

**LiveCode para Android
a través de ejemplos:
nivel básico**

José Manuel Piña Gutiérrez

Rector

LiveCode para Android a través de ejemplos: nivel básico

Dulce Marisol Zavaleta Luna
Julián Javier Francisco León



**UNIVERSIDAD JUÁREZ
AUTÓNOMA DE TABASCO**

Primera edición, 2016

D.R. © Universidad Juárez Autónoma de Tabasco
Av. Universidad s/n. Zona de la Cultura
Colonia Magisterial, C.P. 86040
Villahermosa, Centro, Tabasco.

El contenido de la presente obra es responsabilidad exclusiva de los autores. Queda prohibida su reproducción total sin contar previamente con la autorización expresa y por escrito del titular, en términos de la Ley Federal de Derechos de Autor. Se autoriza su reproducción parcial siempre y cuando se cite la fuente.

Marcas registradas

LiveCode es una marca registrada de LiveCode Ltd., en Europa.
Java y todas las marcas basadas en Java son marcas registradas de Sun Microsystems, Inc., en los Estados Unidos y otros países.
El robot de Android es reproducido de realizado y compartido por Google Inc. y está siendo utilizado de acuerdo a los términos descritos en la licencia Creative Commons 3.0

ISBN: 978-607-606-407-8

Edición y corrección: Francisco Morales Hoil
Calíope Bastar Dorantes

Diseño y formación: Ricardo Cámara Córdova

Hecho en Villahermosa, Tabasco, México

Índice

PREFACIO	7
Contenido del material educativo	7
A quién va dirigido	8
Convenciones utilizadas	8
INTRODUCCIÓN A LIVECODE	11
¿Qué es LiveCode?	11
Estructura en LiveCode	11
Sintaxis en LiveCode	13
INSTALACIÓN	15
Requisitos de instalación	15
Instalación de herramientas de desarrollo	15
LiveCode	15
Instalación de herramientas de soporte	16
Java SDK	16
Android SDK	16
PRÁCTICAS	21
Práctica 1. Hola Mundo	21
Práctica 2. Textos	26
Práctica 3. Cálculos	29
Práctica 4. Uso de funciones y comandos	31
Práctica 5. Utilización de imágenes	34
Práctica 6. Transferir aplicaciones desarrolladas en LiveCode a un dispositivo Android	38
Práctica 7. Archivos multimedia	40
Aplicación que reproduce un video	40
Aplicación que reproduce un audio	42

Práctica 8. Creación de interfaz gráfica personalizada	43
Práctica 9. Implementación de código en interfaz gráfica personalizada	48
Práctica 10. Aplicaciones ajustables al tamaño de pantalla	51
Uso de la propiedad fullscreenmode	51
Escalar los elementos del stack por código	53
Práctica 11. Animación de elementos de una aplicación	58
Práctica 12. Manejo de notificaciones locales	69
establecerHora	73
crearRecordatorio	73
localNotificationReceived	74
preOpenStack (opcional)	75
DICCIONARIO DE LIVECODE	77
TIPS PARA NOMBRAR VARIABLES	81
CONCLUSIÓN	83
BIBLIOGRAFÍA	85

Prefacio

LiveCode es un entorno de desarrollo y lenguaje de programación que se caracteriza por el alto nivel que presenta y las múltiples plataformas de desarrollo que ofrece.

Actualmente, ya no es necesario aprender lenguajes especializados para desarrollar en cada plataforma. Más bien, es necesario saber qué herramientas permiten construir software en diferentes sistemas operativos y conocer las distintas novedades que ofrecen para cada plataforma.

El presente material educativo se encarga de presentarle al usuario cómo programar en LiveCode, haciendo énfasis en las herramientas que ofrece para la plataforma Android.

Contenido del material educativo

Este material educativo se divide en tres secciones: Introducción, Instalación y Prácticas.

Introducción. Describe los términos que el usuario necesita conocer para desarrollar aplicaciones en LiveCode. Explica, a grandes rasgos, cómo se estructura una aplicación en este entorno y la sintaxis que maneja.

Instalación. Indica cómo instalar las herramientas que se utilizarán para desarrollar aplicaciones Android en LiveCode.

Prácticas. Esta sección se conforma de 12 prácticas. Éstas muestran al usuario los pasos para desarrollar el producto final de cada una. En total se obtendrán 10 aplicaciones; una de ellas requiere tres prácticas, que son las siguientes:

Prefacio

1. Hola Mundo. Primera aplicación para explicar la estructura de las aplicaciones en Android.
2. Textos. Aplicación que realiza preguntas al usuario y muestra las respuestas en pantalla.
3. Cálculos. Calcula la cantidad aproximada de días vividos por el usuario.
4. Funciones y comandos. Muestra la medida en pulgadas del usuario.
5. Imágenes. Cambia la imagen mostrada conforme al botón pulsado.
6. Videos. Reproduce un video.
7. Audios. Reproduce un audio.
8. Interfaz videos. Reproductor de video con interfaz gráfica propia, adaptable al tamaño de la pantalla. Se realiza en las prácticas 8, 9 y 10.
9. Animaciones. Aplicación que anima una imagen conforme al botón pulsado.
10. Recordatorios. Crea una notificación local en el dispositivo Android.

Las aplicaciones del 1 a 5 son ejecutadas en LiveCode y de la 6 a 10 son probadas en un dispositivo Android. Cabe destacar que los ejemplos expuestos en este material educativo fueron probados en las versiones 6.6.2 y 7.0.1 de LiveCode bajo Windows 8.1.

A quién va dirigido

Este material educativo va dirigido a cualquier persona que desee aprender a programar en LiveCode, sea programador o no. Así mismo, va dirigido a las personas que están familiarizadas con cualquier lenguaje de programación. En este material educativo se abordan los conceptos básicos del lenguaje LiveCode; sin embargo, no se utilizan términos dirigidos solamente a programadores, sino a todo público. No obstante, se requiere lógica y análisis para la realización de cada práctica.

Convenciones utilizadas

El material educativo “LiveCode para Android a través de ejemplos: nivel básico” se compone de 12 prácticas. Para facilitar el entendimiento de las mismas, se han establecido las siguientes convenciones:

Convenciones utilizadas

El material educativo “LiveCode para Android a través de ejemplos: nivel básico” se compone de 12 prácticas. Para facilitar el entendimiento de las mismas, se han establecido las siguientes convenciones:

- **Resaltado de códigos.** Los códigos utilizados en las prácticas son señalados como figuras y resaltados en rectángulos grises. El número de líneas del código se resalta en negritas.
- **Uso de fuentes para dar énfasis.** Para facilitar la identificación de los términos, se utilizan fuentes y estilos de texto distintos. Estos son los siguientes:
 - **Caecilia Negritas.** Resalta los elementos en pantalla que son mencionados.
 - **Courier New.** Fuente general utilizada en el código de las aplicaciones.
 - **Cursiva.** Indican las propiedades de los objetos en LiveCode.
 - **Negritas.** Señalan las palabras del diccionario de LiveCode.
 - **“Palabras entre comillas”.** Se refieren a los nombres asignados a las variables, *handlers* y *stacks* que se utilizan en las aplicaciones.
 - **Cursivas.** Resalta los sustantivos utilizados para referirse a los elementos de LiveCode.
 - **[Corchetes].** Indican las notas del autor.
- **Nivel de dificultad.** Cada práctica establece un nivel de dificultad, mismo que se clasifica de la siguiente manera:
 - **Muy fácil.** La forma de elaborar la aplicación se explica a detalle, de forma que, si el usuario sigue las instrucciones al pie de la letra, la aplicación final será idéntica a la mostrada en la práctica.
 - **Fácil.** Requiere que el usuario recuerde conceptos vistos en prácticas anteriores, pero los pasos son explicados a detalle para que la aplicación final sea idéntica.
 - **Media.** El usuario debe recordar conceptos anteriores y explorar la interfaz de la herramienta para que la aplicación sea similar a la especificada en la práctica. Estas prácticas implican razonamiento y creatividad por parte del usuario.

Introducción a Live Code

¿Qué es LiveCode?

LiveCode es un entorno y lenguaje de programación con soporte para la creación de aplicaciones móviles, de escritorio y servidores, utilizando una sola plataforma de trabajo. Los sistemas operativos que maneja LiveCode son Android, iOS, MacOS, Windows y Linux. Así mismo, permite el desarrollo de sitios web de lado cliente y servidor. Para mayor información, visita la siguiente referencia: <https://livecode.com/resources/guides/beginners-guide/>

Estructura en LiveCode

LiveCode funciona utilizando un conjunto de tarjetas que están asociadas a una pila. Esta pila de tarjetas es el elemento fundamental de la estructura de LiveCode. Los elementos que conforman la estructura de un programa en LiveCode son los siguientes (LiveCode Ltd., 2014a):

Stack (Pila). Es el elemento principal en LiveCode, ningún otro elemento puede existir sin un *stack*. Se puede entender coloquialmente como las “ventanas” de LiveCode, ejemplos de éstas son: paletas, cuadros de diálogo y ventanas estándar.

Card (Tarjeta). Es el segundo elemento fundamental en LiveCode. Se caracteriza por mostrar la información y los objetos que el usuario coloca en ella, de modo que ésta será diferente dependiendo de los datos que contenga. Por la manera en que LiveCode se organiza, cada pila debe tener al menos una *card*. Esto implica que cuando un *stack* es creado, éste ya la incluye. Aunque un *stack* puede contener múltiples *cards*, sólo es visible una a la vez, por lo que también son conocidas como “pantallas”.

Un ejemplo para explicar dónde se encuentran las *cards* en un programa, es en un videojuego. Generalmente, en un videojuego hay una pantalla de inicio, la cual presenta las siguientes opciones: juego nuevo, continuar partida y salir. Al seleccionar la opción “juego nuevo”, el programa lleva al usuario a otro apartado donde el juego comienza. No obstante, si el usuario selecciona la opción “continuar partida”, el juego muestra otra interfaz para cargar las partidas salvadas en sesiones anteriores. En LiveCode, las pantallas de juego nuevo, inicio y cargar partidas serían tres *cards*¹ distintas.

Objetos. LiveCode es un lenguaje de programación basado en objetos. Por tanto, es imprescindible la existencia de un objeto en un programa. RunRev Ltd (2014a) especifica que los tipos de objetos que pueden ser agregados son:

- **Campos.** Para mostrar o ingresar datos.
- **Botones.** Para crear eventos.
- **Barras de desplazamiento.** Para establecer una cantidad.
- **Gráficos.** Básicos o dibujados a mano.
- **Imágenes.** Normalmente fotos.
- **Reproductores.** Para videos, audio, etc.

Scripts. Indica al objeto qué hacer cuando interactúa con el usuario. El *script* se encuentra organizado en secciones de código y técnicamente se considera como una propiedad de un objeto (LiveCode Ltd., 2010).

Un *handler*, o en español *controlador*, es una sección de código. Cada *handler* puede ser ejecutado de manera independiente. Existen cuatro tipos de *handlers* (LiveCode Ltd., 2010):

Commands or message handlers (Controladores de comandos o controladores de mensajes). Inician con la estructura de control **on** seguido con el nombre del mensaje al que responde el *handler* (ver Figura 1):

```
1 on mouseUp
2     answer "Hola Mundo"
3 end mouseUp
```

Figura 1. Ejemplo de *message handler*

¹ A partir de las prácticas se utilizará el termino tarjetas en lugar de *cards*. Éste es el único termino que se traduce del inglés al español por facilidad de uso para los autores.

Este comando responde al mensaje “mouseUp”. Cada mensaje es enviado por un evento en LiveCode. En este caso, cuando el cursor es levantado del objeto que contiene el *script*, éste envía el mensaje “mouseUp”.

En resumen, los *command handlers* o *message handlers* son secciones de código que indican operaciones que el objeto debe realizar cuando un evento sucede.

Functions (Funciones). Son *handlers* que devuelven algún valor en específico; normalmente son llamados por otro *handler* (ver Figura 2).

```
1 function calcularEstatura tEstatura
2   return (tEstatura*100)/2.54
3 end calcularEstatura
```

Figura 2. Ejemplo de función

getProp handlers. Se utilizan cuando una propiedad personalizada de un objeto es requerida por una declaración de LiveCode.

setProp handlers. Cambian el valor de las propiedades de un objeto.

Sintaxis en LiveCode. La sintaxis en LiveCode está delimitada por su plataforma. Ésta consiste en el uso de un lenguaje de alto nivel similar al inglés natural. Sin embargo, por el momento sólo se encuentra limitado al procesamiento de datos y texto. Extensiones de terceros, como la creación de bases de datos y acciones correspondientes a móviles, utilizan la sintaxis tradicional de llamadas a funciones y comandos (Livecode Blog, 2013).

Un ejemplo de la sintaxis de LiveCode, es el uso de las palabras *ask*, *answer* y *put*. Éstas se traducen del inglés como: preguntar, responder y colocar, respectivamente. Sin embargo, cada palabra corresponde a un comando. En la Práctica 2 se explicará a detalle el funcionamiento de dichos comandos.

Instalación

Requisitos de instalación

- Windows XP, Vista, 7, 8 y 8.1²
- 2GB de memoria RAM.

Instalación de herramientas de desarrollo

LiveCode

1. Visite el sitio web livecode.com/download/ y haga clic en el botón **Download LiveCode Only** para ir a la página de descargas.
2. Elija una versión de LiveCode. El sitio mostrará una lista de versiones que se clasifican en **Community Edition**, **Indy Edition**, **Business Edition** y **Comunity Server**. Se recomienda instalar **Community Edition** con la leyenda **STABLE** para evitar errores. En los ejemplos de este material educativo se utilizó LiveCode Community Edition 7.0.1.
3. Guarde el archivo en la computadora y al terminar la descarga, ejecútelo.
4. Siga los pasos de instalación del programa. Haga clic en **continue**. Recuerde aceptar los términos y condiciones del programa.
5. Seleccione el tipo de instalación que desea utilizar, se recomienda hacer clic en **you only**.
6. Haga clic en **Install**.
7. Al terminar la instalación active la casilla **Launch LiveCode**.

Para más información sobre cómo instalar la versión más reciente de Android SDK en LiveCode, consultar la siguiente dirección:
<http://livecode.com/app-in-a-day/installation/android/>

² LiveCode también se encuentra disponible para MacOs y Linux. No se mencionan dichos sistemas operativos debido a que la guía de instalación sólo describe el procedimiento utilizado en Windows. Los pasos descritos a continuación pueden variar conforme al sistema operativo.

Instalación

Requisitos de instalación

- Windows XP, Vista, 7, 8 y 8.1²
- 2GB de memoria RAM.

