica del código padre. Hereda las clases CCParticleSystemPoint, CCParticleSystemQuad, CCParticleSystem.
- **CCParticleSystemPoint**: Hereda de la clase CCParticleSystem. Usa 1 vértice (x,y) por partícula, contiene las texturas y movimientos giratorios que estas generan. El tamaño no puede ser mayor de 64 partículas. El sistema no puede ser escalar ya que las partículas se renderizan utilizando la extensión GL_POINT_SPRITE. En iPhones de 3era generación e iPads esta clase resulta más lenta que CCParticleSystemQuad.
- **CCParticleSystemQuad:** Es una subclase de la clase CCParticleSystem y genera partículas del tipo quad. El tamaño de las partículas puede ser de cualquier número real, el sistema puede ser escalar, las partículas pueden rotar, es más rápido que CCParticleSystemPoint pero consume más memoria RAM y GPU que dicha clase.
- **CCParticleSystem:** Esta es la clase padre, cuyo código fuente ofrece toda la lógica del sistema de partículas, y es la más abstracta de las clases mencionadas anteriormente. Se instancia y calcula la posición, velocidad, fuerza, masa y gravedad. La duración de la vida de las partículas, la tasa de emisión, velocidad, dirección, aceleración radial, etc.

3.2.1.1 Sistema Fuego

El código empleado para generar las partículas de fuego fue tomado de la librería de cocos2D, sin embargo se realizaron cambios importantes para adaptarlo a las necesidades de nuestra aplicación y al efecto que deseábamos mostrar. En el código presentado a continuación se muestra el init de la librería de Cocos **CCParticleExamples.m** con cambios realizados en el archivo **CCParticleFire**, y se muestra el código de la clase **EfectoFuego.m** donde se hace la llamada con los parámetros necesarios para la generación de las partículas.

```
// CCParticleExamples.m de la librería cocos2D
// Partícula Fuego
//
@implementation CCParticleFire
-(id) init
{     //Se inicia la llamada de las partículas
      return [self initWithTotalParticles:250];
}

-(id) initWithTotalParticles:(int) p
{
      if( (self=[super initWithTotalParticles:p]) ) {

        // duración de la particula
        duration = kCCParticleDurationInfinity;

        // Modo Gravedad
        self.emitterMode = kCCParticleModeGravity;

        // Modo Gravedad: gravity
        self.gravity = ccp(0,0);

        // Modo Gravedad: aceleración radial
        self.radialAccel = 0;
        self.radialAccelVar = 0;

        // Modo Gravedad: velocidad de las partículas, dichos parámetros
        // se ajustaron de acuerdo a nuestras necesidades
        self.speed = 60;
        self.speedVar = 20;

        // El ángulo de inicio
        angle = 90;
        angleVar = 10;

        // posición del emisor. Se ubica a la mitad de la pantalla.
        CGSize winSize = [[CCDirector sharedDirector] winSize];
        self.position = ccp(winSize.width/2, 60);
        posVar = ccp(40, 20);
```

Continua en la siguiente página

```
        // Vida de las partículas. Dura 0.25 frames
        life = 3;
        lifeVar = 0.25f;

        // tamaño, en pixels
        startSize = 54.0f;
        startSizeVar = 10.0f;
        endSize = kCCParticleStartSizeEqualToEndSize;

        // emisores por frame
        emissionRate = totalParticles/life;
        // color de las partículas en RGBA: inicio y final
        startColor.r = 0.76f;
        startColor.g = 0.25f;
        startColor.b = 0.12f;
        startColor.a = 1.0f;
        startColorVar.r = 0.0f;
        startColorVar.g = 0.0f
        startColorVar.b = 0.0f;
        startColorVar.a = 0.0f;
        endColor.r = 0.0f;
        endColor.g = 0.0f;
        endColor.b = 0.0f;
        endColor.a = 1.0f;
        endColorVar.r = 0.0f;
        endColorVar.g = 0.0f;
        endColorVar.b = 0.0f;
        endColorVar.a = 0.0f;

      //se añade una textura de fuego
       self.texture = [[CCTextureCache sharedTextureCache] addImage: @"fire.png"];

        // additive
        self.blendAdditive = YES;
      }

      return self;
}
@end
```

Luego se realiza la clase propia **EfectoFuego.m** donde se hace la llamada a esta clase

```
// EfectoFuego.m llamada al método que genera las partículas de fuego
//se declaran las variables que se observaran dependiendo del tipo de sistema operativo. El archivo pvr es el tipo gráfico que se emplea en los móviles
#ifdef __IPHONE_OS_VERSION_MAX_ALLOWED
#define PARTICLE_FIRE_NAME @"fire.pvr"
#elif defined(__MAC_OS_X_VERSION_MAX_ALLOWED)
#define PARTICLE_FIRE_NAME @"fire.png"
#endif

-(void) onEnter
{
      [super onEnter];
      self.emitter = [CCParticleFire node];
      [self addChild: emitter z:10];
      emitter.texture    =    [[CCTextureCache    sharedTextureCache]    addImage: PARTICLE_FIRE_NAME];
      // Se calcula el tamaño de la pantalla
      CGSize size = [[CCDirector sharedDirector] winSize];
      emitter.position = ccp( size.width /2 , size.height/2 );

}
```

En el código anterior se observa la llamada al método onEnter, que suele ser utilizado después del init en Cocos2D para realizar acciones en las escenas. En este caso se instancia el nodo de la clase **CCParticleFire**, se asigna la textura y se calcula para asignar luego el tamaño de la pantalla donde aparecerá el emisor de la partículas.

Figura 54. Aparece la llama en pantalla

Figura 55. La llama se dispersa con el tacto

La textura de fuego fue extraída de la librería de Cocos2D como un archivo pvr, que es reconocido por los sistemas móviles como una textura y la renderiza como tal. En algunos casos se emplea PVRTC de 2 a 4 bits y RGBA de 16 y 32 bits como es el caso del elemento

Fuego en el que se empleó RGBA. El framework de Cocos2D encapsula los objetos para dibujar en pantalla con OpenGL [3.1] y mezclar las texturas usando un canal alpha que está definido en clases específicas para ello. En este caso se aplicó cambios sutiles al color RGBA en la clase CCParticleFire.

3.2.1.2 Elemento Aire

Para desarrollar el elemento aire, se pensó utilizar el micrófono del dispositivo, de este modo el usuario al soplar podrá visualizar las partículas volar, y dichas partículas adoptarán formas de hojas mediante la utilización de sprites[3.2].

