|  |
| --- |
| Uady  Universidad autónoma de yucatán |
| Paquete Computacional |
| Proyecto: Estrategias Innovadoras para la enseñanza de las matemáticas en el nivel básico |
| **M. I. Jorge Carlos Reyes Magaña**  **M. C. Luis Ramiro Basto Díaz**  **M. C. Genny Rocío Uicab Ballote** |
| **01/10/2011** |

|  |
| --- |
| El paquete computacional consiste en una serie de aplicaciones móviles que sirven de apoyo al proyecto PRIORI convocatoria 2009. Clave del proyecto: FMAT-10-012. |

Manual de Usuario

Manual Técnico

Contenido

[Introducción 4](#_Toc305451624)

[Manual Técnico 6](#_Toc305451625)

[Manual técnico de la aplicación “Comparando Longitudes” 6](#_Toc305451626)

[Introducción 6](#_Toc305451627)

[Paquete alerta 7](#_Toc305451628)

[Paquete Canvas 7](#_Toc305451629)

[Paquete imágenes 8](#_Toc305451630)

[Paquete memorama 8](#_Toc305451631)

[Paquete Midlet 9](#_Toc305451632)

[Paquete preguntas. 9](#_Toc305451633)

[Manual técnico de la aplicación “Ordenando Fracciones” 10](#_Toc305451634)

[Introducción 10](#_Toc305451635)

[Cocos2D para Iphone 11](#_Toc305451636)

[Documentación técnica 17](#_Toc305451637)

[Ciclo de ejecución de la aplicación 18](#_Toc305451638)

[Manual técnico de la aplicación “¿Cuánto falta? ¿Cuánto sobra?” 20](#_Toc305451639)

[Introducción 20](#_Toc305451640)

[Estructura 20](#_Toc305451641)

[Escenas y componentes 23](#_Toc305451642)

[Clases y funciones 24](#_Toc305451643)

[Manual técnico de la aplicación “Batalla naval” 28](#_Toc305451644)

[Introducción 28](#_Toc305451645)

[Objetivo 28](#_Toc305451646)

[Restricciones 28](#_Toc305451647)

[Funciones Principales 28](#_Toc305451648)

[Manual de Usuario 33](#_Toc305451649)

[Manual de usuarios de la aplicación “Comparando longitudes” 33](#_Toc305451650)

[Opción Jugar 34](#_Toc305451651)

[Opción Instrucciones 39](#_Toc305451652)

[Manual de usuario de la aplicación “Ordenando fracciones” 40](#_Toc305451653)

[Primer nivel 40](#_Toc305451654)

[Segundo nivel 41](#_Toc305451655)

[Tercer nivel 42](#_Toc305451656)

[Manual de usuario de la aplicación “¿Cuánto falta? ¿Cuánto sobra?” 44](#_Toc305451657)

[Instalación 44](#_Toc305451658)

[Ejecución 46](#_Toc305451659)

[Manual de usuario de la aplicación “Batalla Naval” 51](#_Toc305451660)

[Ejecución 51](#_Toc305451661)

# Introducción

El objetivo general del proyecto Estrategias Innovadoras para la enseñanza de las matemáticas en el nivel básico es:

“Considerando nuestro referente, el proyecto tiene como propósito, proporcionar a los educandos de nivel básico una nueva perspectiva de estudiar matemáticas, a través de actividades diseñadas para trabajar tanto en un entorno tecnológico, como resolución de problemas de ingenio, a través del uso de materiales didácticos tangibles o juegos didácticos. Las actividades se enmarcarán en los planteamientos de la reforma educativa vigente de nuestro país (Planes y programas de estudio correspondientes), de que los estudiantes construyan los conocimientos a través de actividades que susciten su interés, mantengan su atención y los hagan involucrarse en la resolución de un problema. “

Apegándonos a dicho objetivo se han diseñado diferentes aplicaciones móviles que se ejecutan en los dispositivos Iphone (Ipod Touch, Ipad), Palm Pre, Teléfonos celulares con tecnología J2ME (Java Micro Edition). Las aplicaciones diseñadas fueron adaptadas de diversos libros llamados Ficheros de actividades didácticas para Matemáticas, que se utilizan a nivel básico en diferentes grados. Se adaptaron de acuerdo a las características de cada actividad que sean transferibles a un dispositivo digital sin la pérdida del objetivo original para el cual fue diseñada.



Fichero Actividades didácticas

Siguiendo esta metodología se diseñaron cuatro aplicaciones:

* Comparando longitudes
* ¿Cuánto falta? ¿Cuánto sobra?
* Ordenando fracciones
* Batalla Naval

El proyecto contó con la colaboración de los estudiantes:

Daniel Rueda Jimenez

Wilberto Dzul Canché

Aurelio Mex Che

Ricardo Herrera Escalante

# Manual Técnico

## Manual técnico de la aplicación “Comparando Longitudes”

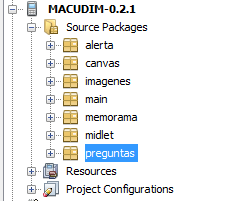
### Introducción

La aplicación Comparando Longitudes fue desarrollada utilizando la plataforma Java, en su versión MicroEdition. El programa utiliza el perfil MIDP 2.0 y la configuración CLDC 1.1 la cual es la más común en la mayoría de los dispositivos móviles que se utilizaron para su prueba.

Para la instalación del programa en cada dispositivo, de manera general basta con copiar el archivo .jar que se encuentra en la carpeta dist del proyecto en una localidad que sea accesible de la memoria (ya se interna o removible del dispositivo) y proceder a su instalación. Cada dispositivo o teléfono celular tiene una forma distinta de cómo llegar al menú de aplicaciones y es por esta razón que no se presenta de manera detallada esos pasos. Sin embargo de manera general se puede establecer que desde la opción de las Aplicaciones del teléfono, se selecciona la opción de “Instalar” para que se otorguen los permisos necesarios y así proceder posteriormente a la ejecución.

El programa es compilable con NetBeans versión 6.1 y también posteriores (descargable desde <http://netbeans.org/downloads/index.html> ). De preferencia se debe seleccionar la versión más completa de Netbeans o en su defecto la que contenga al menos el compilador de J2ME.

El proyecto tiene la siguiente estructura.



Dentro de cada paquete del proyecto se tienen distintos archivos fuentes (con extensión .java) y se describirá cada uno en la siguiente sección.

### Paquete alerta

En este paquete se tienen los archivos:

* AlertaConfirmacion
* AlertaConfirmacionListener

Sirven para la manipulación de mensajes que aparecen durante la aplicación. Permite conocer si el usuario presionó “Si” de acuerdo al mensaje que se haya recibido en el constructor de la clase AlertaConfirmacion.

### Paquete Canvas

En el paquete se encuentran los archivos:

* CanvasJuego
* CanvasJuegoListener
* CanvasMemorama
* CanvasMemoramaListener
* CanvasMultijuego
* CanvasVistaAnimal
* Cronometro
* FormPregunta
* FormPreguntaListener

Todos los archivos que terminan en su nombre con Listener hacen referencia a clases que son interfaces del paradigma orientado a objetos, es decir clases que obligan la implementación de ciertos métodos. Éste paquete contiene todos los Canvas que se pintan en el programa, donde un Canvas significa lienzo donde se colocan los dibujos de la aplicación, los cuadros del memorama, las imágenes de los animales, etc.