Instalación de herramientas de desarrollo

LiveCode

1. Visite el sitio web livecode.com/download/ y haga clic en el botón **Download LiveCode Only** para ir a la página de descargas.
2. Elija una versión de LiveCode. El sitio mostrará una lista de versiones que se clasifican en **Community Edition**, **Indy Edition**, **Business Edition** y **Comunity Server**. Se recomienda instalar **Community Edition** con la leyenda **STABLE** para evitar errores. En los ejemplos de este material educativo se utilizó LiveCode Community Edition 7.0.1.
3. Guarde el archivo en la computadora y al terminar la descarga, ejecútelo.
4. Siga los pasos de instalación del programa. Haga clic en **continue**. Recuerde aceptar los términos y condiciones del programa.
5. Seleccione el tipo de instalación que desea utilizar, se recomienda hacer clic en **you only**.
6. Haga clic en **Install**.
7. Al terminar la instalación active la casilla **Launch LiveCode**.

² LiveCode también se encuentra disponible para MacOs y Linux. No se mencionan dichos sistemas operativos debido a que la guía de instalación sólo describe el procedimiento utilizado en Windows. Los pasos descritos a continuación pueden variar conforme al sistema operativo.

Instalación de herramientas de soporte

Java SDK

1. Descargue el SDK de Java. Éste se encuentra disponible en el sitio de descargas de Oracle: oracle.com/technetwork/java/javase/downloads/index.html. Una vez abierto el sitio, seleccione **JDK**.
2. Acepte la licencia de Oracle y elija **Windows x86**.
3. Guarde el archivo en el sistema, ejecútelo y siga los pasos que indica el instalador. Si ya existe JDK de Java en el equipo, la aplicación le preguntará si desea reinstalar el software.
4. Haga clic en **siguiente** durante toda la instalación. Durante el proceso, el SDK de Java permite especificar la ruta y los componentes a instalar. Sin embargo, LiveCode no necesita de ninguna configuración en específico y funciona con las configuraciones por defecto.

Android SDK

1. Descargue el SDK de Android desde el siguiente link: developer.android.com/intl/es/sdk/index.html. En el apartado **Other Download Options**, seleccione y guarde el SDK de la plataforma **Windows**.
2. Ejecute el archivo y siga los pasos de la instalación. Si al abrir el instalador no le permite continuar, quiere decir que el JDK no fue instalado apropiadamente.
3. Haga clic en **siguiente**, hasta llegar al final de la instalación.
4. Active la casilla para ejecutar el AVD Manager y el SDK de Android.
5. Finalice la instalación.
6. Seleccione las versiones 2.2, 2.3.3 y 4.1 en el **SDK Manager**.
7. Instale las versiones seleccionadas. Haga clic en **Install Package** y acepte los términos y condiciones. Los archivos necesarios comenzarán a descargarse. [Nota: El tiempo de descarga puede variar entre 30 minutos y 3 horas, dependiendo la velocidad de su conexión a Internet].
8. Configure el dispositivo para probar las aplicaciones desarrolladas en Android. Una vez terminada la descarga, puede configurar un dispositivo virtual o físico. A continuación, se explica la configuración de cada alternativa:

Configurar un dispositivo virtual

Para ello se utilizará *Android AVD Manager*. Siga los siguientes pasos:

1. Ejecute AVD Manager y seleccione **New**.
2. Establezca un nombre para su dispositivo virtual y el tamaño de pantalla que tendrá. Dependiendo del tamaño, el sistema considerará al dispositivo como un móvil o tableta. Un móvil promedio tiene una pantalla de 4 pulgadas.
3. Establezca como **target** la versión de Android que tendrá su dispositivo virtual. Se recomienda utilizar la versión 2.3.3.
4. Escriba la capacidad de almacenamiento que desea que tenga la tarjeta SD. Es recomendable utilizar la configuración indicada en la Figura 4.

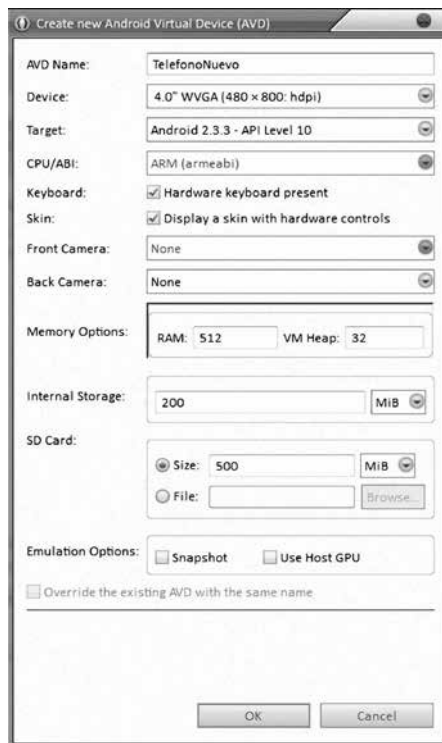


Figura 4. Creación de un dispositivo virtual

Instalación

5. De manera opcional, puede activar la casilla **Snapshot**. Su función es acelerar la emulación de Android en la computadora.
6. Haga clic en **Ok** para crear el dispositivo virtual.

Configurar un dispositivo físico

En el caso de un dispositivo físico, es necesario contar con móvil o tableta Android e instalar el driver correspondiente. Para ello, puede obtener más información de la página Android Developers en developer.android.com/intl/es/sdk/win-usb.html, o seguir los siguientes pasos:

1. Descargue Adb Driver Installer de adbdriver.com.
2. Ejecute el archivo.
3. Conecte el dispositivo a la computadora y active el modo *Depuración USB*, que se encuentra en las opciones del programador dentro de los ajustes del dispositivo.
4. Cuando **Adb Driver** detecte el dispositivo, haga clic en **Install** (ver Figura 5).

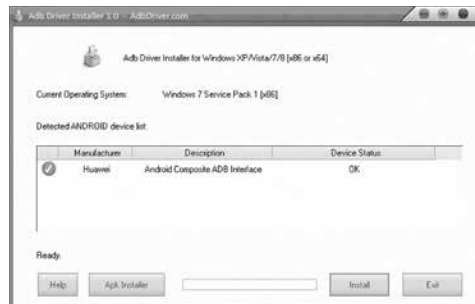


Figura 5. Adb Driver Installer

5. Abra LiveCode y seleccione el menú **Edit > Preferences**.
6. Despliegue la sección **Mobile Support**.
7. Establezca el directorio del SDK de Android. La carpeta debe ser similar a la siguiente: **C:/ProgramFiles/Android/android-sdk**. Si ha instalado Android con anterioridad, busque entre las carpetas descargadas aquella con el nombre **SDK**. El resultado del procedimiento se observa en la Figura 6.



Figura 6. Añadiendo la carpeta del SDK de Android.

Prácticas

Practica 1. Hola Mundo

Objetivo: Comprender la estructura de un programa en LiveCode mediante el desarrollo del ejemplo clásico: “Hola Mundo”.

Nivel de dificultad: Muy fácil.

1. Para crear una aplicación en LiveCode, es necesario agregar un *Mainstack*, traducido como pila principal. Para ello, haga clic en **File > New Mainstack** como se señala en la Figura 1.1

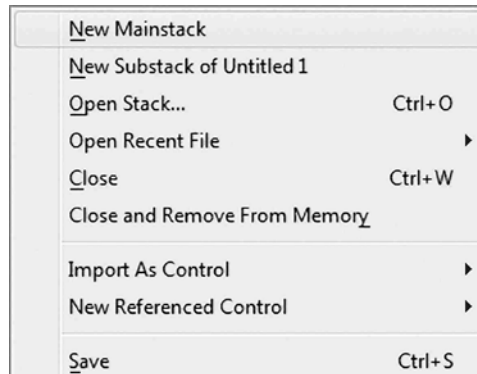



Figura 1.1. Creación de una pila principal.

2. Agregar objetos. Desplácese a la paleta de herramientas y seleccione el **edit (pointer) tool**  (ver Figura 1.2).

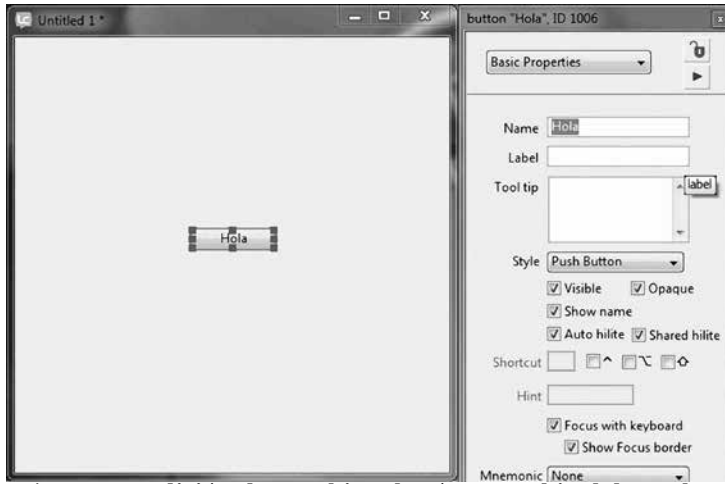



Figura 1.3. Adición de un objeto botón y cambio del nombre

5. Haga clic en el botón **Code**  de la parte superior. Éste abrirá una nueva ventana y mostrará ya escrito un *message handler* “mouseUp” como se aprecia en la Figura 1.4.

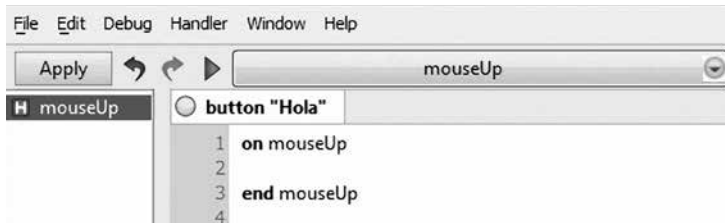


Figura 1.4. Message Handler del objeto botón

6. Agregue el código de la Figura 1.5 al *message handler*, tal como se ilustra en la Figura 1.6

```
1 answer "Hola Mundo"
```

Figura 1.5. Hola Mundo en LiveCode

Práctica 1: Hola mundo

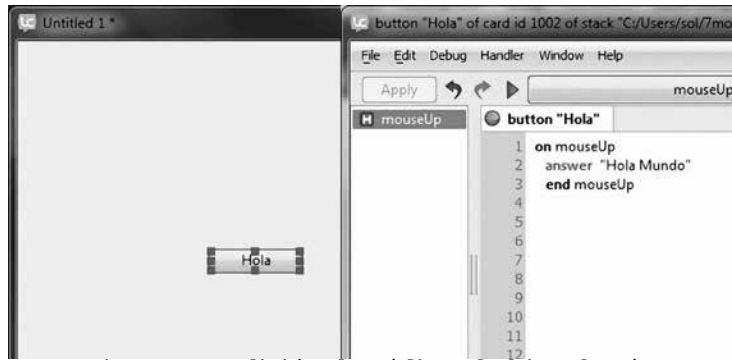


Figura 1.6. Adición de código al objeto botón


7. Haga clic al botón **Apply** para guardar los cambios. Si el código fue escrito correctamente, la parte inferior debe decir “No errors occurred”. De no ser así, verifique el código ingresado.
8. Ejecute el *script* del botón. Para ello, seleccione el **Run (browse) Tool**  y haga clic en el botón “Hola”. Si los pasos han sido efectuados correctamente, el programa se verá como en la Figura 1.7.



Figura 1.7. Ejecución del objeto botón

9. Guarde el proyecto. Haga clic en el menú **File > Save As** y nombre el proyecto como “Hola Mundo” (ver Figura 1.8).

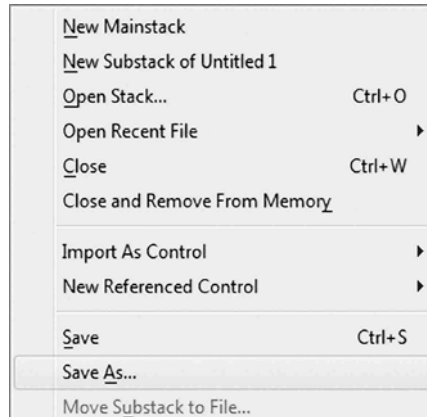


Figura 1.8. Procedimiento para guardar un programa en LiveCode

Explicación de la práctica 1

En este punto, usted ya ha agregado todos los elementos de la estructura de una aplicación en LiveCode. Los elementos fueron los siguientes:


- *Mainstack* (pila principal) con una *tarjeta* ya incluida.
- *Objects* (en este caso el *object button*).
- Un *script* asociado al *object button*.
- *Message handler*. Éste especifica el comportamiento deseado al momento de ejecutar la aplicación.

Practica 2. Textos

Objetivo: Realizar un programa que incluya entrada y salida de texto, utilizando los comandos `ask`, `answer` y `put`.


Nivel de dificultad: Muy fácil.

Este ejemplo es similar a la práctica anterior. La diferencia radica en que el `script` que se utiliza en el `message handler` “`mouseUp`”, admite entrada de texto. Los pasos a realizar en esta práctica, se mencionan a continuación:

1. Abra el proyecto “Hola Mundo”, realizado en la Práctica 1. Haga clic en el menú `File > Open Recent File` (si se trata de un proyecto reciente) o `File > Open Stack` para buscar en el explorador de archivos.
2. Cambie el nombre del `object button` “Hola” a “Bienvenidos”. Seleccione el `object button`. Haga clic en `Inspector` y reemplace el nombre “Hola” por “Bienvenidos”.
3. Reescriba el `script` del `object button`. Haga clic en `Code`  y escriba el código de la Figura 2.1.

```
1 on mouseUp
2 local tGeneración, tExpectativas
3   ask "Saludos a todos los alumnos de programación en LiveCode" \
4     & "versión 7.0.1 ¿A qué generación perteneces?"
5   put it into tGeneración
6   answer "Bienvenidos generación " & tGeneración
7   ask "¿Qué esperas de este curso?"
8   put it into tExpectativas
9   answer "¡En serio espero: " & tExpectativas & "!"
10 end mouseUp
```

Figura 2.1. Manejo de textos: `ask`, `put`, `answer`.

4. Guarde los cambios. Haga clic al botón **Apply** para guardar los cambios. Si el código fue escrito correctamente, la parte inferior debe decir “No errors occurred”. De no ser así, verifique el código ingresado.
5. Ejecute el `script` del botón. Seleccione el **Run (browse) Tool**  y haga clic en el botón “Bienvenidos”.
6. Guarde el proyecto como “Textos”.