El proceso de detección del soplido en el micrófono se puede separar en dos partes. Primero, se obtiene la entrada del micrófono y segundo se detecta el sonido que emite el usuario al soplar. El ruido que emite alguien al soplar directamente al micrófono está compuesto de sonidos de baja frecuencia, por lo que se empleó un filtro de baja intensidad para reducir los sonidos de altas frecuencias que no pertenecen a dicho soplido; cuando el nivel de la señal filtrada emita picos altos sabremos que alguien está soplando.

Para obtener el input del micrófono es necesario emplear la clase **AVAudioRecorder** que se encuentra en el framework **AVFoundation**, el cual hay que añadir a la biblioteca del proyecto.

Se instancia la variable de tipo **AVAudioRecorder** en el header o archivo cabecera (archivo .h). También se instancia la clase **CoreAudio**, pues se necesitarán algunas de sus constantes en un paso siguiente.

Seguidamente se realiza en el método onEnter (sector donde se ejecuta la escena) la obtención de la entrada del micrófono o donde se comienza a escuchar el micrófono.

```
-(void) onEnter
{
  [super onEnter];
   //se instancia el objeto de tipo NSURL para manejar URL (Uniform Resource
Locator) que indica un directorio
   NSURL *url = [NSURL fileURLWithPath:@"/dev/null"];
  //Se configura las opciones de audio
   NSDictionary *config = [NSDictionary dictionaryWithObjectsAndKeys:
                           [NSNumber numberWithFloat: 44100.0],
                           AVSampleRateKey,
                           [NSNumber numberWithInt:
                           kAudioFormatAppleLossless], AVFormatIDKey,
                           [NSNumber numberWithInt: 1],
                           AVNumberOfChannelsKey,
                           [NSNumber numberWithInt: AVAudioQualityMax],
                              AVEncoderAudioQualityKey,nil];
   NSError *error;
   //comienza la grabación o percepción del audio
   grabador = [[AVAudioRecorder alloc] initWithURL:url settings:config
                error:&error];
```

La función principal de **AVAudioRecorder** es, tal como su nombre indica, grabar (record) audio, y como función secundaria provee información del nivel de audio. Así que en esta parte descartamos la entrada de audio para enviarla al directorio athe*/dev/null* para luego

explícitamente encender la medición del audio. Se emplea un timer para comprobar los niveles de audio 30 veces por segundo.

```
//Si no ha originado ningún error el recorder comienza la medición de audio

if (grabador) {
      [grabador prepareToRecord];
      grabador.meteringEnabled = YES;
      [grabador record];
      nivelTimer = [NSTimer scheduledTimerWithTimeInterval: 0.03 target: self
                   selector:@selector(levelTimerCallback:)  userInfo:  nil  repeats:
YES];

  } else
       NSLog([error description]);
}
```

Detectando el soplido

Cada vez que el método timer es activado la variable lowPassResults se vuelve a calcular. Por conveniencia se convierte a una escala 0-1, donde el cero es silencio y uno es un volumen muy alto. Sabremos que alguien está soplando cuando el nivel del filtro pase dicho umbral. Escoger el nivel es un poco delicado, si se coloca muy bajo se activará fácilmente con cualquier sonido y si está muy alto la persona tendrá que soplar con la fuerza de un vendaval para que sea reconocido, para esta aplicación se empleó el valor de 0.55. Una vez se activa el filtro se envía la señal para comenzar a mostrar las partículas.

```
- (void)levelTimerCallback:(NSTimer *)timer {
      [grabador updateMeters];

      const double ALPHA = 0.05;
      double fuerzadelCanal = pow(10, (0.05 * [grabador
                                      fuerzadelCanal:0]));

      minResultados = ALPHA * fuerzadelCanal + (1.0 - ALPHA) *
                    minResultados;

      if (minResultados > 0.55)
      {
        self.emisor = [[CCParticleExplosion alloc]
                       initWithTotalParticles:5];
        [self addChild: emisor z:1];
        emisor.texture = [[CCTextureCache sharedTextureCache] addImage:
                          @"hoja.gif"];
        emisor.autoRemoveOnFinish = YES;


      }
}
```

Las texturas mostradas son sprites con formas de hojas del tipo png de un tamaño de 90x90, son controladas por la clase **CCtexture2D** que es parte de clase de ParticleSystem.m al cual

pertenece el objeto **emisor (emisor.texture)** que genera gráficos empleando OpenGL 2D desde imágenes o texto.

Las partículas de aire emplea el mismo proceso que para el elemento fuego, sólo que en este caso la gravedad cambia a modo radial y se añade el calculo tangente como se muestra en el código siguiente:

```
// duración de la emisión
 duration = 0.1f;

 self.emitterMode = kCCParticleModeGravity;

 // Gravedad
 self.gravity = ccp(0,0);

 // Velocidad de las partículas
 self.speed = 70;
 self.speedVar = 40;

 // Radial
 self.radialAccel = 0;
 self.radialAccelVar = 0;

 // Tangente
 self.tangentialAccel = 0;
 self.tangentialAccelVar = 0;
```

Figura 56. Elemento Aire: Se activan las partículas al soplar

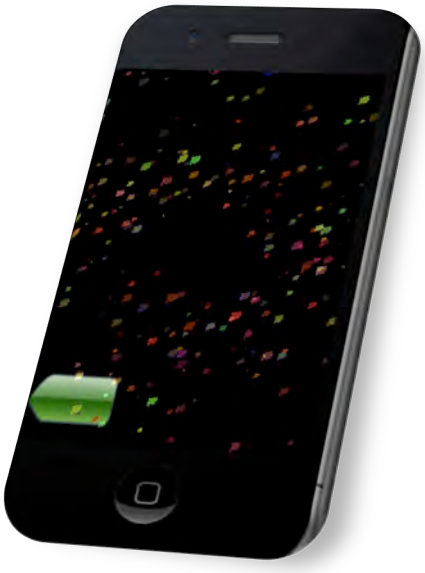


Figura 57. Elemento Aire: Las partículas desaparecen

3.2.2 Chipmunk Librería Física

Tanto para el juego como para el elemento tierra, se empleó una librería física de cuerpos rígidos en 2D denominada Chipmunk [28], que facilitaría la utilización de fuerzas, gravedad, fricción, entre otros.