**CanvasJuego**. Contiene el tablero principal de la aplicación, por ejemplo en el método *inicializarComponentes* se establecen las dimensiones del tablero que contiene las cartas y la parte lógica del mismo. Así como las dimensiones y las coordenadas de pintado. *pintarMesa()*realiza gráficamente el trazo de las líneas y rectángulos correspondientes al memorama.

**CanvasMemorama**. El método principal de esta clase es: *commandAction* que responde a las diferentes acciones que se pueden realizar desde el tablero del memorama, dependiendo de la selección del usuario realiza la tarea que se indique, ya sea mostrar las preguntas o mostrar los animales desde una imagen mayor.

**CanvasVistaAnimal.** Debido a que las imágenes de los animales son pequeñas existe una vista que permite apreciar con un tamaño más grande el detalle de cada animal y dicha pantalla se muestra desde este archivo. El método principal de esta clase es: *pintarVista(Graphics g)*.

**Cronometro.** Clase que crea el cronómetro de la aplicación.

**FormPregunta**. La aplicación consta de una serie de preguntas que el usuario debe responder para reforzar el conocimiento que se ha aprendido. Para dicho fin, se crean las preguntas de esta clase que internamente está construido como un formulario.

### Paquete imágenes

Dentro de este paquete se almacenan todas las imágenes que son necesarias para la aplicación, como por ejemplo la serie de sprites que hacen el efecto de cuando las cartas del memorama son seleccionadas. Asimismo se incluyen las imágenes de los animales con sus respectivas medidas de los cuernos.

### Paquete memorama

Este paquete incluye los archivos:

* AdministracionMesa
* Memorama
* Tarjeta
* TarjetaMiniSprite
* TarjetaSprite

**AdministracionMesa**. Contiene la lógica de cuando el usuario se desplaza a través del memorama, por medio de las funciones: moverTarjetaIzquierda(),moverTarjetaDerecha(),moverTarjetaArriba(), etc.

**Memorama**. Clase que crea los sprites del tablero y la animación al seleccionar alguna de las cartas por medio del método *animacionTarjetaAbrir*().

**Tarjeta**. Clase de entidad para cada una de las tarjetas del juego.

**TarjetaMiniSprite**. Clase de entidad para la versión pequeña de cada una de las tarjetas.

**TarjetaSprite**. Contiene los métodos: ocultarTarjeta(),establecerTarjetaAmpliada(), que son considerados los más importantes para esta sección de código.

### Paquete Midlet

Contiene la clase:

* **Midlet**. Esta clase es la principal que ejecuta la aplicación.

### Paquete preguntas.

Contiene los archivos:

* AdministradorPreguntasDeAnimales
* AnimalConCuerno
* Pregunta1
* Pregunta2
* Pregunta3
* PreguntaDeAnimalesConCuerno
* RespuestaMultiple
* RespuestaSimple

**AdministradorPreguntasDeAnimales**. Clase que administra todo el cuestionario, así como las diferentes clases de preguntas, del mismo modo hace referencia a las demás clases que son parte de este paquete.

**AnimalConCuerno.** Clase de entidad asociada a cada animal con su imagen correspondiente.

**Pregunta 1,2,3.** Clases que contienen las preguntas que se mostrarán en el cuestionario.

**PreguntaDeAnimalesConCuerno**. Clase de entidad para los animales con cuerno.

## Manual técnico de la aplicación “Ordenando Fracciones”

### Introducción

Otra aplicación que forma parte de este proyecto se desarrolló sobre la plataforma IOS, es decir se ejecuta sobre dispositivos móviles Apple, en particular la aplicación desarrollada fue probada con los teléfonos Iphone y reproductores de música de la misma marca conocidos como IpodTouch.

|  |  |
| --- | --- |
| http://t0.gstatic.com/images?q=tbn:ANd9GcQ7FbSgwDLSvnzUz6XfTZtsVuQZ-VMjw7PsHdHDxy2PzpAwCnv_ | http://t1.gstatic.com/images?q=tbn:ANd9GcTcJaLfPjhmVinbI_oTxK32lTm3dzYNIe4ZYJZnO7ZhCHnULFnQgA |
| Iphone | Ipod Touch |
|  |  |

Ambos dispositivos son muy similares, incluso en el sistema operativo que manejan. La principal diferencia radica en que el Iphone incluye el teléfono, y el IpodTouch no.

Estos dispositivos son completamente táctiles, su pantalla contiene la tecnología multitouch que permite manipular objetos con varios dedos simplemente al tocar la pantalla lo cual le brinda al usuario una mayor facilidad y comprensión de las aplicaciones. Asimismo la resolución de la pantalla es mejor y la capacidad de colores que puede desplegar es muy alta, brindando mayor claridad y atractivo a los juegos que se desarrollen, es importante recordar que las aplicaciones han sido pensadas para niños en el nivel básico (primaria) por lo que estas características son importantes para llamar la atención del niño y le resulte interesante y atractivo.

La aplicación “Comparando fracciones”, se desarrolló utilizando un framework (marco de trabajo) llamado Cocos 2D, que se enfoca principalmente en el desarrollo de juegos y el manejo de sprites. De esta forma la primera parte del manual técnico hace referencia a este marco de trabajo.

### Cocos2D para Iphone

Herramientas necesarias para el desarrollo de videojuegos para iPhone

Hardware:

* Computadora Mac con SO Snow Leopard 10.6 o superior
* Dispositivo Móvil Apple (iPod Touch, iPhone, iPad)

Software:

* SDK para iPhone / iPad
* Xcode

Conocimiento básico:

* Programación en C/C++
* Conceptos de programación orientada a objetos

Conocimiento recomendado:

* Uso de sprites

Lecturas recomendadas

* Learning Objective-C: A Primer
* Memory management programming guide for Cocoa
* Xcode Debugging Guide

También es recomendable leer las siguientes guías de desarrollo que brinda Apple:

* Cocoa Fundamentals Guide
* The Objective-C Programming Language
* iOS Development Guide
* iPhone Human Interface Guidelines

Para desarrollar videojuegos en 2D se hará uso del Framework “Cocos2D for iPhone”

* Sitio oficial http://www.cocos2d-iphone.org/
* Descarga http://www.cocos2d-iphone.org/download
* Documentación oficial (Wiki) http://www.cocos2d-iphone.org/wiki/doku.php/

Tutorial de Cocos2D Framework

Para usar el Framework “Cocos2D for iPhone” es necesario haber instalado previamente iOS SDK. Se hará uso del IDE XCode para desarrollar las aplicaciones usando el framework.

* Conceptos básicos de Cocos 2D
* El juego tiene Escenas.
* Cada Escena tiene un número de Capas.
* Las Capas capturan la interacción del usuario y contienen a los sprites.
* El Director administra las escenas.

Una **escena es** una parte independiente del flujo de trabajo de la aplicación.

Algunos la llaman “Pantallas” o “Escenarios”. La aplicación puede utilizar muchas escenas pero solo una de ellas se mantiene activa en un momento dado. En Cocos2D las escenas están implementadas en objetos CCScene.

Cocos2D cuenta con un conjunto de clases llamadas Transiciones (implementadas con el objeto CCTransitionScene) el cual permite hacer transiciones entre dos escenas.

El **Director** (implementado en el objeto CCDirector) es el componente que se encarga de hacer los cambios entre las escenas. Mantiene una pila de las escenas para permitir las llamadas a escena. Pausar una escena y mantenerla mientras otra escena entra y posteriormente regresa a la escena original.

El CCDirector también es responsable de inicialización de OpenGL ES.

