Capítulo 6

Aplicación práctica I

Contenido

1. Introducción
2. Descripción de la aplicación
3. Diseño de la interfaz gráfica. Elección y configuración de componentes
4. Desarrollo de la lógica. Definir el comportamiento
5. Resumen

1. Introducción

Después de conocer al detalle las características de *MIT App Inventor 2,* su interfaz de desarrollo, los conceptos fundamentales sobre programación en los que se basan los bloques con los que se construirán las aplicaciones, además de ver el ejemplo de una primera aplicación inicial, este capítulo se va a centrar en el desarrollo de una aplicación real más avanzada.

En este caso, se completará una aplicación totalmente funcional haciendo uso y poniendo en práctica en ella tanto los conceptos básicos como los componentes que se han estudiado en unidades anteriores.

En los siguientes apartados se irá descubriendo el objetivo que se pretende con dicha aplicación, así como el desarrollo que requiere tanto a lo que a interfaz gráfica se refiere como a la lógica o comportamiento de la misma.

2. Descripción de la aplicación

La aplicación que se pretende realizar es un juego en el que se contabilizarán el número de veces que una foca es capaz de comer pescado en el intervalo de tiempo de diez segundos. Esta tratará de cazar dos peces, que se irán moviendo por las diferentes zonas de la pantalla, y el usuario deberá desplazar la foca hasta ellos con el fin de tocarlos y alimentarla. Al finalizar el tiempo, se mostrará en pantalla el aviso de que la partida ha finalizado.

La aplicación consta de dos pantallas: una primera, de presentación, que se presta a pantalla de bienvenida; y otra segunda, para la pantalla del propio juego.

La primera de ellas mostrará una pantalla de bienvenida al usuario con una imagen que, al tocarla, dará paso a la pantalla de juego.

En la segunda se aprecia un panel que simula la piscina en la que se encuentran la foca y los peces. Además, hay dos botones: el primero de ellos, para comenzar la partida, que además establecerá los contadores de tiempo y puntuación a cero; y el segundo, para pausarla y reanudarla.

Con esta breve introducción al objetivo del juego se puede empezar a deducir qué componentes se requerirán para su desarrollo. Es lógico pensar que cada desarrollador podrá esbozar mentalmente una interfaz gráfica diferente para la aplicación, con lo que podrá alterar el aspecto de la misma respecto a la solución que se le propone a continuación. Se recomienda que primero se sigan fielmente los pasos que se exponen y, una vez terminado el desarrollo y comprendidos todos los pasos, se modifique a antojo con el fin de investigar y practicar con *MIT App Inventor 2.*

Se comenzará creando el nuevo proyecto para tenerlo todo listo al comienzo del siguiente paso. A continuación, se resume el proceso.

Acceder desde el navegador a la dirección de *MIT App Inventor 2* que, como se aprendió en unidades anteriores, es <http://appinventor.mit.edu/>, elegir la opción **Create** e introducir los datos de usuario de la cuenta personal de Google. Una vez dentro de la aplicación, hacer clic sobre la opción de menú **Projects** y elegir **Start new projects…** Como nombre para la aplicación, se puede elegir *unidad6*.

El proyecto contiene inicialmente una pantalla y se añadirá una segunda. Para ello, hacer clic sobre el botón **Add screen…** y dejar el nombre **Screen2** que tiene por defecto. Así se diferenciará a **Screen1,** como la pantalla de bienvenida, y a **Screen2,** como la pantalla de juego.

3. Diseño de la interfaz gráfica. Elección y configuración de componentes

El primer paso del desarrollo de la aplicación será la definición del aspecto que tendrá la aplicación. En este se determinarán los componentes que formarán parte de ella y que serán elegidos en función de los requisitos de la misma. En segundo lugar, deberán establecerse las propiedades iniciales de cada componente con el fin de adaptarlo al diseño establecido. En el apartado siguiente, se definirá la lógica y comportamiento de la aplicación a partir de estos componentes.

Nota

Para el diseño de la interfaz gráfica de la aplicación se hará uso de una serie de recursos de imagen que se podrán descargar de la URL <http://www.recursosadicionales.com/mit/ud6.rar>. Con esto se conseguirá adaptar el desarrollo de la forma más fiel a la explicación.

Para seleccionar los componentes adecuados se debe leer al detalle la descripción de la aplicación con el fin de desglosarla y extraer la máxima información. Cuanta más información se extraiga más fácil será determinar qué componentes son necesarios en la aplicación. Como se verá en adelante, se elegirá algún componente no visible que, aunque no forme parte de la interfaz gráfica, será necesario para el correcto funcionamiento de la aplicación.