Figura 58. Librería física Chipmunk integrada en Cocos2D

Dicha librería posee las siguientes características:

- **Cuerpos rígidos**: Un cuerpo rígido posee las propiedades físicas de un objeto (masa, posición, rotación, velocidad, etc) aunque no posee una figura por sí mismo. A diferencia de las partículas, los cuerpos rígidos pueden rotar.
- **Figuras de Colisión**: Se puede conectar un tipo de figura específica a un cuerpo, definiéndolo. Se pueden conectar tantas figuras a un solo cuerpo como se requieran para definir una estructura compleja, o ninguna si no se requiere una figura específica.
- **Articulaciones**: se pueden unir dos cuerpos mediante articulaciones y de este modo restringir su comportamiento.
- **Espacios**: Los espacios son la unidad básica de simulación en Chipmunk. Se añaden cuerpos, figuras y articulaciones a un espacio, y luego se actualiza dicho espacio como un todo.

Cuerpos rígidos, colisiones y sprites:

Suele existir confusión entre los cuerpos rígidos y la colisión con sus figuras (shapes en inglés) en chipmunk y como están asociadas éstas a los sprites. Un sprite es una representación visual de un objeto, por lo que es dibujado en la posición que se encuentra el objeto rígido. La figura de colisión vendría siendo la representación material del objeto y define como dicho objeto debería colisionar con otros objetos. Un sprite y una figura de colisión tienen poco que ver entre ellas a menos que se requiera coincidir el sprite con la figura de colisión.

```
typedef struct cpBody{
    cpFloat m, m_inv;
    cpFloat i, i_inv;
    cpVect  p, v, f;
    cpFloat a, w, t;
    cpVect rot;
}cpBody
```

En el cuadro se observan las variables reservadas correspondientes a las cualidades de las fuerzas, aceleración, masa, ángulos, etc.

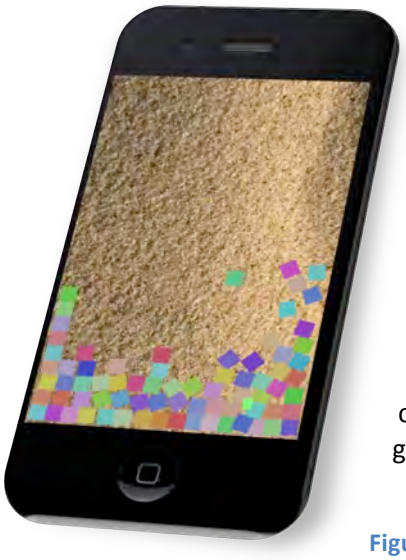
- m, m_inv corresponden a la masa y la inversa.
- i, i_inv corresponde al momento de la inercia y a su inversa.
- p, v, f son la posición, velocidad, y fuerza respectivamente.
- a, w, y t son el ángulo (en radianes), velocidad angular (rad/sec), y la torsión.
- rot es la rotación de un cuerpo cuya unidad es un vector de longitud.
- e es elasticidad.
- u es fricción.

3.2.2.2 Sistema Tierra

Encontrar la forma de representar al elemento Tierra virtualmente, fue un proceso que llevo mucho más tiempo que para el resto de elementos. Primero porque no sabíamos qué sistema podría asociarse con dicho elemento y segundo si encontrábamos la idea, ¿cómo se podría llevar a cabo en el tiempo estipulado?.

Dichas inquietudes produjeron la necesidad de realizar un pequeño brainstorming para determinar qué podía desarrollarse:

- Partículas o granos de arena que se movieran en el espacio y que el usuario pudiera interactuar con ellos mediante el tacto.
- Un reloj de arena, también interactivo mediante el tacto.
- Una caja de arena, similar a la que se utiliza en el feng shui e igualmente interactiva.
- Una caja llena de piedras, que se asociaría a la imagen de la aplicación.



Finalmente decidimos emplear la primera opción, que las partículas que se movieran en el espacio y que el usuario pudiera interactuar con ellas mediante el tacto. Para ello se empleó la librería Chipmunk y la clase escrita en C **drawSpace.c** para dibujar los cuerpos en pantalla. Al mismo tiempo, se trataría de aplicar la rotación de los cuerpos mediante la utilización del acelerómetro [3.3] permitiendo al usuario girar el dispositivo para mover las partículas además de tocarlas y desplazarlas por la pantalla. Puesto que poseen gravedad, masa y aceleración, estás caerán como cualquier objeto terrestre hacia su centro de gravedad, rebotarán y se deslizarán al chocar entre ellas.

Figura 59. Elemento Tierra

La figura 59 representa el sistema Tierra, en una versión beta, sin adaptación de los sprites para cada cuerpo de los objetos. A continuación se procederá a explicar parte del código empleado para la realización de este sistema:

Se instancia el método scene, ya que es parte fundamental de la librería o framework de Cocos2D:

```
+(id) scene
{
      CCScene *scene = [CCScene node];
      EfectoTierra *layer = [EfectoTierra node];
      layer.isTouchEnabled=YES;
      [scene addChild: layer];
      return scene;
}
```

Se devuelve la escena y la capa o "Layer" que será donde se realizará el efecto Tierra.

```
-(id) init
{
    if( (self=[super init] )) {

          [self crearEspacio];
          [self cajaContenedora];
            CCSprite *fondo = [CCSprite spriteWithFile:@"arena@2x.png"];
            fondo.anchorPoint = CGPointZero;
            [self addChild:fondo z:-1];

          [self scheduleUpdate];

        for (int i=160; i<=250; i++)
         {
            [self crearCajaenPosicion:ccp(i,i)];
         }

        mouse = cpMouseNew(space);
        self.isTouchEnabled = YES;
        }
      return self;
}
```

Podemos observar en el código init, la llamada al método "**crearEspacio**" para crear el entorno que contendrá los objetos y es imprescindible para el funcionamiento de la librería Chipmunk. Luego, hay que colocar una especie de barrera en la parte superior, inferior, derecha e izquierda de la pantalla para que los objetos no caigan fuera de ella y puedan ser visibles utilizando el método "**cajaContenedora**". Se coloca el fondo de pantalla, para luego utilizar un "**scheduleUpdate**" que permitirá actualizar el escenario cuando se creen las cajas.

Seguidamente se realiza un ciclo "para" crear cajas (objetos) aleatoriamente en el espacio, siendo una subclase de la clase drawSpace.c que dibuja dichas cajas.