Las capas utilizan al director para que éste le indique cuando moverse. Una **capa** (CCLayer) tiene un tamaño de área dibujable, y sabe cómo dibujarse a sí mismo. Puede ser semitransparente permitiendo ver otras capas que se encuentren atrás. Las capas son las que definen la apariencia y el comportamiento.

En los objetos CCLayer es donde se definen los manejadores de eventos, los eventos son propagados desde el frente hacia atrás hasta que alguna capa capture el evento y lo procese. Ocupan la pantalla por completo. Configuran el manejo de los toques, el acelerómetro y pueden contener otras capas y sprites.

Los **Sprites** son imágenes 2D que pueden ser rotadas, escaladas, animadas, etc.

Los sprites (implementados usando la clase CCSprites) pueden tener otros sprites como hijos. Cuando un padre es transformado, todos sus hijos también son modificados.

Cocos2D for iPhone es una librería basada en Cocoa para crear juegos para iPhone, iPod Touch y iPad.

**1.- Descarga del código fuente**

Una vez instalado las herramientas de desarrollo de Apple y el SDK de iPhone se debe descargar los archivos fuente desde la página oficial.

La versión estable está en http://www.cocos2d-iphone.org/download

Posteriormente a la descarga se debe descomprimir el archivo.

Buscar el archivo “cocos2d-iphone.xcodeproj” y abrirlo con Xcode.

**2.- Ejecutar ejemplos**

El archivo abierto es un proyecto grande que contiene todo el código fuente de Cocos2D, además de una gran cantidad de subproyectos de ejemplos/demos.

Se pueden ejecutar los subproyectos para explorar las capacidades de Cocos2D, así mismo, se pueden leer el código fuente para aprender cómo realizar determinadas acciones en Cocos2D.

Para seleccionar un ejemplo, se necesita configurar el “Active Target” y “Active Executable”.

Hay dos formas de hacerlo. El primer método es usar los submenus “Set active target” y “Set active executable” que se encuentran en el menú Project. El segundo método es colocar estas opciones en la barra de herramientas principal, bajo el menú View, seleccionar “Customize toolbar...” y posteriormente arrastrar los items “Active Target” y “Active Executable” dentro de la barra de herramientas en la parte superior de la ventana. Entonces ya se podrán seleccionar los targets y ejecutables desde ahí.

Sin importar el método elegido, primero se debe seleccionar el “Active Target” que se desea ejecutar (Por ejemplo, “SpriteTest” o “ParticleTest”. Automáticamente el “Active Executable” cambiará para coincidir, pero si por alguna razón no ocurre, seleccionar manualmente el

“Active Executable” correspondiente.

Luego únicamente se necesita dar click en el botón “Build and Run”. El proyecto será compilado y el simulador para iPhone/iPad será lanzado y ejecutará el demo.

**3.- Instalar las plantillas de proyectos**

Actualmente no es necesario instalar algo más para poder usar Cocos2D. Es una librería fuente, e incluyendo las fuentes apropiadas dentro de un proyecto configurado adecuadamente, trabajará sin ningún problema.

Pero para hacer esta configuración más sencilla, cocos2d v0.99 viene con unas plantillas de proyecto para Xcode 3. Estas plantillas permiten crear fácilmente proyectos configurados para cocos2d, con algunos componentes opcionales:

* Una plantilla de cocos2d únicamente
* Una plantilla de cocos2d + box2d
* Una plantilla de cocos2d + cipmunk

Para instalar las plantillas se debe ejecutar la aplicación Terminal y escribir lo siguiente:

cd cocos2d-iphone

./install-template.sh

Es importante reemplazar “cocos2d-iphone” de la primera línea con la ruta donde se encuentra la carpeta de cocos2d descargada previamente.

**4.- Crear un nuevo proyecto**

Posteriormente a la instalación de las plantillas, se puede empezar un proyecto nuevo desde Xcode:

1. Abrir Xcode
2. File -> New project
3. Seleccionar cualquier plantilla de cocos2d

Se puede ejecutar el proyecto para que ver el ejemplo de “Hola mundo” en el simulador de iPhone. A partir de ahí, es solo cuestión de agregar los recursos “reales” y el código para el juego o demo.

**5.- Creación de un videojuego**

En esta sección se creará un nuevo Proyecto de Cocos2D, se agregarán algunos sprites, haciendo que se muevan en la pantalla o responder a toques de la pantalla.

a) Crear un nuevo proyecto “cocos2d Application” y llamarlo “Ejemplo1”

b) Seleccionar el grupo “Classes” y se pueden ver 7 archivos, corresponden al archivo de definición e implementación de las clases:

* “HelloWorldScene” (Escena inicial)
* “Ejemplo1AppDelegate” (Delegate de nuestra aplicación)
* “RootViewController” (Vista principal donde se mostrará la aplicación)

Además se encuentra un archivo “GameConfig.h” que permite modificar el comportamiento de la aplicación cuando se “rote” el dispositivo (Cambiar de vista landscape a portrait).

c) Seleccionar el archivo “*HelloWorldScene.h*” y agregar las siguientes variables:

CCSprite \*sprite1;

CCSprite \*sprite2;

d) Seleccionar el archivo “*HelloWorldScene.m*”, reemplazar lo que aparece dentrodel if del método init por las siguientes líneas:

CGSize winSize = [[CCDirector sharedDirector] winSize];

sprite1 = [CCSprite spriteWithFile:@”Icon.png”];

sprite1.position = ccp(winSize.width/2, winSize.height/2);

sprite2 = [CCSprite spriteWithFile:@”Icon.png”];

sprite.anchorPoint = ccp(0, 0);

[self addChild:sprite1];

[self addChild:sprite2];

e) Seleccionar icono de “Build and Run” o presionar Command+Enter para compilar y ejecutar en el Simulador la aplicación, la cual debe contener 2 imágenes, una en la esquina inferior izquierda y la otra en el centro de la pantalla. (Si no se han guardado los cambios, Xcode indica si se desea guardar antes de compilar).

A continuación se creará un sprite que se mueva en la pantalla y otro que se mueva a la posición de la pantalla que el usuario presione.

**6. Agregar movimiento a un sprite**

a) Para hacer que un objeto se mueva alrededor de la pantalla se necesita crear un método que ajuste la posición del objeto y sea llamado constantemente, para se realiza lo siguiente:

Método actualizarPosicion

-(void) actualizarPosicion:(ccTime) dt{

sprite1.position = ccp(sprite1.position.x + dt\*100, sprite1.position.y);

if (sprite1.position.x > 480){

sprite1.position = ccp(-32, sprite1.position.y);

}

}

El método anterior mueve el objeto sprite1 a través del eje X y si el objeto llega al final de la pantalla lo coloca al inicio de la pantalla.

Para poder hacer que el método actualizarPosicion se llame constantemente, en el método init, agregar dentro del bloque if, antes del cierre del mismo, lo siguiente:

[self schedule:@selector(actualizarPosicion:)];

Al hacer esto, el metodo actualizarPosicion: se ejecutará cada en cada frame de la animación.

b) Para hacer que un objeto se mueva de una posición a otra, en este caso, a la posición donde el usuario tocó la pantalla basta hacer lo siguiente:

Agregar en el metodo init, delante de donde se agregó la calendarización del método ActualizarPosicion (Dentro del bloque if, antes de cerrarlo).