Explicación de la práctica 2

Al dar clic al botón **Bienvenidos**, comienza la ejecución del `script`. La línea 3 de la Figura 2.1 muestra un cuadro de diálogo con un campo de texto. A su vez, el cuadro imprime el siguiente texto: “Saludos a todos los alumnos de programación en LiveCode versión 7.0.1 ¿A qué generación perteneces?” (Figura 2.2).

Esto se debe a que LiveCode interpreta la palabra `ask` como la acción de mostrar un cuadro de diálogo con un `textfield`, dos botones (Ok, Cancel) y el texto indicado. En la ejecución de ejemplo, le fue enviado como respuesta “Febrero 2016-Agosto 2016” y se hizo clic en **Ok**. Cabe mencionar que la línea 2 se traduce del inglés al español como pregunta “Saludos a todos los alumnos [...] ¿A qué generación perteneces?”

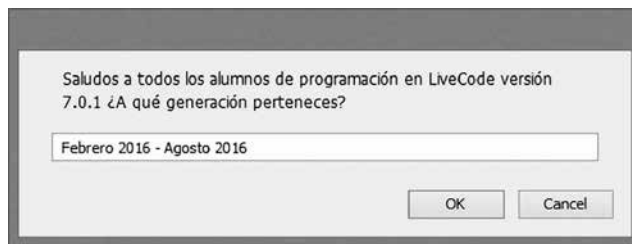


Figura 2.2. Ejecución de la línea 2 del `script`

La ejecución de las líneas 4 y 5 (Figura 2.1) puede observarse en la Figura 2.3. La línea 4 indica con el comando `put` que la máquina debe leer la información introducida en el cuadro de diálogo y asignarla a la variable “`tGeneración`”. En el ejemplo llamamos `it` al texto introducido en el cuadro. De igual manera, la línea 5 se traduce como `colócalo` en `generación`.

La línea 6 usa el comando `answer` para decirle al programa que muestre en un cuadro de diálogo “Bienvenidos generación” y el valor de la variable “`Generación`” (que es la respuesta ingresada previamente). `Answer` se traduce directamente al español como `responder`.

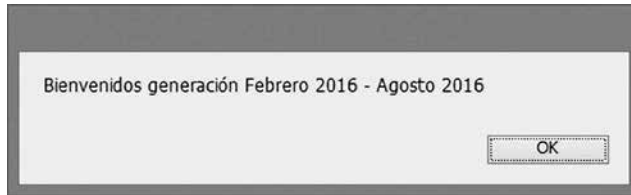


Figura 2.3. Ejecución de la línea 4 del script

La línea 7 indica un nuevo **ask**, en este caso con el texto “¿Qué esperas de este curso?” (ver Figura 2.4).

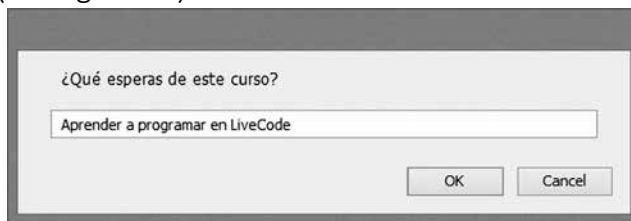


Figura 2.4. Ejecución de la línea 5 del script

Nuevamente, la máquina toma el texto ingresado en el campo y lo guarda en una variable. Después, muestra el texto ingresado con el comando **answer** (ver Figura 2.5).

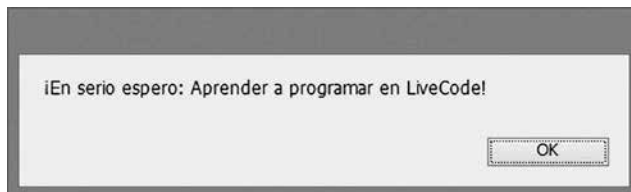


Figura 2.5. Ejecución de la línea 7 del script

Tenga en cuenta que esta práctica ejemplifica la sintaxis similar al inglés que presenta el lenguaje de LiveCode. Retomando lo explicado en el apartado Introducción a LiveCode, este tipo de sintaxis se encuentra limitada y sólo funciona con procesamiento de texto y datos. Para crear bases de datos y operaciones avanzadas con móviles, es necesario utilizar funciones y comandos.

Practica 3. Cálculos

Objetivo: Realizar un programa que calcule la cantidad aproximada de días vividos por una persona y muestre el resultado en un cuadro de diálogo.

Nivel de dificultad: Muy fácil.

1. Cree un nuevo **Mainstack** dando clic en **File > New Mainstack**.
2. Arrastre un *object button* al **Mainstack** creado.
3. Cambie el nombre del *object button* a “Calcular Edad”.
4. Haga clic en el **Inspector** para agregar código al botón “Calcular Edad”.
5. Ingrese el código de la Figura 3.1 en el *message handler mouseUp*:

```
1 local tEdad, tDías
2 ask "¿Cuántos años tienes?"
3 put it into tEdad
4 put (tEdad) * 365 into tDías
5 answer "En total tienes aproximadamente: "& tDías & " días vividos"
```

Figura 3.1. Cálculo de días vividos

6. Verifique que el código esté escrito correctamente y aplique los cambios hechos en el *script*.
7. Guarde el proyecto como “Cálculos”.
8. Ejecute el programa. El programa deberá comportarse conforme a lo señalado en la Figura 3.2.

Práctica 3: Cálculos

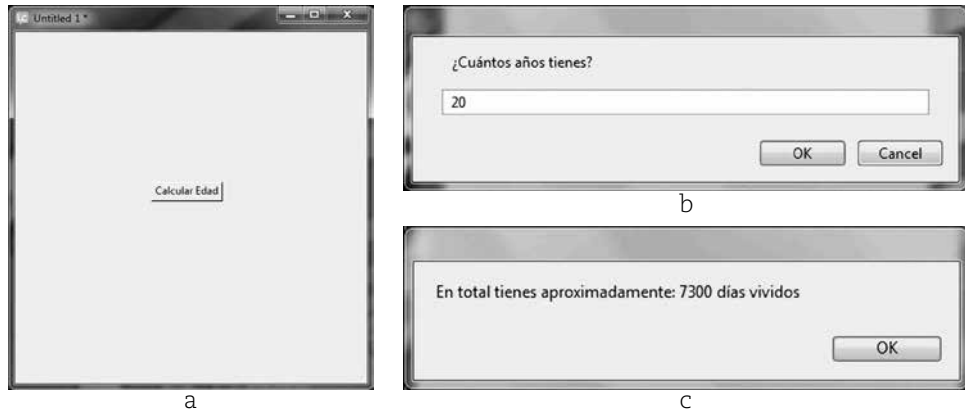


Figura 3.2. Vista previa del botón *calcular edad*.

Explicación de la práctica 3

Como puede observarse en la práctica, se utilizan tres elementos: texto, números y cálculos. En este caso se hace uso de un operador numérico y el operador `&`, donde el último sirve para unir cadenas. Un aspecto observable sobre LiveCode, es que no es necesario especificar el tipo de dato al momento de declarar las variables. LiveCode se encarga de presentar los valores en su tipo correcto.

Practica 4. Uso de funciones y comandos

Objetivo: Realizar un programa que convierta de metros a pulgadas la estatura de una persona, utilizando *functions*, *command handlers* y *message handlers*.

Nivel de dificultad: Muy fácil.

1. Cree un nuevo *Mainstack*.
2. Arrastre un *object button* al *Mainstack* creado.
3. Cambie el nombre del botón a “Estatura”.
4. Haga clic en el **Inspector** para agregar un código al botón “Estatura”.
5. Reemplace el *script* del botón “Estatura” por el código indicado en la Figura 4.1.

```
1 on mouseUp
2   AccionesPrograma
3 end mouseUp
4
5 function calcularEstatura tEstatura
6   return (tEstatura*100)/2.54
7 end calcularEstatura
8
9 command accionesPrograma
10  local tEstatura
11  ask "¿Cuanto mides?"
12  put it into tEstatura
13  answer "En pulgadas mides aproximadamente: "& \
14  calcularEstatura(tEstatura)
15 end accionesPrograma
```

Figura 4.1. Código para calcular estatura

6. Aplique los cambios y verifique que el código esté escrito correctamente.
7. Guarde el proyecto como “Funciones y comandos”
8. Ejecute el programa. El programa deberá actuar como lo indica la Figura 4.2.

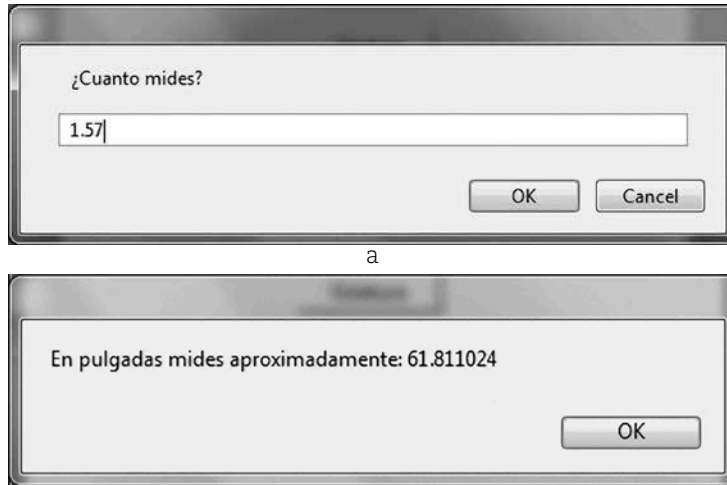


Figura 4.2. Ejecución botón estatura.

Explicación de la práctica 4

La presente práctica introduce el uso de funciones y comandos. En esta ocasión, el código consta de tres partes:

El *message handler* “**mouseUp**”. Su cuerpo de instrucciones sólo consta de la palabra “accionesPrograma”, que corresponde al nombre de un comando. Esto quiere decir que cuando el usuario ejecute el programa y dé clic al botón “Estatura”, LiveCode llamará al comando “accionesPrograma” para que realice las acciones que tiene indicadas en su interior (ver figura 4.3).

```
1 on mouseUp
2   AccionesPrograma
3 end mouseUp
```

Figura 4.3. Llamado de comandos

La función “**calcularEstatura**”. Como se mencionó en el apartado de la estructura de LiveCode, una función debe devolver un valor. Es por ello que consta de tres elementos para su construcción: el inicio de la función, el valor que devolverá y el final de la función. El inicio se declara con la

palabra **function** seguido del *nombre de la función*. El valor que devolverá es declarado con la palabra **return** y la *variable u operación* que el usuario desea que devuelva. Por último, el fin es indicado con la palabra **end** y el *nombre de la función* (ver figura 4.4).

```
5 function calcularEstatura tEstatura
6     return (tEstatura*100)/2.54
7 end calcularEstatura
```

Figura 4.4. Función calcular estatura

El **command handler** “**accionesPrograma**”. Para construir un *command handler*, es necesario declarar en dónde comienza y en dónde termina. El comienzo es declarado con la palabra **command** seguido por el *nombre del comando* y el final con la palabra **end** y el *nombre del comando*. El código escrito entre la declaración corresponde a las instrucciones que el usuario desea realizar en el comando (ver figura 4.5).

```
9 command accionesPrograma
10     local tEstatura
11     ask "¿Cuanto mides?"
12     put it into tEstatura
13     answer "En pulgadas mides aproximadamente: "& \
14     calcularEstatura(tEstatura)
15 end accionesPrograma
```

Figura 4.5. Comando accionesPrograma.

Practica 5. Utilización de imágenes

Objetivo: Crear un programa que muestre una imagen distinta dependiendo del botón pulsado.

Nivel de dificultad: Fácil.

1. Cree un nuevo *Mainstack*.
2. Cambie el nombre del *stack* a “*imagenes*”.
3. Ajuste el tamaño del *stack* a 300 x 500 píxeles. En el Inspector, despliegue el apartado de **size and position**. En el campo **width** escriba 300 y en **height** 500. Los cambios realizados al *stack* se observan en la Figura 5.1.

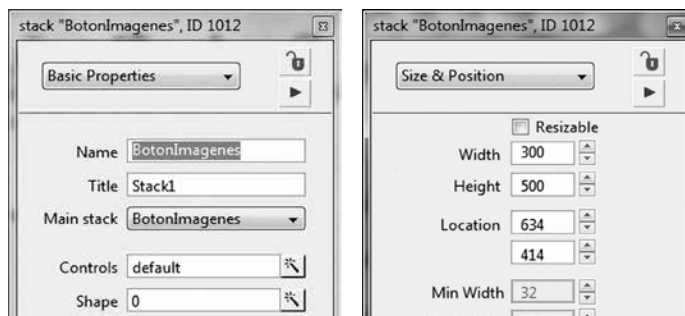


Figura 5.1. Propiedades asignadas al *stack*: a) nombre y b) tamaño

4. Arrastre un *object button* y asignele el nombre de “Gato”.
5. Repita el paso 4, agregando *object button*. Asignele los nombres “Perro” y “Caballo”.
6. Busque en la *paleta de herramientas* el objeto *Image Area* y arrástrelo a la tarjeta. La disposición de los objetos debe quedar como se indica en la Figura 5.2

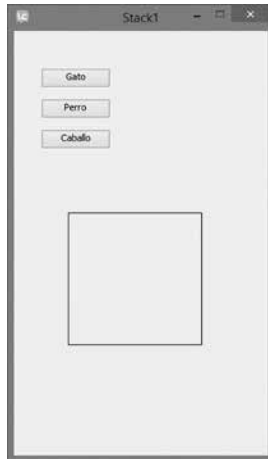


Figura 5.2. Disposición de los objetos arrastrados

7. Seleccione el objeto *Image Area* y abra su cuadro de propiedades.
8. Modifique el parámetro **border width** a 0. Esto es para que el *Image Area* no muestre sus bordes.
9. Fije la posición del *Image Area*. Para ello, despliegue el apartado **size and position** en el *Inspector* y active la casilla de **Lock size and position** (ver Figura 5.3). La finalidad es evitar que su tamaño sea alterado cuando la imagen cambie.