Por último, se instancia el método que establece las articulaciones de la cajas para que puedan ser movidas por el tacto mediante el método cpMouseNew.

```
- (void)crearCajaenPosicion:(CGPoint)location {

      //tamaño de la caja
      float boxSize = 20.0;
      //masa que tundra la caja
      float mass = 0.1;

      cpBody *body = cpBodyNew(mass,  cpMomentForBox(mass, boxSize, boxSize));

      body->p = location;
      cpSpaceAddBody(space, body);
      cpShape *shape = cpBoxShapeNew(body, boxSize, boxSize);

     //cuadrado
      shape->e = 0.4;
      shape->u = 0.6;
      cpSpaceAddShape(space, shape);
}
```

El método **crearCajaenPosicion** como indica su nombre, crea las cajas en una posición especifica de la pantalla. Cada objeto posee masa, elasticidad y fricción, representadas por las variables "mass", "e" y "u", además de establecer un tamaño concreto a cada caja.

```
- (void) cajaContenedora{

      CGSize s = [CCDirector sharedDirector].winSize;

      int x = 4;
      int y = 4;
      int dmargin = x*2;
      int width = s.width - dmargin;
      int height = s.height - dmargin;
      cpShape * shape;
      cpBody *groundBody = cpBodyNewStatic();
      shape = cpSegmentShapeNew(groundBody, cpv(x,y), cpv(x+width, y), 0.0f);
      shape->e = 0.3; // elasticidad
      shape->u = 0.2; //friccion
      cpSpaceAddStaticShape(space, shape);

      shape = cpSegmentShapeNew(groundBody, cpv(x+width, y), cpv(x+width, y+height ), 0.0f);
      shape->e = 0.1; shape->u = 0.1;
      cpSpaceAddStaticShape(space, shape);

      shape = cpSegmentShapeNew(groundBody, cpv(x+width, y+height), cpv(x, y+height ), 0.0f);
      shape->e = 0.1; shape->u = 0.1;
      cpSpaceAddStaticShape(space, shape);

      shape = cpSegmentShapeNew(groundBody, cpv(x, y+height ), cpv(x, y), 0.0f);
      shape->e = 0.1; shape->u = 0.1;
      cpSpaceAddStaticShape(space, shape);
}
```

El método **cajaContenedora** crea bordes no visibles en la parte superior, inferior, izquierda y derecha de la pantalla que funcionan como fronteras entre el espacio interactivo de la aplicación y el resto de ella. Permite que los objetos creados no desaparezcan de la pantalla cayendo más allá del área visible. Éstos también poseen elasticidad y fricción, además de grosor.

3.2.2.3 Juego

El juego se realizó empleando la librería física Chipmunk y la librería **drawSpace.c** para dibujar las cajas en el espacio, algo necesario para que dicha librería física funcione. El juego creará cajas aleatoriamente a medida que el temporizador va contando en tiempo regresivo. Dichas cajas pueden ser eliminadas mediante el tacto, lo que sumará puntos. Al mismo tiempo irán cayendo otro tipo de cajas que al ser eliminadas restarán puntos o no. El objetivo del juego es acumular la cantidad de puntos máximo que se establece para ganar. Se puede pausar, reiniciar o salir del juego lo que representa los controles básicos del interactivo.

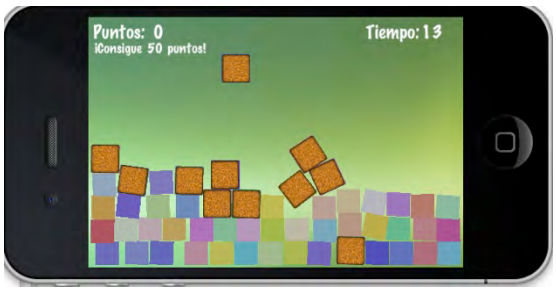


Figura 60. La cantidad de ítems ha alcanzado casi el top de la pantalla

Figura 61. El menú empleado en la versión beta

```
+(id) scene
{
      CCScene *scene = [CCScene node];
      juego *layer = [juego node];
      layer.isTouchEnabled=YES;
      [scene addChild: layer];
      return scene;
}
```

Se comienza declarando la escena y la capa principal para el desarrollo del juego.

```
-(id) init
{
      if( (self=[super init] )) {
          [self createSpace];
          [self cajaContenedora];

               mouse = cpMouseNew(space);
               self.isTouchEnabled = YES;

        [[SimpleAudioEngine sharedEngine] preloadEffect:@"poof.wav"];

           for (int i=160; i<=200; i++){
               [self createBoxAtLocation:ccp(i,i)];
            }
```

En el init, al igual que el resto de la aplicación se instancian los métodos y llamadas a clases o métodos. Se emplean, al igual que en el efecto Tierra los métodos **crearCajaenPosicion** y **cajaContenedora** para crear los objetos, el espacio y las fronteras, así como el método para activar el movimiento de los objetos con cpMouseNew. También, se carga en cache el audio que se ejecutará cuando un objeto sea eliminado y se crean un set de cajas para llenar el escenario.

```
[[CCSpriteFrameCache sharedSpriteFrameCache]
                     addSpriteFramesWithFile:@"juego.plist"];
batchNode = [CCSpriteBatchNode batchNodeWithFile:@"juego.png"];

[self addChild:batchNode];
```

Luego se asigna a la caché el "sprite sheet" que contiene los diferentes sprites que se emplearán en el juego, es decir las texturas de las cajas. Se debe emplear un plist, ya que

contiene las coordenadas de las sprites, de esta forma resulta más sencillo emplear un set de texturas que asignándolas una a una mediante código; al mismo tiempo emplea el archivo png para posicionar los gráficos sobre las coordenadas establecidas en el plist.

```
SmallBlockWoodSprite *block1b = [[[SmallBlockWoodSprite alloc] initWithSpace:space
location:ccp(272, 59)] autorelease];
        [batchNode addChild:block1b];


        CCSprite *fondo = [CCSprite spriteWithFile:@"background@2x.png"];
         fondo.anchorPoint = CGPointZero;
         [self addChild:fondo z:-1];

        //conteo de cajas aleatorias
        [self scheduleUpdate];
        [self schedule:@selector(timerUpdate:) interval:0.7];

        //Tiempo regresivo.
        score = 0;
        time = 60;
        [self schedule:@selector(countDown:) interval:1];

        //Crea y añade el label score como hijo

        labelPoint = [CCLabelTTF labelWithString:@"Puntos:" fontName:@"Marker Felt"
fontSize:24];
        labelPoint.position = ccp(40, 300);
        [self addChild:labelPoint z:1];

        scoreLabel = [CCLabelTTF labelWithString:@"0" fontName:@"Marker Felt"
fontSize:24];
        scoreLabel.position = ccp(90, 300);
        [self addChild:scoreLabel z:1];

        //crea el label de tiempo o temporizador

        labelTime = [CCLabelTTF labelWithString:@"Tiempo:" fontName:@"Marker Felt"
fontSize:24];
        labelTime.position = ccp(390,300);
        [self addChild:labelTime z:1];

        temporizador = [CCLabelTTF labelWithString:@"¡Ya!" fontName:@"Marker Felt"
fontSize:24];
        temporizador.position = ccp(440,300);
        [self addChild:temporizador z:1];

        //label info
        labelInfo = [CCLabelTTF labelWithString:@"¡Consigue 50 puntos!"
fontName:@"Marker Felt" fontSize:18];
        labelInfo.position = ccp(80, 280);
        [self addChild:labelInfo z:1];

        //boton de regreso
        [self boton];

      }
      return self;
}
```