[[CCTouchDispatcher sharedDispatcher] addTargetedDelegate:self priority:0

swallowsTouched:YES];

Agregar los siguientes métodos:

-(BOOL) ccTouchBegan:(UITouch \*) touch withEvent:(UIEvent \*)event{

return YES;

}

-(BOOL) ccTouchEnded:(UITouch \*) touch withEvent:(UIEvent \*)event{

CGPoint lugar = [self convertTouchToNodeSpace:touch];

[sprite2 stopAllActions];

[sprite2 runAction: [CCMoveTo actionWithDuration:1 position:lugar];

}

El primer método (ccTouchBegan:event:) le indica al touch dispatcher que se desean capturar los toques en la pantalla.

El método convertTouchToNodeSpace: del segundo método agregado (ccTouchEnded:event) hace lo siguiente:

1.- Obtiene la posición del toque.

2.- Usando el CCDirector convierte la posición del toque al sistema de coordenadas GL.

3.- Convierte la posición GL en la posición del espacio del nodo. Posteriormente se detiene cualquier animación que el objeto sprite2 está ejecutando y luego cambia su posición en una acción de un segundo de duración.

Finalmente se selecciona el icono de Build & Run y se debe apreciar que el sprite1 (el que se encontraba en el centro de la pantalla) se moverá por sí mismo y el sprite2 (el que se encuentra en la esquina inferior izquierda) se moverá a en la posición donde se realice un “toque” de pantalla.

**7.- Escenas y Menús**

Una escena es representada por una clase CCScene.

Una escena está ejecutándose si la escena se encuentra visible, cuenta con acciones en progreso. Solo una escena se puede ejecutar a la vez. Sin embargo, es posible meter otra escena en la cima de la actual, pausándola y ejecutando la nueva, posteriormente se puede obtener la actual y continua ejecutándose la primera. También se puede reemplazar por completo una escena.

### Documentación técnica

#### Clases del proyecto

El proyecto se encuentra compuesto de las siguientes clases:

* **Controladores de vistas pertenecientes a UIKit** (UIViewControllers)
  + **RootViewController**: Vista raiz para la escena principal de cocos2d.
    - Detecta si debe rotar o no la vista actual
  + **PuntuacionesViewController**: Vista que permite visualizar las puntuaciones almacenadas por nivel.
    - Consta de un UITableView que despliega las 5 (cinco) puntuaciones más altas de cada nivel.
    - Cada sección corresponde a un nivel del videojuego.
    - Obtiene la información de la base de datos local
  + **AgregarPuntuacionViewController**: Vista que se usa para que el usuario pueda registrar su puntuación una vez finalizado el juego.
    - Solicita la información del jugador y la almacena en la base de datos
* **Capas de cocos2d** (CCLayer)
  + **EscenaPrincipal**: Brinda el menú principal del juego y permite cargar las otras vistas/escenas del juego.
    - Utiliza el objeto *TheAudioCode* para la reproducción de audio
    - Permite cambiar a EscenaConfiguracion, EscenaJuego y *PuntuacionesViewController*
  + **EscenaConfiguracion**: Permite modificar el comportamiento del juego
    - Permite seleccionar el nivel inicial del juego
    - Permite seleccionar el tipo de juego
  + **EscenaJuego**: Es la escena en la que se desarrolla el videojuego
    - Inicialmente muestra las instrucciones del juego
    - Muestra los objetos del juego (imágenes con fracciones) y detecta cual ha sido seleccionado
* **Sprites para manejo de elementos visuales** (CCSprite)
  + **Bola**: Es la clase que almacena la informacion del objeto en el juego
* **Otras clases** (NSObject)
  + **HighScores**: Clase singleton. Permite las administraciones de los records en la aplicación.
  + **GameLogic**: Clase singleton. Administra la lógica del juego.
    - Define los niveles y subniveles.
    - Crear los valores a utilizar en el juego, dependiendo el nivel actual.
    - Valida las respuestas del usuario.
  + **GameDataManagement**: Administra la lectura y escritura de la base de datos
  + **TheAudioCode**: Permite la reproducción de archivos de audio durante el videojuego
    - Implementa SimpleAudioEngine
* **Otros archivos**
  + **GameConfig.h**: Define las constantes así como las estructuras necesarias a utilizar durante el videojuego.

#### Recursos del proyecto

Los recursos (imágenes, audio) que el proyecto necesita son los siguientes:

* Imagenes:
  + Bola\_Nivel\_#.png
    - Imagen del objeto que aparece en el nivel #
  + BackgroundNivel#.plist / BackgroundNivel#.pvr.ccz
    - Imagen de fondo para el nivel #
    - Es un spritesheet creado utilizando TexturePacker
    - Formato de textura: Compressed PVR
    - Formato de imagen: RGB565
* Sonidos:
  + BackgroundSoundNivel#.mp3
    - Sonido de background para el nivel #

### Ciclo de ejecución de la aplicación

- Inicio de la aplicación

* Menú principal: Selección de opción
  + Iniciar juego
    - Se lee el Nivel y Modo de juego actual
    - Se crean los elementos visuales en base al Nivel
    - Se muestra la pantalla de “EscenaJuego” con Instrucciones y se espera selección de opción
      * Iniciar juego
        + Crea Subnivel correspondiente
        + Despliega en pantalla los objetos
        + Muestra la etiqueta con el tiempo
        + Si se presiona Pausa se muestran las instrucciones
        + Si el jugador termina, se muestra la pantalla “SiguienteSubNivel”
        + Se cambia al siguiente subnivel (o nivel, en caso de que sea el último subnivel)
        + Continua el juego o finaliza si es el ultimo subnivel y nivel
        + Si es nuevo record se muestra la vista “AgregarPuntuacion”
        + Se muestra las estadisticas del juego
        + Se regresa al Menú Principal
      * Regresar
        + Regresa al Menú Principal
  + Configuración
    - Se lee la configuración de juego actual
    - Se actualizan los valores correspondientes
    - Se espera selección de opción
      * Aceptar:
        + Guarda configuración actual
        + Regresa a Menu Principal
      * Cancelar:
        + Regresa a Menu Principal
  + Puntuaciones
    - Se lee la base de datos de records
    - Se muestra “PuntuacionesView” con los valores de la base de datos
    - Si se selecciona OK se regresa al Menu Principal
* Fin de ejecución de la aplicación

## Manual técnico de la aplicación “¿Cuánto falta? ¿Cuánto sobra?”

### Introducción

El Objetivo de esta aplicación es reforzar las operaciones básicas con fracciones tales como la suma y resta de fracciones, con una interfaz llamativa para los usuarios, los cuales se encuentran en un grado de estudio equivalente al 5to grado de primaria. La meta en el juego es encontrar una fracción que sumada o restada a la que aplicación otorga se obtenga la unidad. Por ejemplo: en la pregunta ¿Cuánto falta? La tarea será encontrar la fracción que sumada con la fracción proporcionada dará como resultado la unidad. Lo mismo sucedería con la pregunta ¿cuánto sobra?

La aplicación fue desarrollada utilizando el dispositivo HP Palm Pre, con el sistema operativo WebOS, pantalla multitáctil, conectividad WiFi, bluetooth, cámara de 3 mega pixeles y GPS integrado.

### Estructura

Antes de mostrar la estructura de esta aplicación primero se muestra el diseño de un prototipo de las escenas Principales que contendrá la aplicación.

En la figura 1 se observa el escenario principal de la aplicación con tres ventanas principales: la inicial donde el usuario selecciona la acción que desea ejecutar, otra pantalla de ayuda donde se encuentra la explicación de la mecánica del juego y por último la pantalla de inicio de la aplicación.