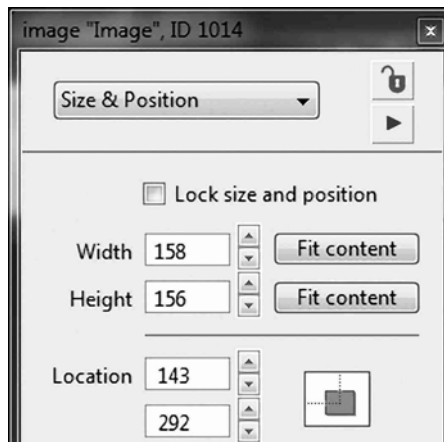



Figura 5.3. Propiedades size & position de la imagen

Práctica 5: Utilización de imágenes

10. Ajuste el tamaño del *Image Area* a 200 x 200 píxeles.
11. Guarde el *Mainstack* como “Imágenes”.
12. Copie las imágenes del caballo, perro y gato en la misma carpeta en que el *Mainstack* fue guardado.³
13. Regrese al *Mainstack* y seleccione el botón “Gato”.
14. Haga clic en *Code*  y escriba el código de la Figura 5.4 en el *message handler mouseUp*.

```
1 set the filename of the image "Gato" to "gato.jpg"
```

Figura 5.4. Código para cambiar la ruta de la imagen.

15. Aplique los cambios
16. Repita los pasos 13-15 con los botones “Perro” y “Caballo”. Recuerde cambiar el nombre de la imagen y su ruta en el código.
17. Guarde el proyecto como “Imágenes”
18. Ejecute el programa, el resultado debe ser como el que se muestra en la Figura 5.5.

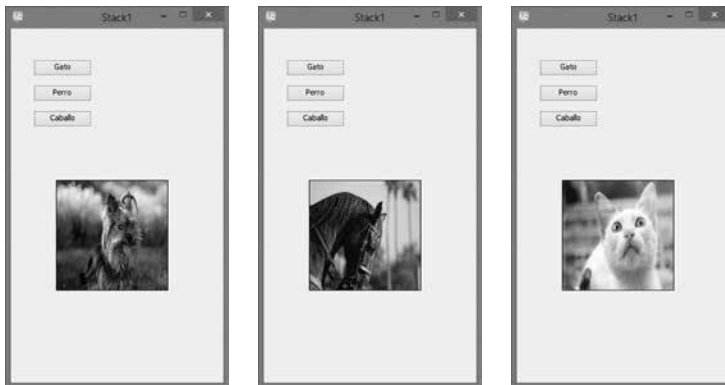


Figura 5.5. Vista previa de la ejecución del programa

Explicación de la práctica 5

Lo que se busca en esta práctica es mostrar una imagen distinta conforme a cada botón pulsado. Para resolver este problema, se creó un *Image Area* en blanco; es decir, que no mostraba ninguna imagen.

Posteriormente, con el código de la Figura 5.4, se cambió la propiedad

³ Las imágenes utilizadas en las prácticas se encuentran incluidas en el CD adjunto a este material educativo.

filename. Esta propiedad corresponde a la ruta en disco de la imagen que mostrará el *Image Area*. Dado que las imágenes fueron copiadas en la misma ruta que el *stack*, la ruta se compone solamente de dos elementos: nombre y extensión. Por ejemplo: Caballo (nombre de la imagen) .jpg (Extensión). Si la imagen se ubicara en **Mis Documentos de Windows**, la ruta debe indicar: carpeta en donde se encuentra, el nombre y la extensión.

Cabe destacar que el programa no maneja tres imágenes diferentes. Más bien, utiliza una imagen que apunta a tres archivos diferentes.

Practica 6. Transferir aplicaciones desarrolladas en LiveCode a un dispositivo Android

Objetivo: Efectuar las configuraciones necesarias para ejecutar el Mainstack desarrollado en un dispositivo Android.

Nivel de dificultad: Fácil.

1. Abra el proyecto “Hola Mundo”. Seleccione el menú **File > Standalone Application Settings**.
2. Haga clic en el apartado Android y active la casilla **Build For Android**.
3. Seleccione las opciones indicadas en la Figura 6.1 y cierre la ventana.
[Nota: el identificador por defecto es **com.yourcompany.yourapp**. No obstante, éste debe ser único en cada aplicación. En este caso se utilizará el identificador “com.LibroLiveCode.HolaMundo”].



Figura 6.1. Configuraciones para probar una aplicación en Android

4. (Opcional) Si el **Mainstack** contiene archivos adicionales, como en la Práctica 5, desplácese al apartado **Add Files** y haga clic en el botón **Add File** para añadir cada archivo utilizado.
5. Haga clic en el menú **Development > Test target** y seleccione el dispositivo a utilizar, ya sea virtual o físico. Si el dispositivo a utilizar se trata de un dispositivo físico, recuerde activar la depuración en los ajustes del teléfono.
6. Ejecute el **Mainstack** en un dispositivo Android. Haga clic en el botón **Test**. LiveCode procederá a crear las clases y paquetes necesarios para que el **Mainstack** sea ejecutable en un dispositivo Android. La aplicación resultante del ejemplo se puede observar en la Figura 6.2.



Figura 6.2. Vista previa de la aplicación “Hola Mundo” en un dispositivo con Android 4.1.

Explicación de la práctica 6

En esta práctica se busca que el usuario aprenda a ejecutar las aplicaciones hechas en un dispositivo Android. No hay código de por medio; por tanto, se omitirá una explicación.

Es recomendable que se ejecuten las aplicaciones hechas hasta el momento en un dispositivo Android (Prácticas 1-5). Esto es para que el usuario se familiarice con el uso de un dispositivo Android.

Practica 7. Archivos multimedia

Objetivo: Desarrollar aplicaciones en Android que reproduzcan audio y video.

Nivel de dificultad: Fácil.

Aplicación que reproduce un video.

1. Cree un *Mainstack* y cambie el nombre del *Mainstack* a “Videos”.
2. Haga clic al botón **Code** y escriba el código de la Figura 7.1.

```
1 on openCard
2 set the rect of this stack to the working screenRect
3 //Creeo el objeto reproductor
4 mobileControlCreate "player", "videoControl"
5
6 //Establezco las propiedades
7 mobileControlSet "videoControl", "filename", \
8 specialFolderPath("engine") & "/video.mp4"
9 mobileControlSet "videoControl", "showController", true
10 mobileControlSet "videoControl", "visible", true
11 mobileControlSet "videoControl", "rect", \
12 the rectangle of this card
13
14 //Empiezo a reproducir el archive multimedia
15 mobileControlDo "videoControl", "play"
16 end openCard
```

Figura 7.1. Código que crea reproductor de video

3. Guarde los cambios.
4. Guarde el proyecto como “Videos”.
5. Copie los archivos utilizados en la aplicación. Haga clic en **File > StandAlone Application Settings** y vaya a la sección **Copy Files**. Pulse el botón **Add File** y añada el video que se encuentra adjunto en el CD de este material. En el ejemplo se buscó que el archivo de video estuviera en la misma carpeta que el *stack* (ver Figura 7.2). Sin embargo, en este caso se adjuntará el video a los archivos que utiliza la aplicación. No es necesario que esté en la misma carpeta del *stack*, el archivo sólo debe estar adjunto.

Nota: En el código es indicado un video de nombre “video” con extensión “mp4”. Observe que coincida el nombre y extensión del video indicado en el código con el video a agregar.



Figura 7.2. Agregando videos al stack

6. Desplácese a la pestaña Android y active la casilla **Build For Android**.
7. Establezca como identificador del proyecto “com.LibroLiveCode.Videos”.
8. Pruebe la aplicación en un dispositivo Android. En el menú Development > **Test Target** elija el dispositivo a utilizar. Posteriormente, haga clic en el botón **Test**. La Figura 7.3 muestra la ejecución de la aplicación.
- 9.



Figura 7.3. Ejecución del reproductor de video

Aplicación que reproduce un audio

1. Cree un nuevo `Mainstack`.
2. **Añada al `Mainstack`** el código indicado en la Figura 7.1.
3. En la línea 8, reemplace el archivo “video.mp4” por “audio.mp3”. El archivo correspondiente a esta ruta se encuentra en el CD incluido con este material.
4. Guarde los cambios. Guarde el `Mainstack` como “Audios”.
5. Añada el archivo de audio como parte de la aplicación.
6. Desplácese a la pestaña Android y active la casilla **Build For Android**. Establezca como identificador del proyecto “com.LibroLiveCode.Audios”.
7. Pruebe la aplicación en un dispositivo Android.

Explicación de la práctica 7

En la figura 7.1, las líneas 1 y 14 establecen las acciones que debe hacer el *message handler* `OpenCard`. Éstas serán realizadas en cuanto la tarjeta sea abierta en la aplicación. La línea 2 (Figura 7.1) establece el valor propiedad `rect`, que especifica las dimensiones y posición de la tarjeta. En este caso, a `rect` se le asignan las dimensiones de la pantalla del dispositivo que ejecuta la aplicación. La finalidad es que ésta se adapte a cualquier tamaño de pantalla.

Posteriormente, se crea el objeto `player` con el comando `mobileControlCreate` (línea 4, figura 7.1), que consta del tipo de objeto a crear (un objeto `player`, que es el reproductor) y el nombre del objeto.

En las líneas 7-12 (Figura 7.1), se establecen propiedades del reproductor con el comando `mobileControlSet`. Las propiedades asignadas son: la ruta del video a reproducir (`filename`), los controles de video (`showController`), la visibilidad del objeto (`visible`) y las dimensiones del objeto (`rect`).

Finalmente, el comando `mobileControlDo` (Línea 15, figura 7.1) indica al objeto “videocontrol” que reproduzca el video en cuanto la aplicación sea abierta.

Practica 8. Creación de interfaz gráfica personalizada

Objetivo: Crear una interfaz gráfica personalizada para una aplicación que reproduzca videos.

Nivel de dificultad: Medio.

1. Cree un nuevo Mainstack.
2. Cree una nueva tarjeta en el Mainstack. Para ello, haga clic derecho en el área del Mainstack y seleccione la opción **New Card**, como se muestra en la Figura 8.1.

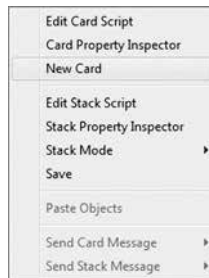


Figura 8.1. Creación de una tarjeta en el Mainstack

3. Establezca el tamaño del stack a 300 x 500.
4. En el Inspector, seleccione la tarjeta recién creada. Haga clic en el botón de la flecha y seleccione la tarjeta número [2] (ver Figura 8.2). Mediante el Inspector, usted puede seleccionar elementos invisibles en la aplicación.

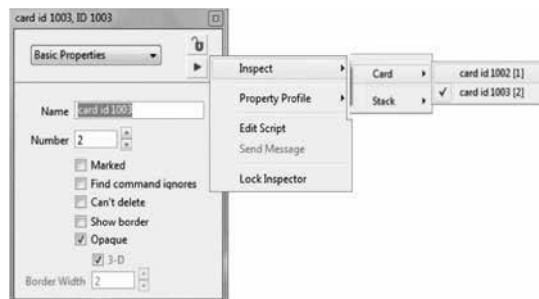


Figura 8.2. Selección de tarjetas distintas de un stack

Práctica 8: Creación de interfaz gráfica personalizada

- Haga clic en el menú **File > Import as control > Image File**. En el explorador, seleccione las imágenes de play, stop y pause del CD adjunto a este material. En esta ocasión, por tratarse de la interfaz gráfica para un reproductor de video, agregue las imágenes para los botones de **play, pause y stop** (ver Figura 8.3).

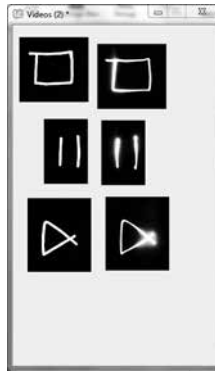


Figura 8.3. Imágenes que se utilizarán en la interfaz gráfica

- Desplácese a la tarjeta anterior y arrastre 3 *object button* en la tarjeta principal.
- Seleccione un botón y con el Inspector establezca las propiedades mostradas en la Figura 8.4. Lo que se busca es ocultar el texto del botón y que no tenga bordes, para agregar el fondo personalizado.

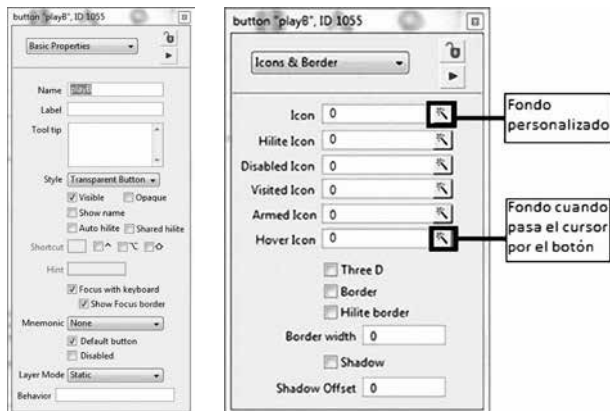


Figura 8.4. Opciones para hacer un botón con fondo personalizado

8. Abra el *Image Chooser*. Haga clic en el botón de la propiedad *icon* (Figura 8.4, inciso b). Su función es establecer un fondo personalizado para un botón.
9. Seleccione el *Image Library This Stack* (Figura 8.5). Éste denota a todas las imágenes incluidas en el *stack*. En este apartado puede acceder a las imágenes añadidas en la tarjeta oculta.
10. Elija la imagen que utilizará para el botón y presione *Ok*. En este ejemplo, se utiliza la imagen de *play* sin bordes coloreados.

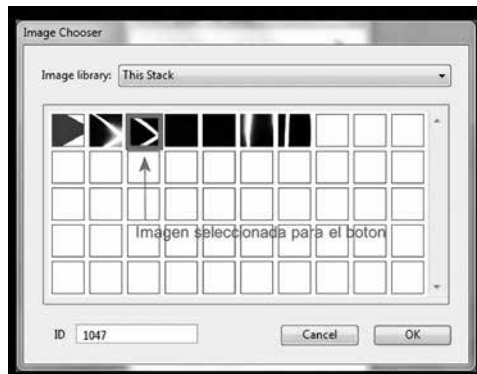


Figura 8.5. Selección del fondo del botón

11. Presione el botón que se encuentra a un costado de la propiedad *Hover Icon* y asígnele una imagen. Esta imagen mostrará el sistema cuando el cursor pase sobre el botón. En este ejemplo, se eligió el botón de *play* con bordes coloreados.
12. Repita los pasos 8 a 10 con los demás botones. La tarjeta resultante debe ser similar a la mostrada en la Figura 8.6.



Figura 8.6. Tarjeta principal con botones personalizados

Práctica 8: Creación de interfaz gráfica personalizada

13. Cambie el fondo de la tarjeta. Desplácese al apartado **Colors & Patterns**. Utilice color negro como *background*. La casilla izquierda establece imágenes y la casilla derecha fondos de color sólido. Use la casilla derecha.
14. Arrastre un objeto *label* en la parte superior del *stack* y utilice el Inspector para establecer las propiedades indicadas en la Figura 8.7.