3.1. Pantalla de bienvenida

La aplicación se iniciará en esta pantalla y en el panel **Components** se identificará como *Screen1*. Se debe seleccionar y, en primer lugar, establecer la propiedad **AlignHorizontal** y **AlignVertical** con el valor **Center,** para que los componentes que se añadan a continuación en ella se alineen en el centro del eje horizontal y vertical. Se otorgará a la aplicación un icono para identificarla entre las apps que se tengan instaladas en el dispositivo *Android.* Se debe elegir el archivo "icon.png" desde la propiedad **Icon,** además de la propiedad **ScreenOrientation** a **Portrait** para que mantenga la orientación vertical de pantalla. Desmarcar la opción **Scrollable** para que no aparezca una barra de desplazamiento vertical y, por último, añadir como **Title** el valor *Unidad 6*.

Los componentes de esta pantalla serán pocos y sencillos. En primer lugar, se añadirá un componente **VerticalArrangement** para el que se definirá a **Center** tanto la propiedad **AlignHorizontal** como **AlignVertical.** Para las dimensiones, elegir **Fill parent** en la propiedad **Width** y **Height** para que se ajuste al ancho y alto de la pantalla del dispositivo.

Acto seguido, se colocará un componente **Button,** en el que se cargará la imagen de inicio. Hay que comenzar ahora subiendo los recursos de imagen que se han descargado desde la URL anteriormente especificada, con el objeto de poder usarlos en los componentes de la aplicación. Después de subirlos, hacer clic sobre la propiedad **Image** y elegir el fichero "inicio.png". Eliminar el texto del botón desde la propiedad **Text** para que no aparezca nada y se vea la imagen correctamente. Las propiedades **Width** y **Height** se dejarán con los valores **Automatic** por defecto. Tocando esta imagen se dará paso a la pantalla de juego. Este comportamiento se definirá en el siguiente apartado. El nombre elegido para la aplicación es *Hungry Seal,* como puede verse en la imagen cargada en el botón, que en español significa *La foca hambrienta*. Posteriormente, se elegirá a través del editor de bloques un código de color RGB adecuado como color de fondo. Las propiedades **Width** y **Height** se dejarán con los valores **Automatic** por defecto.

Después de añadir todos los componentes necesarios, se van a renombrar todos para que después sea más fácil identificarlos desde el editor de bloques. Seleccionar cada componente y renombrarlo haciendo clic sobre el botón **Rename** desde el panel **Componentes.** En la siguiente tabla se encontrará la relación de nombres para cada componente.

Componentes de la pantalla de bienvenida

Componente	Descripción
VerticalArrangement1	Distribución vertical de los componentes
btnInicio	Botón con la imagen de inicio

3.2. Pantalla de juego

Para la segunda de las pantallas que contiene al juego en sí, y con el fin de organizar en ella todos los componentes, es recomendable ubicarlos dentro de componentes de disposición de pantalla o **Layout.** Se añadirá en primer lugar un componente **VerticalArrangement,** que englobará a todos los componentes.

Dentro de este se irán colocando los demás componentes de arriba hacia abajo con el orden correspondiente.

Del mismo modo que ocurrió con el componente **Screen1** correspondiente a la pantalla de bienvenida, selecciónese el componente **Screen2** y establézcase la propiedad **AlignHorizontal** con el valor **Center,** para que los componentes que se añadan a continuación en ella se alineen en el centro del eje horizontal, y la propiedad **ScreenOrientation** a **Portrait,** para que mantenga la orientación vertical de pantalla. Desmárquese la opción **Scrollable** para que no aparezca una barra de desplazamiento vertical.

Para el componente de disposición vertical, elegir **Center** para las propiedades **AlignHorizontal** y **AlignVertical**, y **Fill parent** para **Width** y **Height.** Con esto se ocupará todo el ancho y alto de la pantalla del dispositivo.

Recuerde

Para que las pantallas se adapten a la resolución del dispositivo, se debe desmarcar la opción *Scrollable* del componente Screen, para que no se muestre la barra de desplazamiento vertical.

El primero de los componentes que se colocará dentro de la disposición vertical será otro de disposición horizontal, **HorizontalArrangement,** para el que se establecerá a **Center** su propiedad **AlignHorizontal.** Obsérvese que no se puede cambiar a priori la propiedad **AlignVertical,** al estar inhabilitada. Para poder cambiar su valor se debe establecer la propiedad **Height** a **Fill parent.** Ahora se debe cambiar la propiedad **AlignVertical** a **Center** y, seguidamente, volver a establecer **Height** a Automatic. Además, se debe definir a **Fill parent** la propiedad **Width.**