Ayuda

Explicación

Btn Aceptar

Principal

Ayuda

Inicio

Ejercicio

Fracción

Solución

Btn Calificar

Pregunta

Figura 1: escenario principal

De igual manera, en la figura 2 se muestran los escenarios de las respuestas que brinde el usuario. Los dos escenarios son similares para las respuestas correctas como las incorrectas, al final de cada una de estas pantallas se muestra una pantalla de estadísticas, la cual muestra el número de aciertos, número de errores, y un botón para jugar de nuevo.

Respuesta Correcta

Respuesta Incorrecta

Imagen repuesta Correcta

Etiqueta

Imagen con resultado Correcto

Etiqueta

Estadísticas

Numero de Aciertos

Numero de Errores

Btn Jugar de nuevo

Figura 2:escenarios de respuestas

### Escenas y componentes

Las escenas y los componentes se irán mostrando conforme el usuario se desplace en la aplicación. A continuación se muestra la estructura de estas escenas en el ambiente de programación.

Como se puede observar, la aplicación consta de 4 archivos .js, 3 HTML y un stylesheet además de otros archivos a los que por ahora no haremos referencia.

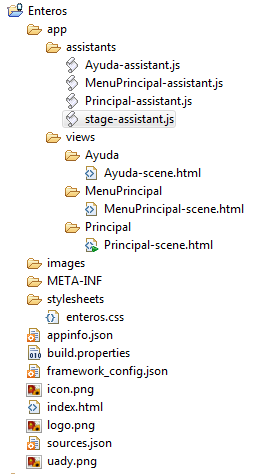


Figura 3: estructura de la aplicación

A continuación se pasará a dar una breve explicación de las clases y funciones incluidas en la aplicación.

### Clases y funciones

Ayuda-Scene.html

<div class="fondo4">

<div class="Leyenda"></div>

<div id="Aceptar"></div>

</div>

En esta plantilla html se crea la parte gráfica de la escena “ayuda”. Se puede observar que se hace referencia a unas clases “fondo4”, y “Leyenda” ambas incluidas en nuestra plantilla de diseño css, además asignamos un id “aceptar” que es el espacio en el que mostraremos el botón aceptar.

Ayuda.js

AyudaAssistant.prototype.setup = **function**() {…};

En la función setup se cargan los componentes que se mostraran al cargar la escena en este caso será el botón aceptar, además es en esta función en la que se le asigna al botón la acción que realizara al ser presionado.

AyudaAssistant.prototype.cambiar = **function**(event){…};

En esta función se hacen los cambios de escena y esto se realia al ser presionado el botón aceptar.

MenúPrincipal-scene.html

<div class="fondo">

<div id="Titulo"></div>

<div id="Iniciar"></div>

<div id="Ayuda"></div>

</div>

Como podemos ver en esta escena se carga la clase “fondo” contenida en la plantilla de diseño css y además se crean tres etiquetas en las que se mostraran el titulo, y dos botones los cuales son iniciar y ayuda.

MenúPrincipal.js

MenuPrincipalAssistant.prototype.setup = **function**() {

…};

En esta función se manda a pantalla los botones iniciar, y ayuda haciendo referencia a las etiquetas del la plantilla html.

MenuPrincipalAssistant.prototype.cambiar = **function**(event){

…

};

En esta función se hacen los cambios de escena hacia la escena principal.

MenuPrincipalAssistant.prototype.cambiar1 = **function**(event){

…

};

En esta escena se hace el cambio a la escena ayuda.

Principal-scene.htm

En esta plantilla igualmente contiene las etiquetas para poder mandar los botones y demás componentes que nos servirán en la escena.

Principal.js

PrincipalAssistant.prototype.setup = **function**(){

…

};

En esta función se cargan todos los componentes que se desplegaran en la pantalla tal como botones, imágenes, etc. Además se asigna la acción que cada botón realizará al ser presionado.

PrincipalAssistant.prototype.generar = **function**(event){

…

};

En esta función se llama a la función que genera los números aleatorios y además valida algunas restricciones a las fraciones por ejemplo que el numerador no sea igual al denominador, etc.

**function** GeneraNumero(){

…

};

En esta función se genera un número entre el rango [1,9]

**function** llenaArreglos(numeradores, denominadores){

…

};

En esta función se agregan los números generados a los arreglos.

**function** HazSuma(numer, denor, nume, deno){

…

};

En esta función se verifica que en el caso en el que se necesite una suma, se verifica que el resultado sea el correcto.

**function** HazResta(numer, denor, nume, deno){

};

En esta función se verifica si el resultado es el correcto.

PrincipalAssistant.prototype.Mensajes = **function**(){

**…**

};

En esta función se manda a la pantalla el mensaje del estado de la respuesta dada (correcto ó Incorrecto).

**function** CalculaRespuesta(){

};

En esta función en caso de que el resultado proporcionado sea incorrecto, calculamos la respuesta correcta.

**function** Aciertos(){

…

};

En esta función se cuentan los aciertos

**function** Errores(){

…

};

En esta función se cuentan los errores.

PrincipalAssistant.prototype.digitospantalla = **function**(){

…};

En esta función se manejan los gráficos, desde esta función se mandan los números a pantalla, esta función maneja la fracción que se va a resolver.

PrincipalAssistant.prototype.digitospantallaresp = **function**(){

**…**

};

En esta función se manejan los gráficos de las fracciones, pero en el área de la respuesta.

PrincipalAssistant.prototype.regresar1 = **function**(event){

};

En esta función es la que utilizamos para realizar la acción del botón “jugar de nuevo”, en esta función se hace el cambio al menú principal

PrincipalAssistant.prototype.Regresar = **function**(event){

…

}

En esta función se realizan las acciones de cambio de graficas entre el mensaje de resultado y el menú de selección de nidos.

En cada una de las funciones siguientes se le asigna un valor, para diferenciar el número de la tarjeta que ha sido seleccionada.

PrincipalAssistant.prototype.Targeta1 = **function**(event){targeta=1; **this**.generar();};

PrincipalAssistant.prototype.Targeta2 = **function**(event){targeta=2; **this**.generar();};

PrincipalAssistant.prototype.Targeta3 = **function**(event){targeta=3; **this**.generar();};

PrincipalAssistant.prototype.Targeta4 = **function**(event){targeta=4; **this**.generar();};

PrincipalAssistant.prototype.Targeta5 = **function**(event){targeta=5; **this**.generar();};

PrincipalAssistant.prototype.Targeta6 = **function**(event){targeta=6; **this**.generar();};

PrincipalAssistant.prototype.Targeta7 = **function**(event){targeta=7; **this**.generar();};

PrincipalAssistant.prototype.Targeta8 = **function**(event){targeta=8; **this**.generar();};

PrincipalAssistant.prototype.Targeta9 = **function**(event){targeta=9; **this**.generar();};

PrincipalAssistant.prototype.Targeta10 = **function**(event){targeta=10; **this**.generar();};

PrincipalAssistant.prototype.Targeta11 = **function**(event){targeta=11; **this**.generar();};

PrincipalAssistant.prototype.Targeta12 = **function**(event){targeta=12; **this**.generar();};

## Manual técnico de la aplicación “Batalla naval”

### Introducción

La Batalla naval o juego de los barquitos es un juego de lápiz y papel tradicional de adivinación que involucra a dos participantes. Se compone de dos tableros por jugador, dividido cada uno en cuadrículas. Los tableros típicos son cuadrados de 5 por 5 casillas, y cada posición se identifica con números para las columnas (de 1 a 5) .En uno de los tableros el jugador coloca sus barcos y registra los tiros del oponente. En el otro, se registran los tiros propios.