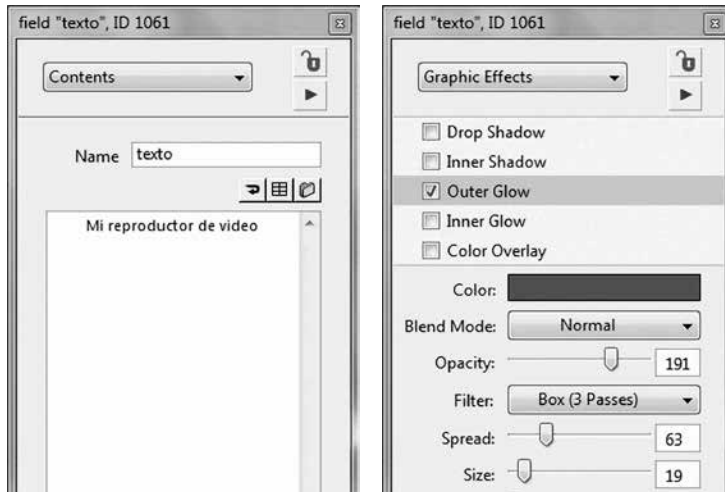


Figura 8.7. Propiedades del objeto *label*

15. Guarde el proyecto como "Interfaz videos". Al finalizar esta práctica, el diseño de la tarjeta debe quedar como en la Figura 8.8.



Figura 8.8. Interfaz gráfica personalizada de la aplicación

Explicación de la práctica 8.

Esta práctica introduce al usuario a la personalización de la interfaz gráfica de la aplicación. Anteriormente, se había buscado que se realizará la funcionalidad de la aplicación, sin personalizar su aspecto. Por tanto, no habrá explicación de código. La funcionalidad a implementar en esta interfaz se abordará en la práctica 9

Practica 9. Implementación de código en interfaz gráfica personalizada

}

Objetivo: Implementar el código de una aplicación para reproducir videos, en la interfaz gráfica personalizada.

Nivel de dificultad: Media.

1. Abra el proyecto “Interfaz videos”.
2. Seleccione la tarjeta número [1] con el Inspector.
3. Escriba en ella el código de la Figura 9.1.

```
1 on openCard
2 //Creo el reproductor
3 mobileControlCreate "player", "videoControl"
4 //propiedades
5 mobileControlSet "videoControl", "filename", \
6 specialFolderPath("engine") & "/video.mp4"
7 mobileControlSet "videoControl", "showController", false
8 mobileControlSet "videoControl", "visible", true
9 mobileControlSet "videoControl", "rect", "0,50,350,400"
10
11 //le doy play
12 mobileControlDo "videoControl", "play"
13 end openCard
```

Figura 9.1. Creación de reproductor de video

4. Implemente las acciones de los botones. Escriba el código de la Figura 9.2 en el script del botón **play**.

```
1 on mouseUp
2     mobileControlDo "videoControl", "play"
3 end mouseUp
```

Figura 9.2. Acciones del botón play

5. Repita el procedimiento con los botones de pause y stop. Cambie el valor “play” por “pause” y “stop”.
6. Aplique los cambios en el script y guarde el proyecto. Al ejecutarse en un teléfono móvil, la aplicación debe verse como en la Figura 9.3.

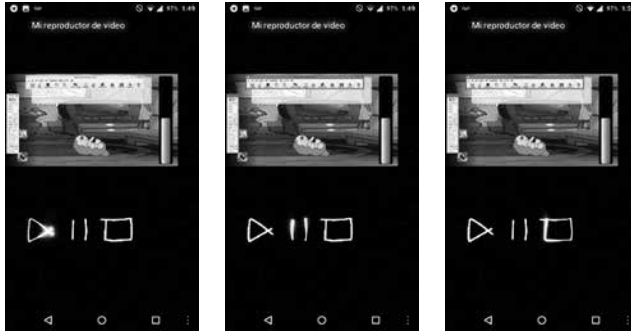


Figura 9.3. Vista previa de aplicación personalizada presionando play, pause y stop

Explicación de la práctica 9

El código de la Figura 9.1 es similar al de la Figura 7.1. La diferencia radica en dos parámetros. La propiedad `showController`, es `false` y `rect` tiene los valores de "0, 50, 350, 400". El primero es para que el controlador que traía el objeto de video esté oculto y utilizar el controlador personalizado en su lugar. La segunda propiedad, como se había explicado en la Práctica 7, especifica la posición y dimensiones del objeto.

`Rect` se compone de cuatro parámetros: `left`, `top`, `right`, `bottom`. Estos significan: izquierda, arriba, derecha, abajo. Sin embargo, no funcionan como indica su traducción, sino que son distancias existentes entre los costados del `stack` y el objeto. En la Figura 9.4 se especifica el funcionamiento de cada parámetro.

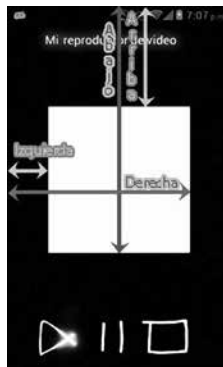


Figura 9.4. Funcionamiento de los valores de `rect` en relación al objeto.

Práctica 9: Implementación de código en interfaz gráfica personalizada

Cabe mencionar que los valores de la propiedad `rect` indican las dimensiones y posición del objeto, debido a la forma en que funcionan.

En cuanto al `script` de la Figura 9.2, el comando `mobileControlDo` ordena al control “`videoControl`” que reproduzca el video cuando el usuario haga clic en el botón de **play**.

Practica 10. Aplicaciones ajustables al tamaño de pantalla

Objetivo: Modificar las aplicaciones de manera que sean presentables en diferentes tamaños de pantalla.

Nivel de dificultad: Media.

Existen dos maneras de ajustar el tamaño de las aplicaciones en diferentes pantallas (LiveCode Lessons, 2013). Éstas son:

- Utilizando la propiedad **fullscreenmode**, que se encuentra disponible a partir de la versión 6.5 de LiveCode.
- Escalar los elementos del **stack** por código.

Uso de la propiedad fullscreenmode

1. Abra el **stack** “Interfaz videos”.
2. Con el Inspector, seleccione el **stack** principal.
3. Presione el botón **Code** y agregue el código indicado en la Figura 10.1.
4. Aplique los cambios en el **script**.

```
1 on preOpenStack
2   set the fullscreenmode of me to "exactFit"
3 end preOpenStack
```

Figura 10.1. Uso de la propiedad fullscreenmode

Explicación de la propiedad fullscreenmode de la práctica 10

El objetivo del Código 10.1 es ajustar el tamaño del **stack** antes de que la aplicación sea visible para el usuario. Para ello se utiliza el *message* “preOpenStack”. En la línea 2 del código 10.1, *me* hace referencia al objeto al que pertenece el código.

El tipo y nombre del objeto *me*, puede consultarse en el encabezado de la pestaña. Por ejemplo, en este caso *me* significa el **stack** “Práctica 10” (ver Figura 10.2).

Práctica 10: Aplicaciones ajustables al tamaño de pantalla

```
a ○ stack "Práctica 10" b
1 on preOpenStack
2   set the fullscreenmode of me to "exactFit"
3 end preOpenStack
4
5
```

Figura 10.2. Nombre del objeto y tipo de objeto en el código

La ventaja de la propiedad `fullscreenmode`, es que LiveCode se encarga de escalar la aplicación obedeciendo al tamaño de la pantalla. Dependiendo el parámetro asignado, LiveCode redimensionará los objetos al tamaño de la pantalla. La desventaja es que los elementos pequeños pueden verse distorsionados en altas resoluciones.

Por ejemplo, en la Figura 10.3 se muestra el escalado de los elementos resultante en una tablet y un móvil. El parámetro “exactFit” consiste en escalar para llenar la pantalla. Los elementos son estirados de forma que la posición de los objetos sea la misma.

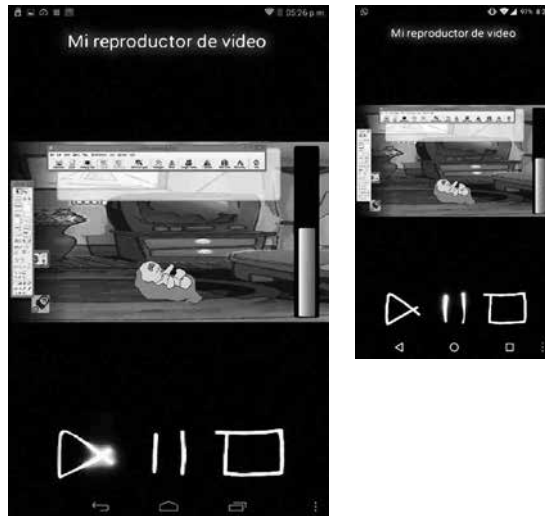


Figura 10.3. Aplicación de videos en móviles de diferentes tamaños.

a) Tablet Acer Iconia B1-750 b) Moto E 2015 xt1027

Existen otros parámetros que pueden utilizarse en la propiedad “`fullscreenmode`”. Estos se describen a continuación:

- **“letterbox”**. Conserva la relación de aspecto, es decir, la proporción entre su altura y anchura. Puede haber espacios en blanco si la relación no coincide con la pantalla del dispositivo.
- **“noBorder”**. Mantiene la relación de aspecto. Si no coincide, los extremos del `stack` no son visibles.
- **“noScale”**. Centra el `stack` al centro. Ningún objeto es escalado.
- **“showAll”**. Escala de forma que todo el contenido del `stack` es visible.

Se recomienda utilizar el parámetro `“ExactFit”` para el escalado de la aplicación, ya que es posible observar todos los objetos en cualquier dispositivo sin que sean distorsionados significativamente.

Escalar los elementos del `stack` por código

Para la implementación del código que escala el `stack`, se realizó una planeación previa de la distribución que tendrán los objetos que lo conforman. La distribución se hizo con base en los valores que debe tomar la propiedad `loc` de cada objeto. `loc` se conforma de dos ítems: `a`, `b`. Donde un ítem es un elemento separado por una coma, `a` es la distancia que hay del borde izquierdo del `stack` al centro del objeto, y `b` la distancia que hay del borde superior al centro del objeto (ver Figura 10.4).

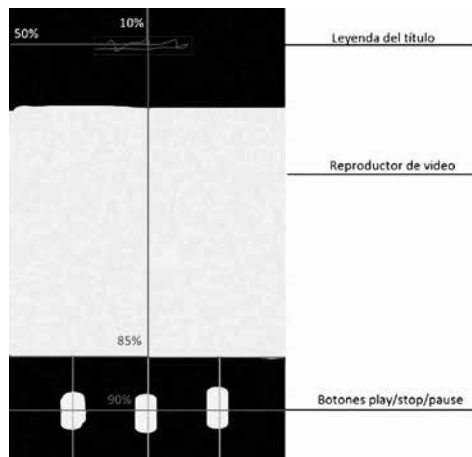


Figura 10.4. Planeación de la distribución de los elementos del `stack`

Teniendo esto en cuenta, efectúe los siguientes pasos:

1. Seleccione la tarjeta principal.
2. Escriba el código señalado en la Figura 10.5 como *script* de la tarjeta.
3. Aplique los cambios.

```
1 local sAnchuraPantalla, sAlturaPantalla
2
3 on preOpenCard
4   local tPuntoMedio, tUbicacionTexto
5   set the rect of this stack to the screenRect
6   //obtengo los valores de altura y anchura
7   put the width of this stack into sAnchuraPantalla
8   put the height of this stack into sAlturaPantalla
9   //Ubicación leyenda del título
10  put item 1 of the loc of this card into tPuntoMedio
11  put tPuntoMedio & "," & (sAlturaPantalla*0.10) into \
12  tUbicacionTexto
13  set the loc of field "texto" to tUbicacionTexto
14  //Alineación de los botones
15  repeat with x = 1 to 3
16    set the loc of button ("boton" & x) to \
17    ((sAnchuraPantalla/4*x) & "," & (sAlturaPantalla*.90))
18  end repeat
19 end preOpenCard
20
21 on openCard
22   local tArriba, tDerecha, tAbajo, tTamaño,
23   //Creo el reproductor
24   mobileControlCreate "player", "videoControl"
25   //medidas del reproductor.
26   put the round of (the height of this stack*.15) into tArriba
27   put sAnchuraPantalla into tDerecha
28   put the round of (the height of this stack*.85) into tAbajo
29   put ("0," & tArriba & "," & tDerecha & "," & tAbajo) \
30   into tTamaño
31   //propiedades
32   mobileControlSet "videoControl", "filename", \
33   specialFolderPath("engine") & "/video.mp4"
34   mobileControlSet "videoControl", "showController", false
35   mobileControlSet "videoControl", "rect", tTamaño
```

```
36 mobileControlSet "videoControl", "visible", true  
37 end openCard
```

Figura 10.5. Escalado de objetos de la aplicación por código.

Como se puede observar en la Figura 10.6, el escalado por código fue efectuado en el reproductor de video. Los botones del reproductor fueron acomodados de forma que utilizaran toda la pantalla, pero no fueron estirados. Esto se debe a que los botones son imágenes pequeñas, si son estiradas para abarcar pantallas grandes, pueden distorsionarse.

La ventaja es que el programador puede controlar a voluntad cuáles elementos desea redimensionar y cuáles desea acomodar. La desventaja es que requiere razonamiento previo y lógica para implementar esta manera de ajustar los elementos.

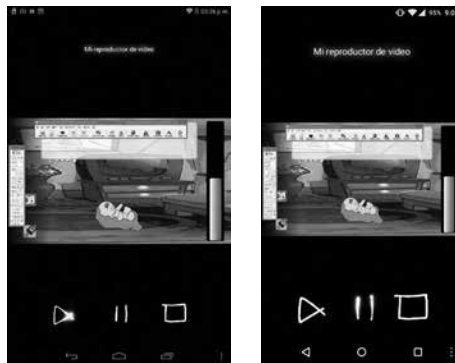


Figura 10.6. Distribución de los elementos de la aplicación
a) Tablet Acer Iconia B1-750 b) Moto E 2015 xt1027

Explicación del escalado por código de la práctica 10

El código se compone de dos *message handlers*: `preOpenCard` y `openCard`.

Message handler `preOpenCard`. Su ejecución ocurre mientras la aplicación es lanzada y termina hasta que la tarjeta aparece ante el usuario.

En la línea 1 de la Figura 10.7 se realiza la declaración de las variables que se utilizarán en el *script*. En otras palabras, éstas pueden ser utilizadas por cualquier *handler* en el *script* y conservarán sus valores (para más información, ver el apartado “Tips para nombrar variables”).