Se prosigue a instanciar el objeto de la clase **SmallBlockWoodSprite** que selecciona un sprite determinado del plist**,** es decir, hace aparecer la caja con su textura.

Seguidamente se instancia y ejecuta el background y el método timerUpadate que realiza la generación de las cajas de forma aleatoria dependiendo del tiempo transcurrido, en este caso se genera una caja cada 0.7 fracción de segundo.

Al mismo tiempo se genera el método countDown que es el temporizador del juego, que irá de forma regresiva un segundo cada vez.

Lo siguiente es crear los Labels para el temporizador y el marcador de puntos. Se utiliza la clase CCLabelTTF y la fuente Marker Felt.

```
-(void) countDown:(ccTime)delta
{
      if (time != 0)
      {
        time -= 1;
        [temporizador setString:[NSString stringWithFormat:@"%i", time]];
      }else {
        [self endScene:TRUE];
      }
}
```

El método **countDown** genera el temporizador hasta que se hace 0 y termina la escena. Cuando esto ocurre inmediatamente se dispara el método menú, que verifica si has obtenido la puntuación adecuada para ganar o repetir el juego y mostrar la información adecuada.

```
- (void)addPoints
{
    score = score + 1;
    [scoreLabel setString:[NSString stringWithFormat:@"%i", score]];
}
```

addPoint realiza el conteo del marcador cada vez que la caja correcta es eliminada enviando al mismo tiempo un label con la puntuación al init.

```
-(void) timerUpdate:(ccTime)delta
{
      int numSeconds=1;
      numSeconds++;

      //posición random de las cajas en pantalla coordenadas Y
      int max = 310;
      int min = 240;
      double scale = (double) (max - min) / RAND_MAX;
      int val = min + floor(rand() * scale);

      //random coordenadas X
      int max2 = 410;
      int min2 = 50;
      double scale2 = (double) (max2 - min2) / RAND_MAX;
      int val2 = min2 + floor(rand() * scale2);

      //si el tiempo es divisible entre 2 aparece una caja, sino otra
    if (time%2 == 0)
      {
          SmallBlockWoodSprite    *block1b    =    [[[SmallBlockWoodSprite    alloc]
initWithSpace:space location:ccp(val2, val)] autorelease];
        [batchNode addChild:block1b];

      } else
        {
          [self crearCajaenPosicion:ccp(val2,val)];

        }

}
```

Es en **timerUpdate** donde se realiza el proceso para calcular la posición donde deberían aparecer las cajas de forma aleatoria, además si el tiempo en un momento es divisible entre 2 generará un tipo de caja, sino generará otra.

Continúa en la siguiente página

```
- (void)endScene:(BOOL)win {

    if (gameOver) return;
    gameOver = TRUE;

    CGSize winSize = [CCDirector sharedDirector].winSize;

    NSString *message;
    if (win) {
       if (score >= 50)
       {
          message = @"¡Has Ganado!";
          [self stopAllActions];
          [self unschedule:@selector(timerUpdate:)];
          [[SimpleAudioEngine sharedEngine] playEffect:@"win.wav"];
       }else{
              message = @"¡Intentalo de nuevo!";
              [self stopAllActions];
              [self unschedule:@selector(timerUpdate:)];
              [[SimpleAudioEngine sharedEngine] playEffect:@"lose.wav"];
       }
    } else {
        message = @"Intentalo de Nuevo";
       [self stopAllActions];
       [self unschedule:@selector(timerUpdate:)];
       [[SimpleAudioEngine sharedEngine] playEffect:@"lose.wav"];
    }

    CCLabelBMFont     *label     =     [CCLabelBMFont     labelWithString:message
fntFile:@"Arial.fnt"];
    label.scale = 0.1;
    label.position = ccp(winSize.width/2, 180);
    [self addChild:label];

    CCLabelBMFont   *restartLabel   =   [CCLabelBMFont   labelWithString:@"Restart"
fntFile:@"Arial.fnt"];

    CCMenuItemLabel  *restartItem  =  [CCMenuItemLabel  itemWithLabel:restartLabel
target:self selector:@selector(restartTapped:)];
    restartItem.scale = 0.1;
    restartItem.position = ccp(winSize.width/2, 140);

    CCMenu *menu = [CCMenu menuWithItems:restartItem, nil];
    menu.position = CGPointZero;
    [self addChild:menu];

    [restartItem runAction:[CCScaleTo actionWithDuration:0.5 scale:1.0]];
    [label runAction:[CCScaleTo actionWithDuration:0.5 scale:1.0]];

}
```

El método **endScene** genera el menú de opciones, evalúa si el usuario ha ganado o no. Se verifica el score para ello. Muestra las label con el mensaje de "Ganaste" o "Perdiste". Muestra el botón de "Restart" y genera los sonidos cuando se gana o pierde.

Esto es prácticamente lo más importante del código del juego, el resto son métodos propios de Cocoa como para controlar el tacto, la memoria, etc.

3.2.3 Efectos

Otra de las bondades que ofrece Cocos2D en su librería son los efectos, estos permiten generar distorsiones, movimientos horizontales, verticales, ondas, aumento de tamaño de los sprites (acercar y alejar), efecto espejo, efecto lente pez, efecto líquido, entre otros. Lo que ha facilitado realizar el Efecto Agua que será descrito a continuación.

3.2.3.1 Efecto Agua

El efecto agua es el último elemento de los cuatro que se desarrollaron para la aplicación iD-Stress. Este efecto debería ser lo suficientemente bueno, para simular el tacto del dedo con el agua, que generara ondas y luego desaparecieran lentamente. Para ello, se utilizó el efecto Ripple 3D que ofrece la librería de efectos de Cocos2D.