Antes de comenzar, cada jugador posiciona los barcos de forma secreta o invisible al oponente, generalmente con el tablero en posición vertical como pizarra. Cada uno ocupa, según su modelo, una cierta cantidad de posiciones, ya sea horizontal o verticalmente. De esta forma, no se permiten lugares solapados, ya que cada uno ocupa posiciones únicas. Ambos participantes poseen y deben ubicar igual número de naves.

Una vez todas las naves han sido posicionadas, se inicia una serie de rondas. En cada ronda, cada jugador en su turno indica una posición del tablero de su oponente. Si esa posición es ocupada por una parte de un barco, el oponente indica destruido y el atacante marca esa posición. Ahora bien, si la posición indicada, efectivamente, no posee un barco alojado, se indica con agua, y será marcada con un pin blanco. Quien descubra primero todas las naves será el vencedor.

### Objetivo

El objetivo de la aplicación móvil es de proporcionarle al estudiante de nivel básico una herramienta de aprendizaje amena con la que pueda aprender y poner en práctica sus conocimientos básicos de matemática en particular al plano cartesiano.

### Restricciones

La principal limitación con la que se tiene en esta aplicación móvil (Midlet) es la que el dispositivo donde instale y donde se ejecute deberá contar con una máquina virtual de java J2ME con una pantalla de tamaño mínimo de 140 x 140 pixeles del dispositivo.

### Funciones Principales

MIDLET.JAVA : En este archivo se encuentra el manejador de pantallas que desempeña el papel fundamental de la aplicación ya que con este se puede crear las pantallas de la aplicación.

En el archivo Midlet.java se encuentran declarados ,el manejador de pantallas ,las pantallas que se usarán (introducción,instrucciones) los botones que utilizará el juego y el reproductor de música.

Constructor Midlet: El constructor se encarga de inicializar los valores de las variables ya mencionadas manejar las pantallas cargar la música de fondo y de insertar los botones.

Parámetros de entrada: Ninguno.

Parámetros de salida: Ninguno.

Excepción: Genera una excepción sino se encuentra el archivo de sonido de fondo.

Función startApp: Se encarga de poner en marcha el Midlet.

Parámetros de entrada: Ninguno.

Parámetros de salida: Ninguno.

Excepción: Ninguna

Función destroyApp: Se encarga de destruir el Midlet.

Parámetros de entrada: Ninguno.

Parámetros de salida: Ninguno.

Excepción: Ninguna

Función commandAction: Se encarga de verificar que botones fueron presionados.

Parámetros de entrada: Ninguno.

Parámetros de salida: Ninguno.

Excepción: Ninguna

MIDPIntro.JAVA : En este archivo se realiza la función de crear la pantalla de introducción cuenta con la cadena que se escribirá de mensaje de bienvenida , con las coordenadas donde será escrito dicho texto y su correspondiente Fuente de texto.

Constructor MIDPIntro: El constructor se encarga de inicializar los valores de los tipos de fuentes que utilizará el texto de introducción.

Parámetros de entrada: Ninguno.

Parámetros de salida: Ninguno.

Excepción: Ninguna.

Función paint:

Este método se encargará de posicionarse en una ubicación adecuada para dibujar el texto utilizando la fuente de este.

Parámetros de entrada: Tipo de Dato Graphics que utlizará para pintar en la pantalla.

Parámetros de salida: Ninguno.

Excepción: Ninguna.

MIDPIntrucciones.JAVA: En este archivo se realiza la función de crear la pantalla de intrucciones, cuenta con la cadena que se escribirá de mensaje de bienvenida , con las coordenadas donde será escrito dicho texto y su correspondiente Fuente de texto y además escribirá las instrucciones de cómo se utiliza el Midlet.

Constructor MIDPIntrucciones:

El constructor se encarga de inicializar los valores de los tipos de fuentes que utilizará el texto de introducción.

Parámetros de entrada: Ninguno.

Parámetros de salida: Ninguno.

Excepción: Ninguna.

Función paint:

Este método se encargará de posicionarse en una ubicación adecuada para dibujar el texto utilizando la fuente de este.

Parámetros de entrada: Tipo de Dato Graphics que utlizará para pintar en la pantalla.

Parámetros de salida: Ninguno.

Excepción: Ninguna.

MIDPCanvas.JAVA: Este archivo almacena las funciones para el funcionamiento del movimiento de los tanques con las teclas del dispositivo ,se encarga de pintar la pantalla principal en general.

Constructor MIDPCanvas: El constructor se encarga de inicializar las imágenes de los tanques y la mira que se utilizará en el juego tambien se encarga de inicializar las banderas del juego.

Parámetros de entrada: Ninguno.

Parámetros de salida: Ninguno.

Excepción: Produce excepción si alguna de las imágenes no son cargadas apropiadamente.

Función paint:

El método esta encargada de las principales funciones como son dibujar totalmente la pantalla principal del juego verifica los tanques asignados hasta el momento el límite de tanques que deberán ponerse sobre el mapa verifica de igual manera si ya puede comenzar los ataques y saber si fueron exitosos o no.

Parámetros de entrada: Tipo de Dato Graphics que utlizará para pintar en la pantalla.

Parámetros de salida: Ninguno.

Excepción: Ninguna.

Función leDiste: Método para determinar si fué un ataque exitoso o no, comparando las coordenadas donde fueron efectuados los disparos con las pocisiones donde se encuentra cada tanque guardado en los arreglos de posiciones.

Parámetros de entrada: Recibe la coordenada X \_x y la coordenada Y \_y.

Parámetros de salida: Devuelve true si fué exitoso el ataque,de otro modo devuelve False.

Excepción: Ninguna.

Función coorX: Método que sirve para determinar y pintar que coordenada está.

Parámetros de entrada: Recibe como parametro un entero \_x

Parámetros de salida: Devuelve un entero que será la coordenada que aparecerá en la pantalla.

Excepción: Ninguna.

Función coorY:

Método que sirve para determinar y pintar que coordenada está.

Parámetros de entrada: Recibe como parametro un entero \_Y

Parámetros de salida: Devuelve un entero que será la coordenada que aparecerá en la pantalla.

Excepción: Ninguna.

Función ciudad: Método que se encarga de determinar mendiante cuadrante en que estado está posicionado el cursor.

Parámetros de entrada: Recibe dos int ,int \_x ,int \_y que son las coordenadas actuales que comparara.

Parámetros de salida: Devuelve un string en el que envia el nombre del estado en el que está pocisionado.

Excepción: Ninguna.

Función keyPressed:

Método que determinar la dirección y movimiento del cursor por cada tecla pulsada en el dispositivo.

Parámetros de entrada: Recibe int keyCode que es el número de tecla pulsada.

Parámetros de salida: No devuelve nada.

Excepción: Ninguna.

Función keyRepeated: Método que determinar la dirección y movimiento del cursor por cada

tecla que se mantiene pulsada en el dispositivo.

Parámetros de entrada: Recibe int keyCode que es el número de tecla pulsada.

Parámetros de salida: No devuelve nada.

Excepción: Ninguna.

# Manual de Usuario

## Manual de usuarios de la aplicación “Comparando longitudes”

Este manual utiliza el emulador que trae NetBeans

La pantalla inicial de la aplicación es la siguiente:



Presionar cualquier tecla del emulador para continuar



Botón central de selección.