La implementación del *handler* comienza con la línea 2 de la Figura 10.7. En ella se establece que el *stack* tendrá el tamaño de la pantalla del dispositivo que ejecuta la aplicación. Posteriormente, en la línea 5 se establece que el tamaño del *stack* tenga el tamaño de la pantalla. Esto sólo cambia el tamaño del *stack*. No reubica el resto de los elementos.

En las líneas 7 y 8 se almacenan los valores de la altura y la anchura del *stack* en las variables *sAnchuraPantalla* y *sAlturaPantalla*. La finalidad es utilizar dichas variables como base para redimensionar los elementos de la aplicación.

En la línea 10 se asigna *item 1* del *loc* de la tarjeta a la variable *tPuntoMedio*. Éste último guarda la coordenada *x* (ítem 1) del punto medio de la tarjeta, y se utilizará para centrar la leyenda del título. En las líneas 11 y 12 es realizado el cálculo de la ubicación y es asignado a la variable *tUbicacionTexto*. La línea 13 especifica que la disposición del texto será el valor contenido en *tUbicacionTexto*.

Finalmente, se establece la ubicación de los botones en las líneas 15-18. Su funcionamiento consiste en dividir el área de la tarjeta en cuatro partes (ver Figura 10.4) y asignar la ubicación mediante un ciclo que se basa en el nombre de los botones (*boton1*, *boton2*, *boton3*).

```
1 local sAnchuraPantalla, sAlturaPantalla
2
3 on preOpenCard
4   local tPuntoMedio, tUbicacionTexto
5   set the rect of this stack to the screenRect
6   //obtengo los valores de altura y anchura
7   put the width of this stack into sAnchuraPantalla
8   put the height of this stack into sAlturaPantalla
9   //Ubicación leyenda del título
10  put item 1 of the loc of this card into tPuntoMedio
11  put tPuntoMedio & "," & (sAlturaPantalla*0.10) into \
12  tUbicacionTexto
13  set the loc of field "texto" to tUbicacionTexto
14  //Alineación de los botones
15  repeat with x = 1 to 3
16    set the loc of button ("boton" & x) to \
17    ((sAnchuraPantalla/4*x) & "," & (sAlturaPantalla*.90))
18  end repeat
19 end preOpenCard
```

Figura 10.7. Message handler "preOpenCard".

Message handler openCard. Su ejecución comienza en cuanto la tarjeta es abierta, en otras palabras, cuando ésta aparece ante el usuario.

Este bloque de código se centra en acomodar el reproductor de video. Las líneas 26 a 28 (ver Figura 10.8) calculan la posición que tendrá el reproductor. Posteriormente, las líneas 29-30 guardan la ubicación calculada en la variable “tTamaño”. [Nota: el carácter \ es utilizado para indicar que la siguiente línea se ejecutará como una sola. En otras palabras, para LiveCode, las líneas 29-30 son interpretadas como una línea. Su función es facilitar la lectura del código.]

La ventaja de este método es que los elementos de la aplicación no son distorsionados y cambian a una ubicación proporcional al tipo de pantalla que está utilizando el dispositivo. La desventaja, es que la colocación de cada objeto requiere un cálculo previo, por lo que el programador debe utilizar su lógica para calcular la ubicación de los elementos.

```
1 on openCard
2   local tArriba, tDerecha, tAbajo, tTamaño,
3   //Creo el reproductor
4   mobileControlCreate "player", "videoControl"
5   //medidas del reproductor.
6   put the round of (the height of this stack*.15) into tArriba
7   put sAnchuraPantalla into tDerecha
8   put the round of (the height of this stack*.85) into tAbajo
9   put ("0," & tArriba & "," & tDerecha & "," & tAbajo) \
10  into tTamaño
11  //propiedades
12  mobileControlSet "videoControl", "filename", \
13  specialFolderPath("engine") & "/video.mp4"
14  mobileControlSet "videoControl", "showController", false
15  mobileControlSet "videoControl", "rect", tTamaño
16  mobileControlSet "videoControl", "visible", true
17  end openCard
```

Figura 10.8. Message handler “openCard”

Practica 11. Animación de elementos de una aplicación

Objetivo: Programar diferentes tipos de animación de una imagen, dependiendo del botón que sea presionado.

Nivel de dificultad: Media.

1. Cree un nuevo *stack* y guárdelo como “Animaciones”.
2. Arrastre 6 *object buttons*.
3. Inserte una imagen a la tarjeta. Haga clic en **File > Import As Control > Image File** y seleccione la imagen a agregar.
4. Personalice la interfaz de manera que se vea similar a la Figura 11.1.



Figura 11.1. Interfaz personalizada del *stack* “Animaciones”

5. Escriba el código indicado en la Figura 11.2 en la tarjeta principal.

```
1 on rotarImagen
2   desactivarBotones
3   set the resizeMode of image "logo" to "normal"
4   repeat with x=0 to 180
5     set the angle of image "logo" to (the angle of image \
6     "logo" + 2)
7     //wait for 1 milliseconds
8   end repeat
9   set the angle of image "logo" to 0
10  activarBotones
11 end rotarImagen
12
13 on escalarImagen
```

```
14  local tAlturaImg, tAnchuraImg, tAumento
15  desactivarBotones
16  put the height of image "logo" into tAlturaImg
17  put the width of image "logo" into tAnchuraImg
18  repeat with x=1 to 10
19    put 10-x into tAumento
20    set the height of image "logo" to \
21    tAlturaImg*(1+(tAumento/10))
22    set the width of image "logo" to \
23    tAnchuraImg*(1+(tAumento/10))
24    wait for 50 milliseconds
25  end repeat
26  set the height of image "logo" to tAlturaImg
27  set the width of image "logo" to tAnchuraImg
28  activarBotones
29  end escalarImagen
30
31  on aparecer
32    desactivarBotones
33    repeat with x=1 to 50
34      set the blendLevel of image "logo" to 100-(2*x)
35      wait for 3 milliseconds
36    end repeat
37    //set the blendLevel of image "logo" to 1
38    activarBotones
39  end aparecer
40
41  on desaparecer
42    desactivarBotones
43    repeat with x=1 to 50
44      set the blendLevel of image "logo" to 2*x
45      wait for 3 milliseconds
46    end repeat
47    set the blendLevel of image "logo" to 1
48    activarBotones
49  end desaparecer
50
51  on deslizarFuera
52    local tLocInicial, tEjeX, tEjeY
53    desactivarBotones
54    put the loc of image "logo" into tLocInicial
55    put item 1 of tLocInicial into tEjeX
```

Práctica 11: Animación de elementos de una aplicación

```
56     put item 2 of the loc of this stack into tEjeY
57     move image "logo" to tEjeX,tEjeY*3 in 2 seconds
58     wait for 1 second
59     set the loc of image "logo" to tLocInicial
60     activarBotones
61 end deslizarFuera
62
63 on deslizarDentro
64     local tLocInicial, tEjeX, tEjeY
65     desactivarBotones
66     put the loc of image "logo" into tLocInicial
67     put item 1 of tLocInicial into tEjeX
68     put item 2 of the loc of this stack into tEjeY
69     set the loc of image "logo" to tEjeX,tEjeY*2
70     move image "logo" to tLocInicial in 2 seconds
71     activarBotones
72 end deslizarDentro
73
74 on desactivarBotones
75     repeat with x=1 to 6
76         disable button ("b" & x)
77     end repeat
78 end desactivarBotones
79
80 on activarBotones
81     repeat with x=1 to 6
82         enable button ("b" & x)
83     end repeat
84 end activarBotones
```

Figura 11.2. Código del Mainstack "Animaciones".

6. Aplique los cambios del scrip.
7. Seleccione cada botón del *stack*, y cambie sus nombres por b1, b2 hasta llegar al botón b6.
8. Modifique el script de cada botón y mediante el *mensaje handler* `on mouseUp` llame a los *handlers* correspondientes.
9. De manera opcional, puede agregar al *stack* el código de Figura 11.3. Éste se encarga de acomodar los elementos al tamaño de la pantalla.
10. Guarde el proyecto como "Animaciones".

```
1 on preOpenCard
2   local tAlturaPantalla, tAnchuraPantalla, tLocLogo, tEjeY
3   set the rect of this stack to the screenRect
4   // x anchura y altura
5   put the height of this stack into tAlturaPantalla
6   put the width of this stack into tAnchuraPantalla
7   put (tAnchuraPantalla*0.50) & "," & (tAlturaPantalla*0.40)\
8   into tLocLogo
9   set the loc of image "logo" to tLocLogo
10  repeat with tEjeX=0 to 2
11    put tAlturaPantalla*0.60 into tEjeY
12    put (tAnchuraPantalla*0.30) & "," & \
13    (tEjeY+(tAlturaPantalla*(0.05*tEjeX))) into tLocLogo
14    set the loc of button ("b"&(tEjeX+1)) to tLocLogo
15  end repeat
16  repeat with tEjeX=0 to 2
17    put tAlturaPantalla*0.60 into tEjeY
18    put (tAnchuraPantalla*0.70) & "," & \
19    (tEjeY+(tAlturaPantalla*(0.05*tEjeX))) into tLocLogo
20    set the loc of button ("b"&(tEjeX+4)) to tLocLogo
21  end repeat
22 end preOpenCard
```

Figura 11.3. Acomodo de los objetos del stack por código

No se explicará la implementación de la Figura 11.3, dado que se aplicaron los conceptos vistos en la práctica Práctica 10. La vista previa de la aplicación puede observarse en la Figura 11.4.

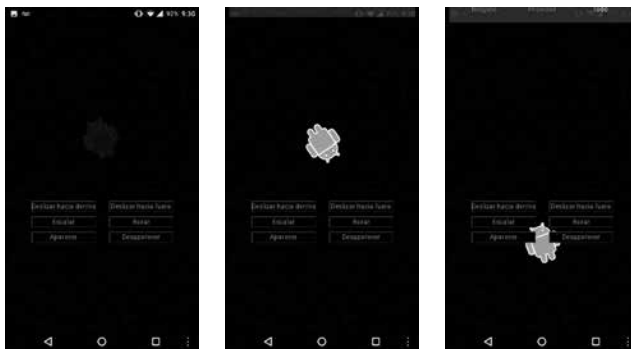


Figura 11.4. Vista previa de las animaciones
a) Aparecer b) Rotar c) Deslizar hacia afuera

Explicación de la práctica 11

A continuación, se explicará a detalle en qué consiste la implementación de los *handlers* utilizados en la práctica.

Rotar Imagen

Para mostrar el efecto de rotación de 360 grados al usuario, se programó una repetición de 0 a 180 en las líneas 4, 5 y 6. En cada repetición el ángulo de la imagen va aumentando de 2 en 2, hasta llegar a 360. Al final de la repetición se establece el ángulo de la imagen como 0, ya que queda en 360, mostrando una imagen ligeramente desalineada. En la línea 3 se estableció la calidad de la imagen a “normal”; esto con el fin de no ocupar memoria al momento de crear la imagen rotada (ver Figura 11.5).

```
1  on rotarImagen
2      desactivarBotones
3      set the resizeMode of image "logo" to "normal"
4      repeat with x=0 to 180
5          set the angle of image "logo" to (the angle of image \
6              "logo" + 2)
7      end repeat
8      set the angle of image "logo" to 0
9      activarBotones
10 end rotarImagen
```

Figura 11.5. Command handler “rotarImagen”

Las palabras “desactivarBotones” y “activarBotones” (ver líneas 1 y 8 de la Figura 11.5), se utilizan para llamar a los respectivos métodos que llevan esos nombres. Sin embargo, el por qué de su implementación será tratado más adelante.

Escalar Imagen

La implementación de este método (ver Figura 11.6), se trató de forma similar al *command handler* “rotarImagen”. En ese caso se guardó el tamaño original de la imagen en las líneas 3 y 4, para reasignarlo en las líneas 11 y 12. Además, se implementó un ciclo del 1 al 10, que utiliza un tiempo de espera de 50 milisegundos por cada repetición (ver línea 9). Este tiempo ayuda a que el efecto sea visible para el usuario.

```
1 on escalarImagen
2   local tAlturaImg, tAnchuraImg, tAumento
3   desactivarBotones
4   put the height of image "logo" into tAlturaImg
5   put the width of image "logo" into tAnchuraImg
6   repeat with x=1 to 10
7     put 10-x into tAumento
8     set the height of image "logo" to \
9     tAlturaImg*(1+(tAumento/10))
10    set the width of image "logo" to \
11    tAnchuraImg*(1+(tAumento/10))
12    wait for 50 milliseconds
13  end repeat
14  set the height of image "logo" to tAlturaImg
15  set the width of image "logo" to tAnchuraImg
16  activarBotones
17 end escalarImagen
```

Figura 11.6. Command handler “escalarImagen”

Aparecer

En este caso, la repetición cambia el valor de la propiedad *blendLevel* que establece la opacidad de la imagen. Si el *blendLevel* de un objeto es de 100, éste es completamente transparente ante el usuario. Por tanto, para lograr el efecto deseado, se redujo el *blendLevel* de 2 en 2 por cada repetición, hasta llegar a 0, donde el objeto es totalmente opaco.

```
1 on aparecer
2   desactivarBotones
3   repeat with x=1 to 50
4     set the blendLevel of image "logo" to 100-(2*x)
5     wait for 3 milliseconds
6   end repeat
7   activarBotones
8 end aparecer
```

Figura 11.7. Command handler “fadeIn”

Desaparecer

`Desaparecer` hace referencia al efecto de desaparición con transparencia. Para ello, se manejó una repetición similar a la del `handler` “`fadeIn`” en la Figura 11.8. La diferencia radica en que en vez de reducir el número 100 de 2 en 2 hasta llegar a 0, va aumentando de 2 en 2 hasta llegar a 100 para que el objeto quede transparente. En la línea 7, se establece el `blendLevel` nuevamente en 0, para indicarle al usuario que la animación ha terminado.

```
1 on desaparecer
2   desactivarBotones
3   repeat with x=1 to 50
4     set the blendLevel of image "logo" to 2*x
5     wait for 3 milliseconds
6   end repeat
7   set the blendLevel of image "logo" to 1
8   activarBotones
9 end desaparecer
```

Figura 11.8. Command handler “`fadeOut`”

Deslizar hacia afuera

La animación `slideOut`, a diferencia de las demás, no utiliza una repetición. Esto se debe a que es un movimiento de deslizamiento que avanza desde la ubicación en que se encuentra hasta afuera del `stack`. El código de la Figura 11.9 comienza guardando la ubicación original de la imagen. Posteriormente, en la línea 4, se le asigna a `x` el valor del ítem 1 de la ubicación original. Esto se debe a que el primer ítem indica la coordenada del eje `x`. Después, se toma el segundo ítem del `loc` del `stack` y se le asigna a la variable `y`. En este caso, `y` hace referencia a la coordenada del eje `y` del punto medio del `stack`. [Nota: Recuerde que el `loc` de un `stack` es el punto medio de todo el `stack`].