Figura 62. Efecto de ondas o Ripple

```
-(id) init
{
      if( (self=[super init] )) {
        [super onEnter];
        CGSize size = [[CCDirector sharedDirector] winSize];

        CCSprite *background = [CCSprite spriteWithFile:@"agua.jpg"];
        background.position = ccp(size.width/2, size.height/2);

        [self addChild: background];

      }
      return self;
}
```

El método **init** realiza las instanciaciones el background y ajusta el tamaño de la ventana con la clase director.

```
//Metodo que realiza el proceso de creación de las ondas en pantalla
-(void) addNewSpriteWithCoords:(CGPoint)p
{
      //Se calcula el eje horizontal y vertical
      int idx = CCRANDOM_0_1() * 1400 / 100;
      int x = (idx%5) * 85;
      int y = (idx/5) * 121;

      id rippleAction = [CCRipple3D actionWithPosition:CGPointMake(p.x,p.y)
radius:200 waves:10 amplitude:50 grid:ccg(32,24) duration:10];
      [self runAction:[CCSequence actions:rippleAction, [CCStopGrid action],
nil]];


}
```

Donde se realiza todo el proceso de generación de ondas en la pantalla es en el método **addNewSpriteWithCoords.** Éste recibe como parámetro los puntos o las coordenadas espaciales donde se ha generado un contacto con la pantalla para luego realizar los cálculos aleatorios de las ondas siguientes. Se instancia la clase CCRipple3D con los parámetros de amplitud de la onda, número de olas, el radio, duración. Una vez es generado el efecto ripple se ejecuta la acción de secuencia y se detienen las olas.

```
-(void) ccTouchesEnded:(NSSet *)touches withEvent:(UIEvent *)event {

      for( UITouch *touch in touches ) {
        CGPoint location = [touch locationInView: [touch view]];
        location = [[CCDirector sharedDirector] convertToGL: location];
        [self addNewSpriteWithCoords: location];
      }

}
```

El método ccTouchesEnded captura la locación donde se ha ejecutado un toque en la pantalla y ejecuta el método **addNewSpriteWithCoords** empleando las coordenadas obtenidas para la generación de las ondas.

CAPITULO IV

CONCLUSIONES

4.- Conclusiones

Después de haber realizado este proyecto se puede concluir a grandes rasgos lo siguiente:

- El sistema SCRUM sin duda alguna ha sido un factor importante al momento de organizar el trabajo del equipo, por su enorme funcionalidad y simplicidad en el manejo del tiempo, en el seguimiento semanal de la actividad de todos los integrantes del equipo sin caer en el estrés o presión excesiva -tan común en la realización de proyectos de gran envergadura- pues cada tarea ha sido escogida por cada miembro del equipo y segmentada por su importancia para ser realizada en el tiempo estipulado. Además permitió la participación de todos los involucrados en su organización.
- Los brainstorming deben hacerse cada vez que se quiera desarrollar un producto nuevo, o simplemente mejorar uno ya hecho. Pues te permite ser lo suficientemente creativo e ir más allá de lo que normalmente te permitirías, ya que no existe ningún tipo de restricciones en su planteamiento. Sobretodo, es importante que todos los miembros del equipo participen, ya que cada uno posee una experiencia distinta tanto en el aspecto laboral como de vida, lo que garantiza la riqueza en las ideas y de los posibles métodos de implementación.
- El trabajo en equipo multidisciplinar resulta muy eficiente, pues cada uno se debe centrar sólo en lo que se especializa sin necesidad de estar haciendo mil cosas que no lo involucran directamente.
- Gracias a este proyecto, se pudo conocer que existen diferencias entre el desarrollo de una aplicación nativa y una aplicación con orientación Web para móvil, permitiéndonos así seleccionar la mejor opción para el desarrollo de la nuestra.
- Al hacer el estudio del estado del arte se pudo tener una perspectiva mucho más amplia sobre nuestra idea de proyecto, pues se pudo comparar desde precios de ventas con respecto a calidad del producto en la app store de Apple, hasta implementaciones que podrían o no funcionar en nuestra aplicación. Éste fungió como timón para asentar mucho más nuestras ideas.
- Después de programar en XCode se pudo apreciar las grandes similitudes entre los lenguajes Objective-C y C y por qué el primero es subconjunto del segundo, al emplear librerías en C que acoplaban perfectamente con las clases de Objective-C.
- La programación en general empleando XCode y el Interface Builder resulta muy sencilla, por supuesto tienen limitaciones con respecto a implementaciones no genéricas, pero pueden ser resueltas al consultar los manuales, foros, guías y tutoriales que existen en la red que existen en abundancia.
- Parece asombroso la cantidad de frameworks y motores gráficos dirigidos a la creación de juegos y aplicativos interactivos para móviles y diferentes sistemas operativos, algunos requieren más tiempo de aprendizaje y práctica que otros, por lo que para el desarrollo de los juegos e interactivos de iD-stress el mejor candidato resultó ser Cocos2D ya que usa el mismo lenguaje Objective-C como base que XCode. Sin embargo, comenzar a emplearlo fue muy engorroso, primero porque desconocíamos que éste debía ser la base de la aplicación y no las clases desarrolladas con UIKit, por lo que se tuvo que hacer una adaptación de las vistas ya desarrolladas en UIKit para que se adaptaran a Cocos2D y seguir trabajando de esa forma, aunque se podían seguir empleando los elementos propios del interface builder en las pantallas que no requirieran como base algún interactivo.

- La realización del juego fue un enorme reto para mi, puesto que nunca había hecho un juego para móvil, ni había programado en el sistema operativo iOS por lo que terminó siendo una experiencia muy enriquecedora y satisfactoria.

Dificultades presentadas:

En el lapso que duró el proyecto, se presentaron muchas y variadas dificultades en el desarrollo de la aplicación, principalmente por la carencia de conocimiento de las herramientas empleadas, no obstante éstas fueron subsanadas en el transcurso del tiempo:

- Unir Cocos2D con el sistema UIKit de XCode fue el primer obstáculo a superar, como se ha comentado anteriormente no se podía colocar un módulo de juego solo en Cocos2D ni un solo modulo con los cuatro elementos (sistema de partículas, efectos) y que estos estuvieran embebidos en el UIKit. En ello duramos varios días, aprendiendo como empezar una aplicación con las características deseadas y finalmente la opción más lógica fue construirla desde Cocos2D e importar todas las clases de UIKit en las diferentes vistas, de este modo la aplicación arranca con Cocos2D pero luego se ejecuta la vista con UIKit.