Dentro de este componente de disposición horizontal, se ubicará una imagen con el nombre del juego a través de un logotipo y los indicadores de aciertos y

tiempo restante. Se debe comenzar añadiendo un componente **Image** y hacer clic en la propiedad **Picture** y elegir el archivo "titulo.png". Las propiedades **Width** y **Height** se definirán a **Fill parent** y **Automatic** respectivamente. Los componentes que indicarán los aciertos y tiempo restante se ubicarán dentro de otro componente **HorizontalArrangement,** que se debe añadir. Configurar la propiedad **AlignHorizontal** a **Center** y las propiedades **Width** y **Height** a **Fill parent** y **Automatic,** respectivamente. Dentro de él, añadir un componente **Image** y otro **Label,** tanto para los aciertos como para el tiempo. El primer **Image** contabilizará los aciertos y para él hacer clic sobre la propiedad **Picture** y elegir el archivo "aciertos.png". En cuanto a la imagen del tiempo, hacer lo propio eligiendo el fichero "reloj.png". Ambas imágenes tendrán **Automatic** como valor para **Width** y **Height.** Para las etiquetas correspondientes, se deberá dejar el valor de la propiedad **Text** de cada una en blanco, dado que inicialmente no se tiene ningún valor. En cuanto a la propiedad **TextAlignment,** establecer el valor **Center.** Seguidamente, establecer la propiedad **Width** con el valor **Fill Parent,** para que se distribuya el ancho de la pantalla por igual, y la propiedad **Height** se mantendrá en **Automatic.**

En la descripción se aprecia que la aplicación dispone de varios componentes, que se desplazan por la pantalla y cambian de posición, como son la foca y los peces. Para ello, lo más lógico es utilizar varios componentes **ImageSprite,** que definirán a cada uno de ellos, y un **Canvas,** que determinará su zona de movimiento simulando el agua de una piscina. Añádase primero el componente **Canvas** y, dentro de él, cada uno de los componentes **ImageSprite,** en concreto tres. Para el **Canvas** añadido, en primer lugar establecer las propiedades **Width** y **Height** a **Fill parent** para que rellenen el espacio tanto en anchura como en altura. Además, se debe elegir una imagen de fondo haciendo clic sobre la propiedad **BackgroundImage** y seleccionando el archivo "fondo.png" del listado. En cuanto a los componentes **ImageSprite,** se deben seleccionar de uno en uno y establecer la imagen correspondiente a cada uno de ellos. Para ello, hágase clic sobre la propiedad **Picture** y elíjase del listado los ficheros "foca.png", "pez1a.png" y "pez2a.png", respectivamente. No es necesario modificar ninguna propiedad más.

Además, nuestra aplicación precisará de dos botones que permitan tanto el inicio de la partida como poder pausarla cuando se requiera. Estos componentes serán de tipo **Button** y se podrán colocar bajo el **Canvas** que simula el

agua. Para organizar los botones y que se coloquen ordenadamente uno al lado del otro se ubicarán dentro de un componente **HorizontalArrangement.** Hacer clic en la propiedad **Image** de cada botón y seleccionar el archivo "botón.png" para proporcionar un aspecto a los botones. Para la propiedad **Text,** los valores **Comenzar** y **Pausa,** respectivamente, y el color de las letras, a través de la propiedad **TextColor,** se definirá con el valor **White.** Por último, establecer la propiedad **Width** con el valor **Fill parent** para cada botón.

Por otro lado, se debe realizar un control del tiempo dando por finalizada la partida a los diez segundos. El componente que permitirá definir este periodo de tiempo será **Clock.** Este componente definirá además los movimientos de los peces. Añádase a cualquier zona de la pantalla dado que será un componente no visible y se colocará en una zona reservada bajo la pantalla. Para simular el movimiento de los peces se utilizarán otras imágenes distintas a las usadas por defecto en los componentes **ImageSprite** y requerirán de otro componente de reloj que marque el tiempo cambio de imagen. Para ambos componentes, desmarcar la opción **TimerEnabled** para que inicialmente no estén habilitados sino que lo hagan cuando se haga clic sobre el botón **Comenzar,** y para el segundo de ellos, establecer a **800** la propiedad **TimerInterval.** Esto hará que parezca más real el cambio entre las imágenes que representan a los peces.

Para anunciar el fin de la partida se mostrará una notificación en la pantalla a través del componente **Notifier.** Añádase y no será necesario modificar ninguna de sus propiedades.

Por último, se va a añadir un componente que aportará un poco de realismo a la aplicación y que servirá para que, cada vez que la foca se coma un pez, el dispositivo *Android* vibre. En caso de usar el emulador, no tendrá efecto. Este componente es **Sound.** Añádase y no modificar tampoco, por el momento, ninguna de sus propiedades.