Flechas de navegación

Después de presionar se llega a la siguiente pantalla que contiene el menú principal.

### Opción Jugar



Se utilizan las teclas de desplazamiento para moverse a través del menú principal, para seleccionar una opción se presiona el botón central del emulador. El comportamiento de la aplicación es similar en un dispositivo real, para aquellos dispositivos que son de tipo Touch, el sistema proporciona un teclado virtual para que sea accesible desde cualquier teléfono que soporte Java.

Al seleccionar la opción Jugar se ve la siguiente pantalla:



Para voltear las cartas del memorama primero se mueve a la localidad deseada usando las flechas de navegación y después se presiona el botón de selección.



Conforme se vayan descubriendo los pares de cartas estos van a ir despareciendo del tablero.

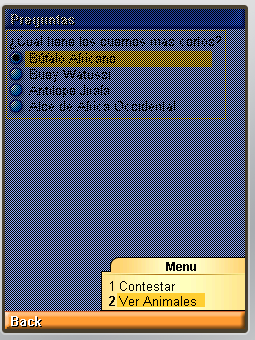
Cuando se tenga el tablero completado se pueda pasar a la opción de preguntas.



Seleccionar la opción de continuar usando la tecla que se encuentra debajo de la palabra continuar.



En esta sección se puede presionar el Menú que muestra las opciones:



Contestar: Permite establecer la respuesta que se considere correcta y muestra un mensaje indicando la evaluación correspondiente.

Ver Animales: Permite ver imágenes de los animales en caso que el usuario no recuerde las medidas que hacen referencia a las preguntas.



### Opción Instrucciones

Al seleccionar esta opción aparece la siguiente pantalla.



La opción de Salir, simplemente termina la aplicación regresando al Menú de las aplicaciones disponibles en el teléfono celular.

## Manual de usuario de la aplicación “Ordenando fracciones”

En la pantalla inicial, se selecciona la opción Iniciar Juego que presenta la siguiente pantalla.



La aplicación es completamente táctil, por lo que para seleccionar cualquier opción basta con tocar la pantalla sobre la palabra que hace referencia a la opción.

### Primer nivel

Para ir despareciendo las imágenes que descienden durante el desarrollo del juego, basta con tocarlas sobre cada objeto. Desparecerán si se presionó de manera correcta.

El objetivo de la aplicación desarrollada es que el alumno ordene una serie de fracciones de menor a mayor. Como ejemplo podemos tener siguientes fracciones:

* , ,, ,

El ordenamiento resultante debe ser:

* , ,, ,

Entonces en ese orden deben irse presionando en la los objetos descendentes.



El objetivo es desapareciendo los objetos ordenadamente. Cualquier sección del juego puede ser pausada seleccionando la palabra Pausa que aparece en el lado superior derecho de la pantalla. En el primer nivel el ordenamiento es más sencillo porque se conserva el mismo denominador. En la pantalla también se puede apreciar un contador de segundos que mide el tiempo jugado.

El primer nivel aparece 3 veces para que pueda pasar al siguiente nivel.

### Segundo nivel

La pantalla del segundo nivel es la siguiente:



La mecánica es la misma, lo único que cambia es que se tienen 2 distintos denominadores en las fracciones que van descendiendo. Cuando desaparecen todas las gotas de este nivel es posible pasar al último nivel de la aplicación. Se presenta 3 veces en total antes de pasar al siguiente escenario.

### Tercer nivel

El último nivel contiene la siguiente pantalla.



La lógica de funcionamiento es la misma y el incremento en la dificultad es que los denominadores son variados, pero se deben ir desapareciendo tocando ordenadamente cada pingüino, en caso de error en el orden la aplicación lo indicia con un pequeño mensaje que aparece en el lado inferior y el objeto no desaparece.

Después de jugar 3 veces esta última etapa aparece la pantalla que muestra las estadísticas.



La opción Regresar, permite iniciar la aplicación desde cualquier nivel ya que dirige hacia el menú principal.

## Manual de usuario de la aplicación “¿Cuánto falta? ¿Cuánto sobra?”

### Instalación

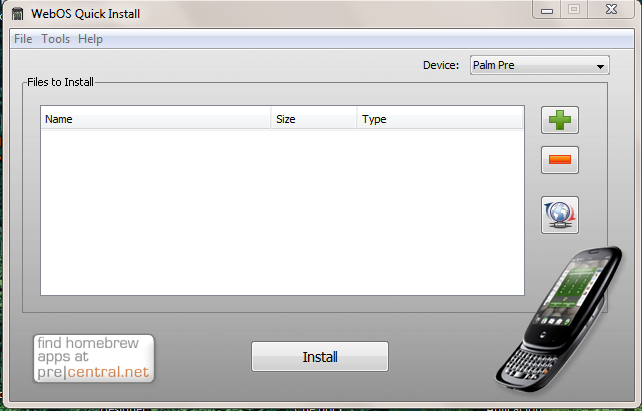
Antes de la instalación se deberá configurar el dispositivo en “Modo Desarrollador” (Developer Mode). Para ello, en la vista de cartas, escribe “webos20090606″. Esto iniciará una Búsqueda Universal, y aparecerá un icono con título “Developer Mode Enabler” (Figura :



Una vez dentro de la aplicación, se activa el “Developer Mode”. Se pedirá reiniciar el equipo, por lo que se deberá hacer. Una vez que se haya terminado de instalar la aplicaci¡ón, se puede repetir estas mismas instrucciones para desactivar el Modo Desarrollador.

Para la instalación de la aplicación se puede utilizar algunas herramientas ya existentes para esto como por ejemplo la que podemos encontrar en el siguiente enlace: <http://www.webosmania.com/webos/webos-quick-install-se-actualiza/>

En este link podemos encontrar una aplicación llamada WebOS Quick Install desarrollada por Jason Robitaille, esta aplicación cuenta con la siguiente interfaz:

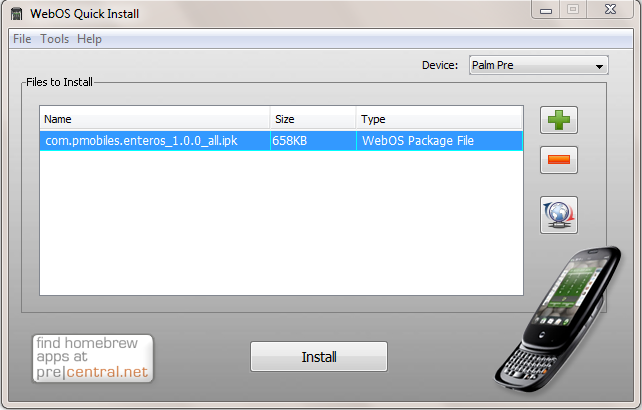


Agregar Archivo

Para instalar una aplicación hacemos clic en el símbolo de “+” seguido de esto se abrirá una ventana por medio de la cual podremos navegar para encontrar nuestra aplicación,



Una vez encontrada la aplicación hacemos clic en select, una vez agregado el archivo hacemos click en install, y ya tendremos instalada nuestra aplicación en nuestro dispositivo.

****

Botón para instalar

### Ejecución

Para poder ejecutar el juego con el dispositivo encendido seleccionamos el icono con la flecha dentro como se muestra en la figura, esto es para poder ingresar al menú principal del dispositivo.