- Los problemas de versiones entre XCode 3 y XCode 4 fue un quebradero de cabeza, puesto que la última versión, la cuatro, sólo funciona en las versiones del sistema Mac OS X 10.6.6 en adelante y puesto que desarrollamos en dos ordenadores, un iMac y un MacBook Pro debían ambos tener dichas versiones para poder desarrollar la aplicación sin problemas. Así que tuvimos que adquirir para la MacBook Pro la actualización del sistema operativo Leopard OSX 10.6.6.

- Al iniciar el proyecto se empleó una versión antigua de Cocos2D ya que en ella se conseguían los ejemplos y tutoriales, sin embargo una vez comenzamos a programar fue necesario emplear una versión más actual, ya que existían métodos y clases obsoletas que no eran soportadas por el XCode 4 y por las librerías físicas como Chipmunk y el SpaceManager.

  Por ejemplo al momento de hacer la declaración en la cabecera de una clase propia de Cocos2D que en la versión anterior se instanciaba de la siguiente manera:
  #include "constraints/util.h" generaba errores con las librerías físicas, ya que en la nueva versión dicha instancia fue cambiada a **#include "util.h"** .

  Igualmente aparecían otra clases de errores que para ser descubiertos tuve que emplear mucho tiempo de investigación. Por ejemplo, en un momento dado había que añadir una nueva arquitectura i386 para que pudiera ser compilada en las máquinas Intel cosa que no era necesario especificar en la versión antigua.

- Para poder asignar imágenes a los objetos en el juego, es necesario emplear un plist (property list) que es una estructura de datos que almacena las preferencias de los sprites que se han de emplear en el juego. Sin embargo, fue complicado que el archivo creado para nuestro juego fuera reconocido por el Cocos2D, lo que involucró una investigación ardua para encontrar los motivos que lo ocasionaban.

- Otro obstáculo menor fue que las fuentes de los labels de texto empleados por primera vez en el juego no aceptaban símbolos especiales, como los de interrogación o

de exclamación "¡¿", por lo que se tuvo que cambiar a fuentes que reconocieran dichos signos empleados en el castellano.

- Los dispositivos iPhone 4 emplean una nueva resolución de imágenes denominado Sistema Retina o "Retina Display" que consiste en una pantalla de alta gama con cuatro veces más pixels que una pantalla del mismo tamaño de un dispositivo normal. Por lo que hay que adaptar todas las imágenes al doble de resolución para que se adapten tanto a este dispositivo como a las versiones anteriores de iPhone.

4.1 Líneas de Futuro

A pesar de que el trabajo ha cumplido los requerimientos primordiales especificados por el cliente y se han satisfecho nuestros objetivos, se pueden nombrar algunas características que no fueron agregadas para esta versión de la aplicación y que probablemente serán completadas a posteriori:

- Soporte para iPad, iPod Touch y Android
- Actualización de versiones de los cuatro elementos en cuanto a estética y rendimiento
- Soporte para el idioma francés
- Actualización estética del mini juego.

Hasta este momento de la memoria, no se ha logrado subir a la App Store de Apple la aplicación, por la carencia de tiempo y algunas dificultades en obtener la licencia de desarrollador para ejecutar la aplicación en iPhone así como también el mismo iPhone para realizar las pruebas de la aplicación. Sin embargo, no se ha descartado en ningún momento esta opción y se seguirá trabajando en ella hasta cumplir este cometido.

CAPITULO V
GLOSARIO Y REFERENCIAS

5. Glosario

Framework [1.1]: Es un esquema o esqueleto para el desarrollo de una aplicación de software.
API [1.2]: Application Programing Interface en inglés, es un conjunto de métodos y funciones que ofrece cierta librería para ser utilizado por otro software. Representa la interfaz de comunicación entre componentes de Software.
Smartphone [1.3]: es un término comercial para denominar a un teléfono móvil que ofrece más funciones que un teléfono celular. Fuente: Wikipedia.
Sofrología [1.4]: es como se denomina a una disciplina consistente en un conjunto de técnicas o métodos de relajación (diferente a la hipnosis) y de modificación de estados de conciencia que tiene como objetivo el establecer el equilibrio cuerpo-mente. Fuente: Wikipedia.
XCode[1.5]: Es el entorno de desarrollo intergado (IDE) de Apple para sus aplicaciones Mac Os X o iPhone.
Interface Builder [1.6]: Software de desarrollo para aplicaciones iPhone que ofrece los elementos visuales necesarios para la construcción de las mismas.
IDE[1.7]: Son las siglas en inglés de Entorno de Desarrollo Intergado. En él se desarrollan los programas de software.
Mockups[2.1]: Es una especie de borrador que funciona para plasmar las ideas de desarrollo de software de forma gráfica antes de llevarlas a un programador para que las implemente.
iOS [2.2]: Sistema Operativo móvil de Apple
Tab Bar [2.3]: Es una barra de navegación propia del sistema iOS, dará acceso a diversas vistas dentro de la aplicación separándola en módulos independientes, indicándole al usuario donde se encuentra en todo momento.
UIKit [2.4]: Es un framework presente en el Interface Builder y que provee las clases para construir y controlar la interfaz gráfica de una aplicación iOS.
Navigation Views[2.5]: Corresponde a las diversas vistas de navegación que se encuentran en una aplicación iPhone.
Capas de Abstracción [2.6]: es una forma de ocultar los detalles de implementación de ciertas funcionalidades en el iOS.
App Stores[2.7]: Son las tiendas virtuales donde se encuentran usualmente las aplicaciones para móvil, aunque también se puede encontrar ebooks, archivos de música, videos, etc. Todos para la compra.
eBooks[2.8]: Son los libros virtuales que pueden ser visualizados en cualquier dispositivo u ordenador.
Game Engine [2.9]: Sistema creado para el desarrollo de videojuegos.
Angry Birds[2.1.1]: Juego para móvil muy popular que consiste en disparar pequeños pájaros con diferentes propiedades a diferentes estructuras con el objetivo de derrotar a unos cerdos que han robado los huevos de éstos pájaros. Emplea un sistema físico.
Space Invaders [2.1.2]: Videojuego arcade que consiste en eliminar aliens mediante el empleo de un cañón laser y obtener la mayor puntuación posible.
Tetris [2.1.3]: Es un videoJuego de Puzzle que cuyo objetivo es hacer coincidir los objetos en forma horizontal y ganar puntos.
Cocos2D [2.1.4]: Es un framework orientado a la creación de aplicaciones interactivas y juegos en 2D para móviles.