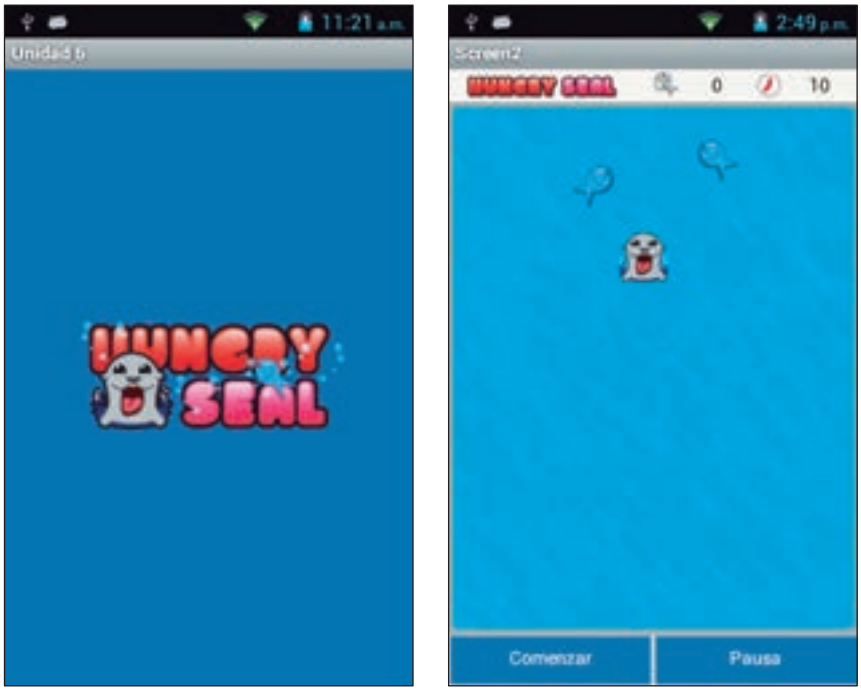
Es conveniente, tras añadir los componentes necesarios, renombrarlos con el objetivo de que después sea más fácil identificarlos desde el editor de bloques. Selecciónese cada componente y renómbrese haciendo clic sobre el botón **Rename** desde el panel **Componentes.** En la siguiente tabla se encontrará, a modo de resumen, la relación de nombres para cada componente utilizado

en la pantalla de juego de la interfaz gráfica de la aplicación por orden de aparición en cada una de las pantallas.

Componentes de la pantalla de juego

Componente	Uso
VerticalArrangement1	Distribución vertical de los componentes.
HorizontalArrangement1	Distribución horizontal de los componentes.
imgTitulo	Imagen para el título.
HorizontalArrangement2	Distribución horizontal de los componentes.
imgAciertos	Imagen Aciertos.
lblAciertosValor	Etiqueta que muestra los aciertos actuales.
imgTiempo	Imagen Tiempo.
lblTiempoValor	Etiqueta que muestra el tiempo restante.
cnvPiscina	Canvas que simula la piscina y contiene a la foca y los peces.
isFoca	ImageSprite que representa a la foca.
isPez1	ImageSprite que representa al pez 1.
isPez2	ImageSprite que representa al pez 2.
HorizontalArrangement3	Distribución horizontal de los componentes.
btnComenzar	Botón para comenzar la partida.
btnPausa	Botón para pausar/continuar la partida.
Clock1	Reloj para contabilizar el tiempo de partida y los movimientos de los peces.
Clock2	Reloj para cambiar el aspecto de los peces y simular movimiento.
Notifier1	Notificación de fin de partida.
Sound1	Provoca la vibración del dispositivo cuando la foca se come un pez.

Finalizada la elección y configuración de los componentes de ambas pantallas, la interfaz gráfica deberá ser similar a la que se muestra a continuación. Es posible que el dispositivo tenga una resolución o densidad de pantalla diferente y no se muestre tal cual se aprecia en las siguientes imágenes. Además, hay que tener en cuenta que los colores de fondo se establecerán a continuación en el siguiente apartado mediante códigos de color.



Interfaz gráfica

Actividades

1. Modifique la orientación de cada pantalla para observar el comportamiento de la interfaz gráfica en modo paisaje.

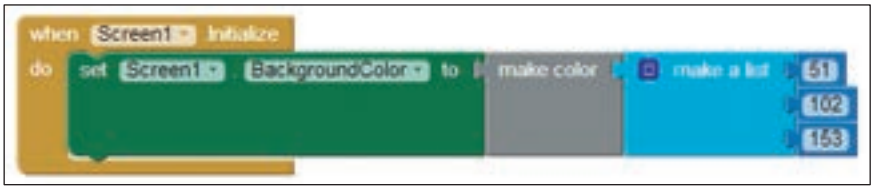
4. Desarrollo de la lógica. Definir el comportamiento

Una vez diseñada la interfaz gráfica, se podrá continuar con el desarrollo de la aplicación definiendo los bloques que determinarán el comportamiento de la misma a través de la interacción del usuario con los componentes que se han añadido a la interfaz gráfica. Se recomienda que, cuando se inicie el desarrollo de una aplicación, se establezcan en primer lugar todos los compontes gráficos de la interfaz antes de continuar con el desarrollo de la lógica de la aplicación. Esto evitará tener que hacer y rehacer gran cantidad del trabajo.