En cuanto al movimiento, consiste en indicar a la imagen que se mueva en `y` hasta llegar a una posición fuera del `stack`. Por lo tanto, en la línea 6 se multiplica `y * 3`. Esto equivale a una altura del 150% de la pantalla.

```
1 on rotarImagen
2   desactivarBotones
3   set the resizeQuality of image "logo" to "normal"
```

```
4     repeat with x=0 to 180
5         set the angle of image "logo" to (the angle of image \
6             "logo" + 2)
7         //wait for 1 milliseconds
8     end repeat
9     set the angle of image "logo" to 0
10    activarBotones
11 end rotarImagen
12
13 on escalarImagen
14     local tAlturaImg, tAnchuraImg, tAumento
15     desactivarBotones
16     put the height of image "logo" into tAlturaImg
17     put the width of image "logo" into tAnchuraImg
18     repeat with x=1 to 10
19         put 10-x into tAumento
20         set the height of image "logo" to \
21             tAlturaImg*(1+(tAumento/10))
22         set the width of image "logo" to \
23             tAnchuraImg*(1+(tAumento/10))
24         wait for 50 milliseconds
25     end repeat
26     set the height of image "logo" to tAlturaImg
27     set the width of image "logo" to tAnchuraImg
28     activarBotones
29 end escalarImagen
30
31 on aparecer
32     desactivarBotones
33     repeat with x=1 to 50
34         set the blendLevel of image "logo" to 100-(2*x)
35         wait for 3 milliseconds
36     end repeat
37     //set the blendLevel of image "logo" to 1
38     activarBotones
39 end aparecer
40
41 on desaparecer
42     desactivarBotones
43     repeat with x=1 to 50
44         set the blendLevel of image "logo" to 2*x
45         wait for 3 milliseconds
```

Práctica 11: Animación de elementos de una aplicación

```
46     end repeat
47     set the blendLevel of image "logo" to 1
48     activarBotones
49 end desaparecer
50
51 on deslizarFuera
52     local tLocInicial, tEjeX, tEjeY
53     desactivarBotones
54     put the loc of image "logo" into tLocInicial
55     put item 1 of tLocInicial into tEjeX
56     put item 2 of the loc of this stack into tEjeY
57     move image "logo" to tEjeX,tEjeY*3 in 2 seconds
58     wait for 1 second
59     set the loc of image "logo" to tLocInicial
60     activarBotones
61 end deslizarFuera
62
63 on deslizarDentro
64     local tLocInicial, tEjeX, tEjeY
65     desactivarBotones
66     put the loc of image "logo" into tLocInicial
67     put item 1 of tLocInicial into tEjeX
68     put item 2 of the loc of this stack into tEjeY
69     set the loc of image "logo" to tEjeX,tEjeY*2
70     move image "logo" to tLocInicial in 2 seconds
71     activarBotones
72 end deslizarDentro
73
74 on desactivarBotones
75     repeat with x=1 to 6
76         disable button ("b" & x)
77     end repeat
78 end desactivarBotones
79
80 on activarBotones
81     repeat with x=1 to 6
82         enable button ("b" & x)
83     end repeat
84 end activarBotones
```

```
1 on slideOut
2   desactivarBotones
3   put the loc of image logo into ubicacionOriginal
4   put item 1 of ubicacionOriginal into x
5   put item 2 of the loc of this stack into y
6   move image logo to x,y*3 in 2 seconds
7   wait for 1 second
8   set the loc of image logo to ubicacionOriginal
9   activarBotones
10 end slideOut
```

Figura 11.9. Command handler “slideOut”.

Deslizar hacia adentro

SlideIn hace referencia al movimiento opuesto a slideOut. Es decir, en vez de moverse de la posición original a una ubicación fuera del stack, comienza desde una ubicación fuera para volver a la ubicación donde debe terminar el elemento.

Por tanto, el código de Figura 11.10 difiere en que primero se establece como localización de la imagen una ubicación fuera del stack. Posteriormente se mueve la imagen a la ubicación original que se tomó al principio del método (ver línea 3, Figura 11.10).

```
1 on slideIn
2   desactivarBotones
3   put the loc of image logo into ubicacionOriginal
4   put item 1 of ubicacionOriginal into x
5   put item 2 of the loc of this stack into y
6   set the loc of image logo to x,y*3
7   move image logo to ubicacionOriginal in 2 seconds
8   activarBotones
9 end slideIn
```

Figura 11.10. Command handler “slideIn”

Desactivar botones

Consiste en inhabilitar los botones para que no puedan ser presionados. Su propósito es evitar que el usuario presione otro botón mientras una animación se está ejecutando en la aplicación.

Práctica 11: Animación de elementos de una aplicación

Su implementación consta de una repetición del 1 al 6, en 3 donde cada repetición desactiva un botón con la id “b” & x, es decir, b1, b2, b3, hasta el 6. Para ello es necesario nombrar cada botón con una id de b1 hasta b6.

```
1 on desactivarBotones
2   repeat with x=1 to 6
3     disable button ("b" & x)
4   end repeat
5 end desactivarBotones
```

Figura 11.11. Command handler “desactivarBotones”

Activar botones

Es llamado al término de cada *handler*. Su función es reactivar los botones que fueron desactivados con el *handler* `desactivarBotones`. El código es idéntico al anterior, a excepción del comando **disable**, que es reemplazado por `enable`.

```
1 on activarBotones
2   repeat with x=1 to 6
3     enable button ("b" & x)
4   end repeat
5 end activarBotones
```

Figura 11.12. Command handler “activarBotones”

Practica 12. Manejo de notificaciones locales

Objetivo: Realizar una aplicación que envíe notificaciones locales al dispositivo Android.

Nivel de dificultad: Media.

1. Cree un nuevo stack con el nombre de “Recordatorios”.
2. Arrastre 2 labels, 2 buttons, 1 check button y un scrolling field en el stack.
3. Acomode los elementos de manera que se vean como en la Figura 12.1.



Figura 12.1. Aspecto del stack “Recordatorios”

Para obtener un aspecto similar al mostrado en la Figura 12.1 se aconseja navegar por los menús del **Inspector** en cada objeto. Para cambiar los colores del texto y el fondo, se utiliza el menú **Colors & Patterns**. Los bordes del objeto en algunos casos se establecen en el menú **Basic Properties** y en otros casos en el menú **Icons & Border**. Dependiendo del objeto que utilice, es posible que estén disponibles ambos menús.

En cuanto al degradado de fondo, existen dos opciones: conseguir una imagen personalizada del degradado, crear una nueva tarjeta y referenciarla desde ahí (puede consultar la Práctica 8 para ver un ejemplo de esta acción); o arrastrar un rectángulo y consultar el menú *Gradient* del Inspector.

4. Seleccione el stack.
5. Escriba el código señalado en la Figura 12.2.

Práctica 12: Manejo de notificaciones locales

```
1  global gHoraNotificacion
2
3  on establecerHora
4      mobilePickDate "time"
5      put the result into gHoraNotificacion
6      if gHoraNotificacion is 0 then
7          answer "La hora no fue especificada"
8      Else
9          convert gHoraNotificacion to seconds
10         end if
11 end establecerHora
12
13 on crearRecordatorio
14     local tMsjNotificacion, tTitulo, tMensajeAlerta, tSonido
15     put field "fieldMsj" into tMsjNotificacion
16     put "Nuevo recordatorio" into tTitulo
17     put field "fieldMsj" into tMensajeAlerta
18     put the hilite of button "bSonar" into tSonido
19     mobileCreateLocalNotification tMsjNotificacion, \
20     tTitulo, tMensajeAlerta, gHoraNotificacion, tSonido
21 end crearRecordatorio
22
23 on localNotificationReceived pMensajeAlerta
24     answer "Nuevo Recordatorio:" && quote & \
25     pMensajeAlerta & quote with "Aceptar"
26 end localNotificationReceived
```

Figura 12.2. Código del stack “Recordatorios”

6. Aplique los cambios del script.
7. Seleccione el botón b1 y llame al comando “establecerHora” en su código.
8. Repita el mismo procedimiento con el botón b2 y el comando “crearRecordatorio”.
9. Opcional: agregue el código de la Figura 12.10, para acomodar los elementos de la aplicación en cualquier tamaño de pantalla.
10. Guarde el proyecto como “Recordatorios”.
11. Configure la aplicación. Haga clic en **File > Standalone Application Settings** y despliegue el apartado de Android.
12. En la opción **Status Bar Icon** agregue la imagen que se utilizará en la notificación de la aplicación (ver Figura 12.3). En este ejemplo se asignó como ícono la misma imagen utilizada en la Práctica 11.

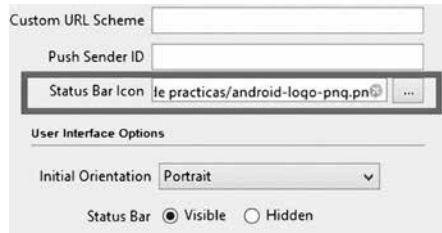


Figura 12.3. Asignando imagen de notificación de la aplicación

13. Ejecute la aplicación.

14. Cree el recordatorio (ver Figura 12.4).

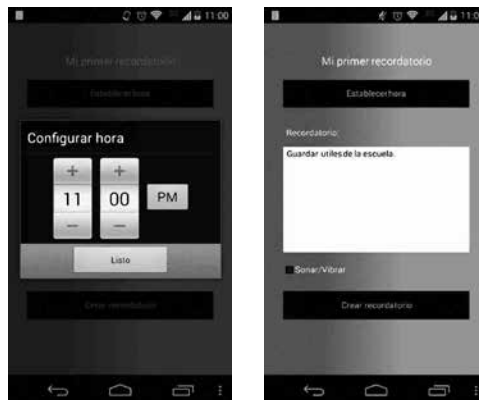


Figura 12.4. Creación del recordatorio.

Si el usuario no utiliza la aplicación a la hora del recordatorio, el recordatorio aparecerá como notificación en la barra de notificaciones (Figura 12.5, inciso a). De lo contrario, el recordatorio aparecerá en un cuadro de diálogo, tal como se muestra en el inciso b).

Práctica 12: Manejo de notificaciones locales

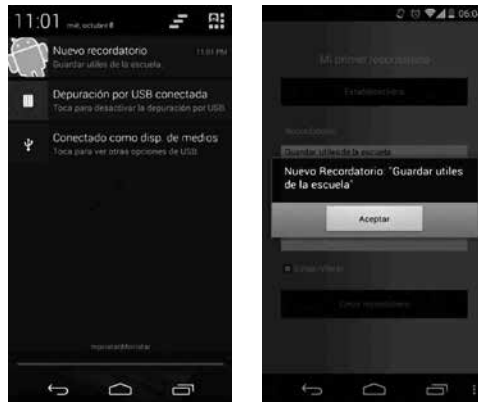


Figura 12.5. Variantes de la notificación

Explicación de la práctica 12

Para el mejor entendimiento del código utilizado en esta práctica, se explicará cada *handler* utilizado en el código de la Figura 12.2.

```
1 global gHoraNotificacion
```

Figura 12.6. Declaración de variable global

El código de la Figura 12.6 hace referencia a la línea 1 de la Figura 12.2; éste declara una variable global llamada “gHoraNotificacion”. Este tipo de variables puede utilizarse desde cualquier objeto que conforma el *stack* de la aplicación sin que pierda su valor.

La variable es declarada antes de todos los *handlers*. Si ésta se crea dentro de uno en específico, será necesario declararla en cada uno de los *handlers* como global. De lo contrario, el *handler* tratará a la variable como si fuese local, aunque ésta haya sido declarada como global en otro *handler*. [Nota: si la aplicación requiere múltiples variables globales, es recomendable que busque alternativas a ellas. Para más información al respecto puede ver el apartado “Tips para nombrar variables”].

establecerHora

Su objetivo es pedir la hora del recordatorio al usuario. Posteriormente, la hora es convertida en segundos para utilizarla en el *handler* “crearRecordatorio”, el cual se explicará más adelante.

Cabe destacar que el comando `mobilePickDate` (línea 2, Figura 12.7) se utiliza para llamar al menú que permite elegir fechas. Éste sólo se encuentra disponible para iOS y Android. El parámetro `time` especifica que sólo se obtendrá la hora.

```
1 on establecerHora
2   mobilePickDate "time"
3   put the result into gHoraNotificacion
4   if gHoraNotificacion is 0 then
5     answer "La hora no fue especificada"
6   Else
7     convert gHoraNotificacion to seconds
8     end if
9 end establecerHora
```

Figura 12.7. Command handler “establecerHora”

crearRecordatorio

La función de este *handler* se resume en tomar los valores necesarios para la creación de la notificación local. Estos son delimitados por el comando `mobileCreateLocalNotification`.

RunRev Ltd (2014d), especifica que los parámetros que se necesitan para utilizar este comando son:

- **Cuerpo de la alerta.** Es el texto que se mostrará en la barra de notificaciones, cuando la aplicación no está ejecutándose. Este valor se especifica en la línea 3 de la Figura 12.8.
- **Mensaje de la alerta.** En Android, es el mensaje que se muestra en la barra de estado, cuando la aplicación no está ejecutándose. Este valor se especifica en la línea 4 de la Figura 12.8. En este caso, se colocó el mensaje que el usuario escribió en el *field*.
- **Alerta enviada.** Es el mensaje que es guardado por la aplicación, y puede ser mostrado al usuario por medio de un *message handler* que

procese el mensaje `localNotificacionReceived`; éste último es descrito más adelante. En la línea 5 se puede observar el valor que le es asignado a este parámetro.