Icono para ingresar al menú Principal

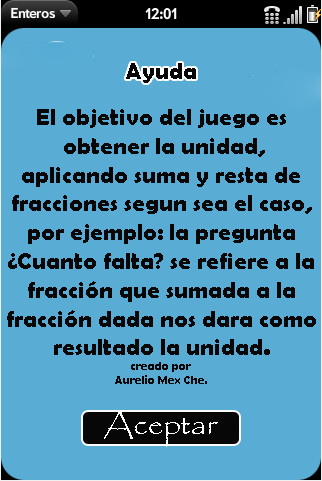


Icono del juego

El siguiente Paso es buscar el icono del juego(es el que se indica en la figura) una vez identificado el icono lo seleccionamos, con esto el juego se abrirá como se muestra enseguida.

En la primera pantalla del juego tenemos dos opciones consultar la ayuda al juego seleccionando el icono con la forma del signo de interrogación ubicado en la parte inferior derecha de la pantalla, además también podemos ver el botón iniciar ambos se enseñan en las siguientes figuras.

Al seleccionar el botón ayuda se mostrará la pantalla que se enseña a continuación, esta pantalla únicamente cuenta con una pequeña explicación acerca del objetivo del juego, y un botón el cual nos lleva nuevamente a la pantalla principal.

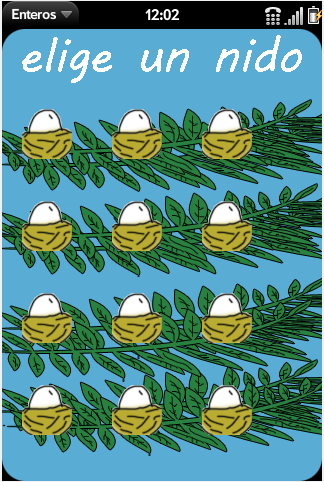




Botón iniciazr

Botón Ayuda

Al Seleccionar el botón iniciar en el menú principal nos mandará a la siguiente pantalla, en esta se muestran 12 nidos, los cuales pueden elegirse uno por uno en el orden que se prefiera.



Al seleccionar alguno de los nidos nos mandara a la pantalla mostrada en la siguiente imagen, la cual consta de una pregunta, una fracción, el área de respuesta y el botón para validar la respuesta proporcionada.



Pregunta

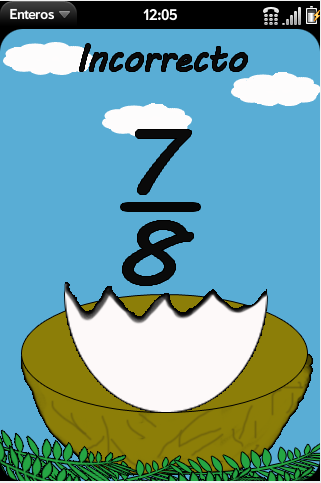
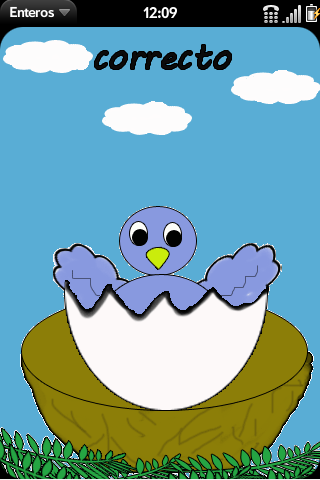
Boton para seleccionar el numerador

Fraccion

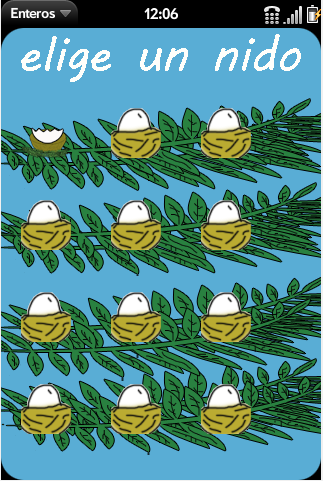
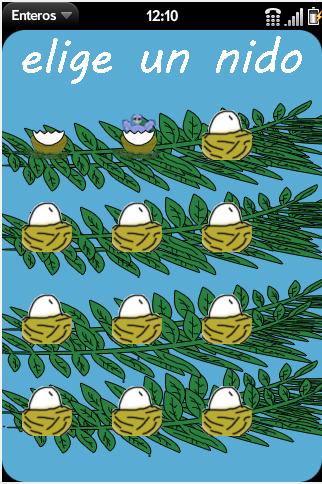
Boton para seleccionar el denominador

Botón para validar

Si el resultado dado es incorrecto al validarlo nos enseñará la pantalla que se muestra en la siguiente figura en la cual se mostrará la respuesta correcta del ejercicio, solo basta con tocar cualquier parte de la pantalla para regresar a la pantalla donde podremos elegir otro nido.



Si la respuesta fue incorrecta, en el menú de selección nos mostrará un icono como el que se muestra en la siguiente figura. En caso que la respuesta sea correcta se mostrará la pantalla que se siguiente, basta con tocar cualquier parte de la pantalla para regresar a la pantalla donde se puede seleccionar otro nido.



Significa que la respuesta que dimos en ese ejercicio fue incorrecta

Respuesta Correcta

Al terminar los doce ejercicios se mostrarán las estadísticas, tal y como se observa en la siguiente imagen, esta pantalla cuenta con un botón con el cual se puede regresar al menú principal. Para salir del juego basta con oprimir el botón que trae el dispositivo en el centro de parte inferior del dispositivo y lanzar la aplicación hacia arriba (esto se puede hacer en cualquier momento del juego).



Botón Regresar

## Manual de usuario de la aplicación “Batalla Naval”

### Ejecución

1.- El objetivo es destruir todos los tanques enemigos antes de perder los 5 tanques propios.

2.- Primero se deberá ejecutar desde el dispositivo móvil el juego Warfare.

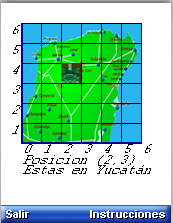
3.- Una vez iniciada la aplicación se verá una pantalla de bienvenida como la que

se muestra en la imagen**:**

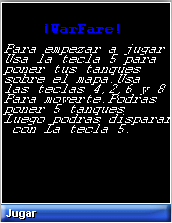
****

4.-Se pulsa el boton correspondiente “Jugar” para empezar a jugar.

5.- En la pantalla principal si es la primera vez que se usa la aplicación existe un botón en el menú que contiene las instrucciones del juego llamado “Instrucciones” ,como se muestra acontinuación:

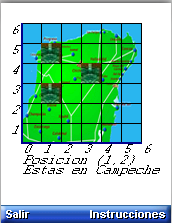


6.- Una vez pulsado el botón de instrucciones estas aparecerán en la pantalla indicando que teclas del dispositivo utilizará para asignar y posicionar los tanques e iniciar los ataques.

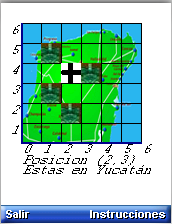


7.- Una vez terminadas de leer las instrucciones se pulsa el boton “Jugar” para volver ala pantalla principal.

8.-Nuevamente se posicionan los tanques como se desee .



9.- Una vez asignados todos los tanques se procederá a iniciar los ataques utilizando la mira dentro del mapa se atacará al enemigo.



10.- Si al atacar se acierta con el enemigo aparecerá una ventana de aviso indicando si el ataque fue exitoso o no,como se muestra en las imágenes,las imágenes corresponden a si el ataque fue exitoso o no,respectivamente.