4.1. Pantalla de bienvenida

La mecánica de la primera pantalla es muy sencilla, al igual que ocurrió con el diseño de su interfaz. Se realizarán dos operaciones sobre ella. La primera es definir el color de fondo de la misma a través de un código de color RGB, y la segunda, añadir un evento "Click" sobre el botón para dar paso a la pantalla del juego.

Para la primera de las operaciones, se hará uso del bloque **Initialize** sobre el componente **Screen1** que realizará las acciones que dentro de él se definan. Añádase seleccionando dicho componente desde el panel **Blocks** y eligiendo el citado bloque. Dentro del mismo, incluir el método **BackGroundColor** aplicado al componente **Screen1.** Seguidamente se conectará a este método la función **make color** desde la categoría **Colors** y que permitirá definir el color mediante un código RGB. Establecer (51,102,153) como valores del código. De este modo, se podrá poner cualquier color que se necesite y no solo los que se proponen desde el panel **Properties** del Diseñador. Finalizada la configuración, el bloque deberá quedar como se observa a continuación.



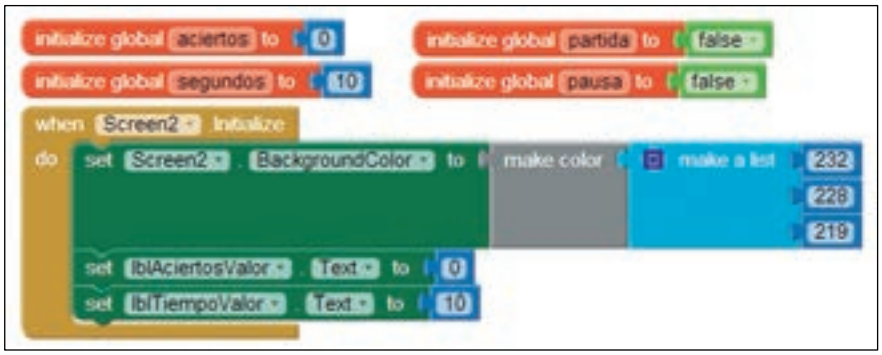
Evento Initialize del componente Screen1

En cuanto a la segunda de las operaciones, consta de un botón que, al pulsarlo, abrirá la segunda de las pantallas que contiene la aplicación, en concreto, la correspondiente a la pantalla de juego. Para ello, se deberá añadir un bloque de evento "Click" sobre el botón e incluir en él la instrucción **open another screen** y pasar como nombre de pantalla *Screen2*. Esta instrucción se encuentra en la categoría **Control.** El bloque quedará como se puede apreciar a continuación.

Evento "Click" del componente btnInicio

Pantalla de juego

La pantalla de juego contiene toda la lógica de la aplicación y, por ello, habrá que extenderse mucho más en comparación con la pantalla de bienvenida. En primera instancia, se deben declarar cuatro variables: dos para almacenar los valores de los aciertos y tiempo, y dos para los estados de juego como partida y pausa. Inicializar las dos primeras a **0** y **10** respectivamente, y a **false** las demás. Acto seguido se establecerá el color de fondo del mismo modo que para la pantalla anterior, solo que en este caso será sobre el componente **Screen2** y con el código de color RGB (232,228,219). Además, dentro del evento **Initialize,** se ha de establecer el texto que mostrarán las etiquetas de aciertos y tiempo inicialmente a **0** y **10** respectivamente a través de la instrucción **set Text to** de cada una. La declaración de variables y el evento **Initialize** quedará como se muestra a continuación.



Declaración de variables y evento Initialize

Actividades

2. ¿A través de qué bloque se lanzará otra pantalla que será definida en él mismo?

Se continúa definiendo el comportamiento que depende directamente del primer componente de reloj **Clock1.** Este componente establece a través del evento **Timer** que a cada intervalo de tiempo de un segundo o, lo que es lo mismo, 1000 milisegundos, se realicen las operaciones que en él se definan. Sin embargo, los intervalos de tiempo para el segundo reloj son de 800 milisegundos, algo menos de un segundo. Esto producirá una alternancia en la imagen de los componentes **ImageSprite** que da la apariencia a los peces a cada intervalo de tiempo definido. Para establecer las imágenes de cada pez se debe añadir la instrucción **set Picture to** de cada **ImageSprite** en el componente **Clock1,** y especificar cada uno los nombres de los ficheros "pez1a.png" y "pez2a.png". Seguidamente, se llamará a los métodos **MoveTo** de cada **ImageSprite** que representa a los peces con el objetivo de definir los movimientos aleatorios que realizarán estos. Para cada valor de entrada, *x* e *y* será de ayuda utilizar el bloque **random integer from valor inicial to valor final** que se encontrará haciendo clic sobre la categoría **Math** del panel "Blocks" y que devuelve un valor aleatorio de tipo entero. La posición se obtendrá desde un valor comprendido entre 0 y el ancho o alto del **Canvas** respectivamente para cada valor *x* e *y*. Con el objetivo