- **Hora de la alerta.** Es la hora en la que la alerta es enviada a la aplicación. Este parámetro especifica el número de segundos desde el Epoch Unix. Epoch 0 es enero 1 1970 00:00:00 GMT (Epoch Converter, 2014). El valor de este parámetro se encuentra indicado en la variable global “`gHoraNotificacion`” que se obtuvo del *handler* “`establecerHora`”.
- **Sonido.** Es una variable booleana que indica si se reproducirá un sonido o no al momento de recibir la notificación. La línea 6 de la Figura 12.8 especifica el valor que se asignó para este parámetro.
- **Valor del ícono.** Es un valor que especifica la imagen que tendrá la notificación: si el valor es de 0, oculta la imagen; pero si es mayor que 0, muestra el valor de la imagen en la notificación. Éste último valor es opcional y en este ejemplo no se especificó.

```
1 on crearRecordatorio
2   local tMsjNotificacion, tTitulo, tMensajeAlerta, tSonido
3   put field "fieldMsj" into tMsjNotificacion
4   put "Nuevo recordatorio" into tTitulo
5   put field "fieldMsj" into tMensajeAlerta
6   put the hilite of button "bSonar" into tSonido
7   mobileCreateLocalNotification tMsjNotificacion, tTitulo,\
8   tMensajeAlerta, gHoraNotificacion, tSonido
9 end crearRecordatorio
```

Figura 12.8. Command handler “`crearRecordatorio`”

`localNotificacionReceived`

Es un *message handler* que procesa el mensaje `localNotificacionReceived`. Éste incluye un parámetro que contiene el mensaje que le fue establecido en la línea 5 de la Figura 12.8. En este caso, se trató el mensaje con un cuadro de diálogo, de manera que cuando el usuario abre la aplicación, éste ve un mensaje con el recordatorio y el botón de “Aceptar”.

```
1 on localNotificationReceived pMensajeAlerta
2   answer "Nuevo Recordatorio:" && quote & pMensajeAlerta & \
3     quote with "Aceptar"
4 end localNotificationReceived
```

Figura 12.9. Message handler "localNotificationReceived".

preOpenStack (Opcional)

Este *message handler*, como se ha visto en prácticas anteriores, se utiliza para ajustar el tamaño de los elementos del *stack* y acomodarlos antes que la aplicación sea visible ante el usuario. El código en la Figura 12.10 se basa en los principios explicados en la Práctica 10, por lo que se omite la explicación de éste.

```
1 on preOpenStack
2   local tAlturaStack, tAnchuraStack, tPuntoMedio, \
3     tlocL1, tLocb1, tLocField, tAlineacionField, tLocL2, \
4     tLocb2, tLocCheck
5   set the rect of this stack to the screenRect
6   set the rect of graphic "rectanguloFondo" to the screenRect
7   put the height of this stack into tAlturaStack
8   put the width of this stack into tAnchuraStack
9   put (tAnchuraStack/2) into tPuntoMedio
10  //ubicacion del label1 "Mi primer recordatorio"
11  put tPuntoMedio & "," & (tAlturaStack*0.10) into tlocL1
12  set the loc of field "label1" to tlocL1
13  //ubicacion de b1 "Establecer hora"
14  put tPuntoMedio & "," & (tAlturaStack*0.20) into tLocb1
15  set the loc of button "b1" to tLocb1
16  set the height of button "b1" to (tAlturaStack*0.08)
17  //ubicacion del fieldMsj
18  put tPuntoMedio & "," & (tAlturaStack*0.50) into tLocField
19  set the loc of field "fieldMsj" to tLocField
20  set the height of field "fieldMsj" to (tAlturaStack*0.30)
21  //ubicacion label2 "Recordatorio:"
22  put the width of field "fieldMsj" into tAlineacionField
23  set the width of field "label2" to tAlineacionField
24  put tPuntoMedio & "," & (tAlturaStack*0.32) into tLocL2
25  set the loc of field "label2" to tLocL2
26  //ubicacion boton bSonar
27  set the width of button "bSonar" to tAlineacionField
28  put tPuntoMedio & "," & (tAlturaStack*0.70) into tLocCheck
29  set the loc of button "bSonar" to tLocCheck
```

Práctica 12: Manejo de notificaciones locales

```
30 //ubicacion boton b2
31 put tPuntoMedio & "," & (tAlturaStack*0.80) into tLocb2
32 set the height of button "b2" to (tAlturaStack*0.08)
33 set the loc of button "b2" to tLocb2
34 end preOpenStack
```

Figura 12.10. Message handler “preOpenCard” para alinear los elementos de la aplicación

Diccionario de LiveCode

A continuación, se enumeran las palabras en negritas que se encuentran en los códigos de cada una de las prácticas. La información siguiente es la traducción de las descripciones de dichas palabras, realizadas por RunRev Ltd (2014d) para la API 6.0.2 de LiveCode

Palabra	Tipo	Sintaxis	Función	Introducido
Answer	Comando	answer [TipoDeIcono] mensaje [with boton [o botones]] [titled tituloVentana] [as sheet]	Obtiene información o una confirmación del usuario antes de continuar con el programa. El usuario debe dar clic a uno de los botones para cerrar el cuadro de diálogo.	Figura 1.5
Ask	Comando	ask [TipoDeIcono] pregunta [with Respuesta Predeterminada] [titled TituloVentana] [as sheet]	Utilizado cuando un <i>handler</i> necesita obtener información del usuario antes de continuar.	Figura 2.1
Beep	Comando	beep [numeroDeVeces]	Obtiene la atención del usuario.	
Command	Estructura de control	Es el sinónimo de la palabra on , por tanto, la sintaxis de ambas estructuras de control es la misma.	Implementa un <i>comando</i> personalizado o procesa un <i>mensaje</i> .	
Disable	Comando	Disable <i>objeto</i>	Evita que un objeto responda al tecleo o a clics.	Figura 11.11
Enable	Comando	Enable <i>objeto</i>	Habilita un controlador para que responda a las acciones del usuario.	Figura 11.12

End	Palabra clave	end funcion	Termina un <i>on</i> , una <i>función</i> , o una estructura <i>setProp</i> o <i>getProp</i> .	
Function	Estructura de control	function NombreFuncion [parámetros] declaraciones end NombreFuncion	Sirve para implementar una <i>función</i> personalizada.	
Global	Comando	Global <i>listaDeVariables</i>	Define una variable que puede ser utilizada en cualquier <i>handler</i> , que conserva su valor entre ellos.	Figura 12.6
If	Estructura de control	If <i>condición</i> then <i>sentencias</i> [else <i>sentenciasElse</i>]	Ejecuta las sentencias sólo bajo ciertas circunstancias.	Figura 12.7
Local	Comando	local <i>ListadeVariables</i>	Define una variable local para un <i>handler</i> o <i>script</i> , que puede ser compartida entre todos los <i>handlers</i> del <i>script</i> .	Figura 12.8
mobileControl Create	Comando	mobileControlCreate <i>tipoDeControl</i> [, <i>nombre</i>]	Sirve para manipular algunos controles (vistas) en iOS y Android.	Figura 7.1
mobileControlDo	Comando	mobileControlDo <i>idONombre, accion, ...</i> <i>parámetros específicos</i> <i>de acción.</i>	Ejecuta ciertos comportamientos de controles nativos creados con el comando <i>mobileControl Create</i> .	Figura 7.1
mobileControlSet	Comando	mobileControlSet <i>idONombre, propiedad,</i> <i>valor</i>	Establecer propiedades de controles (vistas) en móviles.	Figura 7.1
mobileCreate LocalNotification	Comando	mobileCreateLocalNotification <i>cuerpoAlerta,</i> <i>mensajeAlerta,</i> <i>alertaEnviada,</i> <i>horaAlerta, sonido</i> [, <i>valorIcono</i>]	Programa una notificación local.	Figura 12.8

mobilePickDate	Comando	mobilePickDate [estilo] [,horaInicio] [,valorMinimo] [,valorMaximo]	Permite al usuario seleccionar la fecha o la hora, con el selector de hora nativo del dispositivo móvil.	Figura 12.7
Move	Comando	Move object {[from startLoc] to endLoc to pointList relative] motion} \ [in time] [without {messages waiting}]	Mueve controladores o ventanas a través de la pantalla.	Figura 11.9
On	Estructura de control	on nombreDelMensaje [lista de parámetros] [lista de declaraciones] end nombreDelMensaje	Es utilizada para implementar un <i>comando</i> personalizado o para atender un <i>mensaje</i> .	
Put	Comando	put valor [{before into after} contenedor]	Usado para establecer el valor de una variable, poner texto en un campo, poner datos en un archivo, mostrar texto en un cuadro de diálogo, o subir un archivo a un servidor.	Figura 2.1
Repeat	Estructura de control	Repeat <i>tipoDeBucle</i> declaraciones end repeat	Ejecuta el mismo conjunto de acciones a cada miembro de un grupo: cada tarjeta en un <i>stack</i> , o cada línea en una variable.	Figura 10.5
Return	Estructura de control	return valor	Devuelve un valor de una <i>función</i> personalizada o un <i>getProp handler</i> . También devuelve un mensaje de error de un <i>message handler</i> o <i>setProp handler</i> .	
Set	Comando	set [the] propiedad [of objeto] to valor	Asigna el valor a una propiedad.	Figura 5.4

Then	Palabra clave		Separa la condición de las sentencias que son ejecutadas si la condición es verdadera.	Figura 12.7
Wait	Comando	Wait [for] número [seconds ticks milliseconds] [with messages] Wait {until while} condicion [with messages]	Espera una cierta cantidad de tiempo, o un suceso, antes de continuar un <i>handler</i> .	Figura 11.6

Tips para añadir variables

En programación, una variable es un espacio en memoria donde se puede guardar un dato en específico para poder utilizarlo en la aplicación. Los tipos de variables en LiveCode son: globales, locales, parámetros, constantes y propiedades personalizadas.

A lo largo de las 12 prácticas se utilizaron tres tipos de variables. Para mayor referencia sobre los demás tipos que no se usaron, visitar el sitio web de LiveCode:

- **Globales.** Pueden ser accedidas desde cualquier elemento de la aplicación. No obstante, tienen que ser declaradas para ser utilizadas. Si el *script* de un objeto modifica su valor, cuando sea llamado nuevamente conservará dicho valor. Son declaradas con el comando `global`.
- El uso de múltiples variables globales no es recomendado. En programas complejos, su utilización puede llevar a problemas de depuración. Si múltiples objetos modifican el valor de una variable global y ésta no alcanza el valor deseado, el programador deberá encontrar de qué *script* proviene el problema. El nivel de complejidad de dicha tarea aumenta cuando se utilizan múltiples variables globales.
- **Locales del *handler* o temporales.** Se conocen como variables locales del *handler* porque sólo pueden ser utilizadas en el *handler* en que son declaradas. Éstas pierden su valor al momento en que el *handler* termina de ser ejecutado, por lo que también son conocidas como variables *temporales*. Se utiliza el comando `local` en la primera línea del *handler* para declararlas.
- **Locales del *script*.** Al igual que las variables temporales, son declaradas mediante el comando `local`. La diferencia radica en que éstas se declaran al inicio del *script* del objeto y pueden ser utilizadas por todos los *handlers* que integran dicho *script*.

Tips para nombrar variables

- Nombrar de forma consistente las variables, proporciona las siguientes ventajas: el código es fácil de entender, facilita la depuración del programa, hace legible el código y, además, ayuda a que otras personas puedan entenderlo (RunRev Ltd, 2010).

Por ello, se considera importante mencionar la forma en que RunRev Ltd, 2010 nombra las variables de sus scripts como ejemplo en las tutoriales. Éstas se pueden apreciar en la siguiente tabla:

Carácter	Ejemplo	Uso
G	GVar	Variables globales
T	tVar	Variables locales de un <i>handler</i>
S	sVar	Variables locales del <i>script</i>
P	pVar	Parámetros
K	kVar	Constantes
C	cVar	Propiedades personalizadas

Para verificar si las variables han sido declaradas previamente en el código, el usuario puede hacer clic en la opción **Edit > Variable checking** que se encuentra en la ventana de edición de código (**Code**). Si el usuario utiliza variables sin declarar mientras esta opción se encuentra activada, el código será marcado como incorrecto.

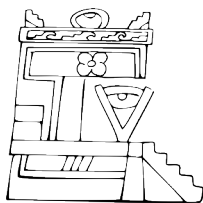
Conclusiones

Al finalizar las prácticas presentadas en este material educativo, se puede observar que LiveCode es un lenguaje de alto nivel y con características similares a la forma en que el humano se comunica. Puede ser utilizado para acrecentar la creatividad y lógica, mediante la creación de aplicaciones multiplataforma. Este material educativo se enfoca en la creación de aplicaciones en Android. No obstante, los usuarios de LiveCode pueden crear aplicaciones web y aplicaciones para Windows, MacOS, iOS y Android. En este material se presentaron 12 prácticas que permiten a cualquier persona interesada en el tema, aprender a realizar sus propias aplicaciones en LiveCode.

Esperamos que este libro “LiveCode para Android a través de ejemplos: nivel básico”, sea de utilidad para las personas que tengan el interés de adentrarse en el mundo de la creación de aplicaciones para computadoras de escritorio o dispositivos móviles.

Bibliografía

- Pruszko, C. (2011). *LiveCode Basics*. Recuperado el 22 de Febrero de 2014, de Learning to Create Games, Apps and Programs: <https://sites.google.com/a/pgcps.org/livecode/>
- Designer Pics. (2016). *Free Images*. Recuperado el 20 de Enero de 2016, de <http://www.designerspics.com/>
- Epoch Converter. (s.f.). *Convertir Epoch – Tiempo Unix*. Recuperado el 8 de Octubre de 2014, de EpochConverter: <http://espanol.epochconverter.com/>
- LiveCode Ltd. (2010). *LiveCode UserGuide*.
- LiveCode Ltd. (2013). *Open Language*. Recuperado el 25 de Febrero de 2014, de <http://livecode.com/blog/2013/02/03/open-language/>
- LiveCode Ltd. (2014). *API (Language Dictionary)*. Recuperado el 15 de Marzo de 2014, de <http://livecode.com/developers/api/6.0.2/>
- LiveCode Ltd. (2014a). *Mobile Guide*. Recuperado el 25 de Febrero de 2014, de <https://livecode.com/resources/guides/mobile/>
- LiveCode Ltd. (2014b). *LiveCode Beginners Guide*. Recuperado el 25 de Febrero de 2014, de <https://livecode.com/resources/guides/beginners-guide/>
- LiveCode Ltd. (2014c). *Getting Started with LiveCode Development*. Obtenido de LiveCode Lessons: <http://lessons.runrev.com/m/4603>



Difusión y Divulgación
Científica y Tecnológica

José Manuel Piña Gutiérrez
Rector

Arturo Díaz Saldaña
Secretario de Investigación, Posgrado y Vinculación

Andrés González García
Director de Difusión y Divulgación Científica y Tecnológica

Francisco Morales Hoil
Jefe del Departamento Editorial de Publicaciones No Periódicas

Esta obra se terminó de editar el 31 de agosto de 2017, en versión electrónica para consulta en el Catálogo de Publicaciones Científicas de la Universidad Juárez Autónoma de Tabasco. El cuidado estuvo a cargo de los autores y del Departamento Editorial de Publicaciones No Periódicas de la Dirección de Difusión y Divulgación Científica y Tecnológica de la UJAT.