OpenGL [3.1]: Es una librería orientada a la creación y manipulación de imágenes, gráficos en el ordenador, y en los dispositivos móviles.
Sprites [3.2]: Se le denominan Sprites a las imágenes que se emplean en los video juegos.
Acelerómetro [3.3]: Sensor presente en los iPhone que captura el movimiento del mismo.

6. Referencias

[1] Millenian Media. www.millennialmedia.com/research/. Consultado: Febrero 2011.

[2] ComScore, *2010 Mobile year in review*. www.comscore.com. Consultado: Febrero 2011.

[3] Sofia Fougueras. Estudio sobre el futuro de las Aplicaciones Móviles en la industria médica y de la salud. http://guiacirugiaestetica.com/estudio-sobre-el-impacto-de-las-aplicaciones-moviles-en-la-industria-medica-y-de-la-salud/. Consultado: Marzo 2011.
[4] Clínica Enlace. http://www.enlacebcn.com/. Consultado: Marzo 2011.

[5] INE (Instituto Nacional de Estadística). *España en Cifras*. http://www.ine.es/prodyser/pubweb/espcif/espcif10.pdf. Consultado: Marzo 2011.

[6] Pedro R. Gil Monte. *El síndrome de quemarse por el trabajo (síndrome de burnout): aproximaciones teóricas para su explicación y recomendaciones para la intervención. 2001*. Universidad de Valencia. Consultado: Abril 2011.

[7] Dr. Antonio Cano Vindel. *Epidemiología y Costes del Estrés Laboral*. 2002. http://www.psicologiacientifica.com/bv/psicologiapdf-78-el-sindrome-de-quemarse-por-el-trabajo-(sindrome-de-burnout)-aproximaciones-teor.pdf. Consultado: Abril 2011

[8] . José María Peiró. *Desencadenantes del Estrés Laboral*, Universidad de Valencia. Ediciones Pirámide.2005. Consultado: Marzo 2011.

[9] Man-Wan Lo. *Occupational Stress in the Information Systems Profession*. 1987. Consultado: Abril 2011.

[10] Proyectos Agiles. *¿Qué es SCRUM?.* http://www.proyectosagiles.org/que-es-scrum. 2008. Consultado: Abril 2011.

[11] Jorge Serrano. *Explicando SCRUM a mi abuela*. 2007. http://geeks.ms/blogs/jorge/archive/2007/05/09/explicando-scrum-a-mi-abuela.aspx. Consultado: Abril 2011

[12] Emiliano Martínez. *iD-Stress : Diseño interfaz gráfica y experiencia del usuario*. 2011. Consultado: Junio 2011.

[13] Pedro Cid Segarra, *VoxUJI Radio iPhone Client*, 2010. Tesis. Consultado: Mayo 2011.

[14] Mike Johnes. *iPhone Simulator*. http://www.freemacware.com/iphone-simulator. 2007. Consultado: Marzo 2011

[15] Selva Digital. *Arquitectura del Sistema Mac OS X.* http://www.sobrepc.com/programacion/MacOSX/macosx2.html. 2010. Consultado: Abril 2011.

[16] Richard A. Sequera. *Lenguaje C.* http://www.monografias.com/trabajos4/lenguajec/lenguajec.shtml. 2010. Consultado: Abril 2011

[17] Emilio Áviles. *Las aplicaciones móviles son el canal de venta del futuro*. http://techmi.es/blog/about/. 2011. Consultado Mayo 2011.

[18] Brian Fling . *Mobile Design and Development*. 2009. Editoria O'Reilly. Pp 75

[19] Brian Fling . *Mobile Design and Development*. 2009. Editoria O'Reilly Pp 77

[20] *Cocos2D for iPhone*. http://cocos2D-iPhone.org. 2011. Consultado: Febrero 2011

[21] *Shiva 3D*. http://www.stonetrip.com/. 2011. Consultado: Febrero 2011.

[22] *iProcessing*. http://www.luckybite.com/iprocessing/. 2011. Consultado: Febrero 2011.

[23] *Adobe Intgrate Runtime*. http://es.wikipedia.org/wiki/Adobe_Integrated_Runtime. 2011. Consultado: Febrero 2011

[24] Jesse Schell. *The Art of Game Design- A Book of Lenses.* Morgan Kauffman. 2008. Consultado: Mayo 2011.

[25] Marc Leblanc. *8Kinds of Fun.* http://algorithmancy.8kindsoffun.com/. 2008. Consultado: Mayo 2011.

[26] Los Cuatro Elementos. http://es.scribd.com/doc/2265962/Los-Cuatro-Elementos. 2007. Consultado: Marzo 2011.

[27] Andrew Witki. *Physically Based Modeling: Principles and Practice Particle System Dynamics.*1997. Consultado: Mayo 2011.

[28] *Chipmunk Physics.* http://code.google.com/p/chipmunk-physics/. 2010. Consultado: Abril 2011.

7. Referencia Figuras

Figura 1: *ITU World Telecommunication/ICT Indicadors Database*. http://www.itu.int/ITU-D/ict/material/FactsFigures2010.pdf
Figura 2: Gizmovil. http://gizmovil.com/2010/11/android-rentabilidad-publicidad-apps
Figura 3: AppleSfera**.** http://www.applesfera.com/apple/los-usuarios-de-ios-doblan-a-los-de-android-en-europa-por-cuanto-tiempo#c306407
Figura 4: Research2guidance [3]
Figura 5: INE (Instituto Nacional de Estadística). *España en Cifras*. http://www.ine.es/prodyser/pubweb/espcif/espcif10.pdf. Consultado: Marzo 2011.
Figura 6: Desencadenantes del estrés laboral. José María Peiró. Universidad de Valencia.
Figura 34: http://es.wikipedia.org/wiki/IOS_(sistema_operativo)
Figura 35: http://iphoneroot.com/wp-content/uploads/2008/04/iphone-java-me.JPG
Figura 36: http://ravishankarb.wordpress.com/
Figura 37:http://www.wayerless.com/
Figura 51: Imagen: http://www.cocos2d.org/doc/programming_guide/index.html
Figura 52: http://www.cocos2d-iphone.org/
Figura 53: Andrew Witki. *Physically Based Modeling: Principles and Practice Particle System Dynamics.*1997.