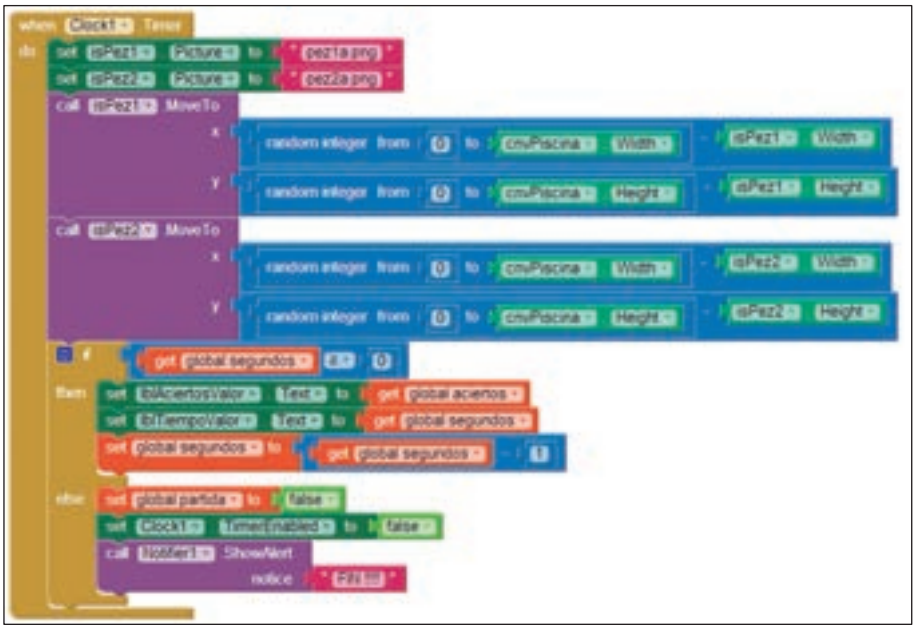
de evitar que los peces se muevan por posiciones incorrectas de la pantalla, es decir, fuera del **Canvas** que simula la piscina, se restará, al valor aleatorio obtenido, tanto la anchura como la altura del propio pez.

Recuerde

Reste la anchura y altura a cada valor aleatorio generado respectivamente que dan como resultado los valores de entrada x e y de las funciones *MoveTo*.

Además de lo definido hasta ahora como operaciones vinculadas a los intervalos definidos en el primer reloj, se deberá ir actualizando en el marcador tanto los aciertos actuales como el tiempo restante de partida. Esto se realizará comprobando a través de un bloque **if then else** si los segundos son mayores o iguales a 0. Hay que recordar que se inicializaron anteriormente con valor 10. En caso afirmativo, se establecerá el valor de aciertos y tiempo en su etiqueta correspondiente, además de ir decrementando en una unidad la variable que almacena los segundos y actualizándola. En caso contrario, se establecerá el valor lógico de **partida** a **false,** se parará el reloj asignando mediante la instrucción **set TimerEnabled to** también el valor **false,** y por último, se mostrará un mensaje que informe al usuario de que la partida ha acabado mediante un componente **Notifier.**

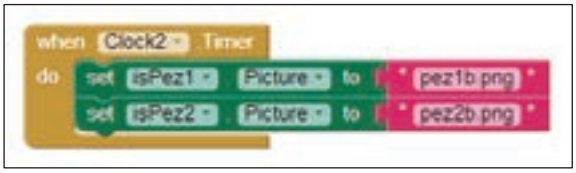
En la siguiente imagen, se podrá comprobar el aspecto del bloque de toda la lógica relativa al evento **Timer** correspondiente al primero de los componentes de reloj **Clock1.**



Evento Timer del componente Clock1

En cuanto al evento "Timer" correspondiente al segundo de los componentes de reloj *Clock2*, se establece otra imagen distinta a la actual a través de la sentencia ***set Picture to,*** "pez1b.png" y "pez2b.png" respectivamente para cada componente **ImageSprite** que simulará el movimiento de los peces al cambio entre ambas imágenes.

Será a continuación donde se podrá comprobar el aspecto del bloque que contiene toda su lógica.



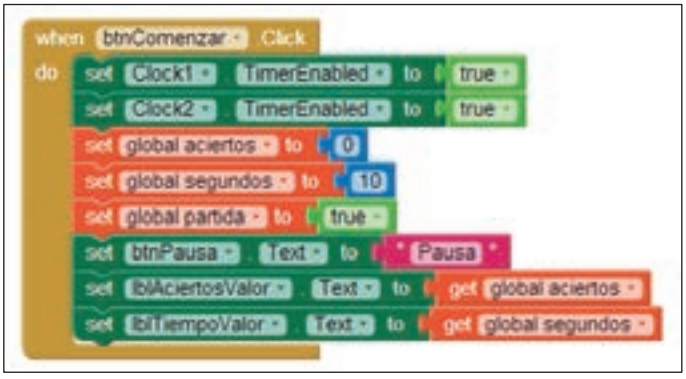
Evento "Timer" del componente Clock2

Seguidamente se definirá el comportamiento de cada uno de los botones con los que, por un lado, se dará comienzo a la partida y, por otro, se realizará una pausa. Como cada botón realizará su función al ser pulsado, se añadirá un bloque de evento "Click" para cada uno de ellos. En cuanto al conjunto de acciones que realizará el botón **Comenzar,** se añadirán en un primero momento los bloques que habilitarán la propiedad **TimerEnabled** de cada uno de los componentes de reloj. A continuación, se establecerán valores en las variables, como **0** para **aciertos, 10** para **segundos** y **true** para **partida.** Estos valores inicializarán los valores del marcador de la partida al pulsar el botón **Comenzar.** Por último, serán definidos los valores de la propiedad **Text** de las etiquetas a los valores establecidos anteriormente a las variables para que sean mostrados al usuario, además del texto del botón de pausa a la cadena **Pausa.**

Actividades

3. Investigue y pruebe las diferentes opciones del componente Notifier.

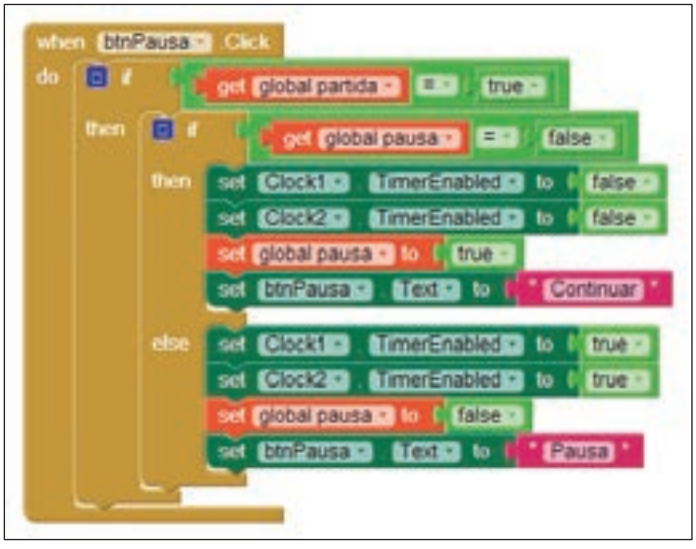
Toda la funcionalidad definida para el botón **Comenzar** la podrá observar en la siguiente imagen.



Evento "Click" del componente botón ***btnComenzar***

Para definir el comportamiento del botón **Pausa** se realizarán diversas comprobaciones mediante bloques **if condición then** e **if condición then else** con el objetivo de determinar si al pulsarlo se está participando en una partida y que esta no se encuentre en estado de pausa respectivamente. Para ello, y por este orden, se añadirán dichas condiciones en las que, solo en caso de cumplirse ambas, se realizará una pausa sobre la partida. Como se ha dicho, se deberán cumplir ambas condiciones, con lo que una vez evaluada como positiva la primera condición, se incluirá la segunda dentro de la zona de acciones **then** de esta. Una vez definidos ambos bloques de condiciones dentro del bloque de evento "Click" del botón de **Pausa** se pasarán a definir las sentencias correspondientes a los estados **en partida y pausa,** y **en partida y no pausa** respectivamente. Para el primero de ellos, se establecerá la propiedad **TimerEnabled** de cada uno de los componentes de reloj a **false** para parar el tiempo de la partida, se asignará **true** a la variable **pausa** y se modificará el texto del botón de **Pausa** a "Continuar" para proseguir con la partida posteriormente. En el segundo caso, se establecerá la propiedad **TimerEnabled** de cada uno de los componentes de reloj a **true** para reanudar el tiempo de la partida, se asignará **false** a la variable **pausa** y se volverá a definir el texto del botón de **Pausa** a "Pausa" para informar de que se puede volver a parar la partida.

En la siguiente imagen se encuentra la función definida para el botón **Pausa.**

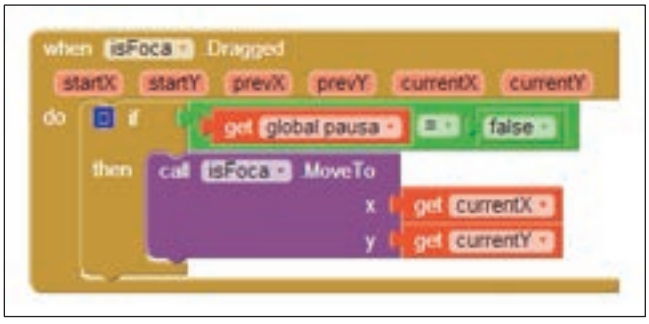


Evento "Click" del componente botón ***btnPausa***

Una vez definida la funcionalidad de los botones, se continuará con los elementos dinámicos del juego, como son la foca y los peces. Cada uno de ellos tiene un comportamiento particular: la foca será arrastrada por el usuario e irá a la caza de unos peces que se moverán arbitrariamente por la pantalla. En adelante, se definirán estos comportamientos, que se resumen en el movimiento de arrastre de la foca y la colisión con los peces.

El evento que permitirá arrastrar la foca es "Dragged", con lo que se hará uso del bloque **when Dragged do** para definir este comportamiento. Como se verá al añadirlo, este evento contiene unos parámetros de entrada que controlarán la posición inicial, actual y final de la foca durante el arrastre y que se usarán para definir el movimiento de la foca. En un primer momento, se comprobará que la partida no se encuentre en pausa, hecho que no permitirá que la foca sea arrastrada. En caso contrario, se podrá desplazar la foca y será a través del bloque de llamada a la función **MoveTo** al que se le pasarán las coordenadas actuales del movimiento de arrastre.

Toda la funcionalidad del movimiento de arrastre de la foca podrá comprobarse en la siguiente imagen.

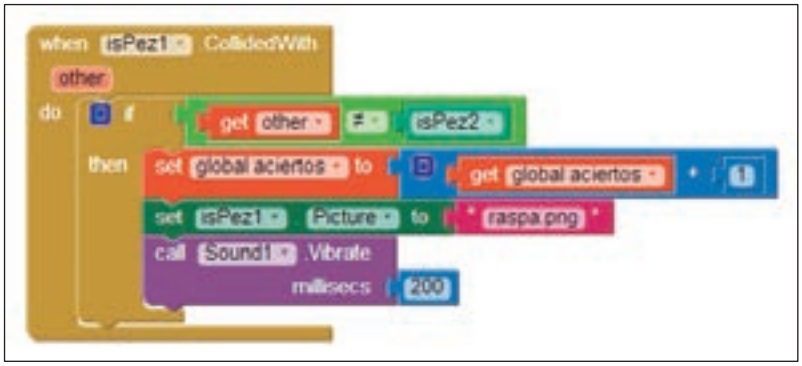


Evento "Dragged" del componente ***ImageSprite isFoca***

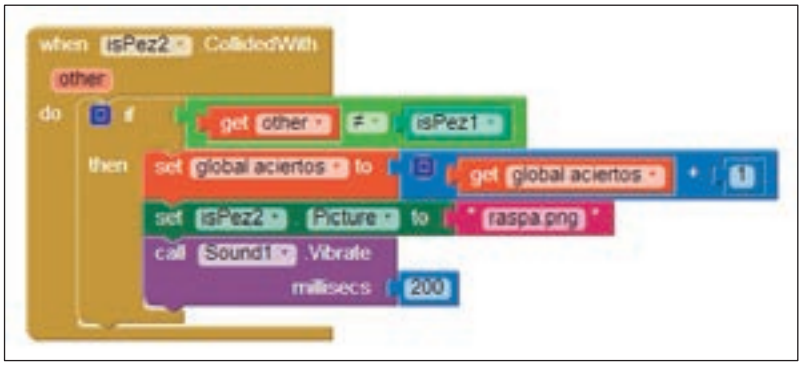
En cuanto a las colisiones, serán definidas para cada componente **ImageSprite** a través del evento "CollidedWith". En él se comprobará que el otro componente involucrado en la colisión de un pez no sea otro de los peces. En este caso no se quiere que esta colisión se detecte como acierto. Descartando este hecho, se tendrá que cualquier otra colisión será con la foca, al no haber

más componentes en el juego, en cuyo caso se incrementará el contador de los aciertos en una unidad, se alterará momentáneamente la apariencia del pez simulando una raspa de pescado, con la imagen "raspa.png" desde la instrucción **set Picture to,** además de provocar una pequeña vibración del dispositivo de 200 milisegundos a través del bloque de llamada a la función **Vibrate.**

Se deberá realizar la funcionalidad definida para cada uno de los peces, tal y como se muestra en las siguientes imágenes.



Evento "CollidedWith" del componente ***ImageSprite isPez1***



Evento "CollidedWith" del componente ***ImageSprite isPez2***

Recuerde

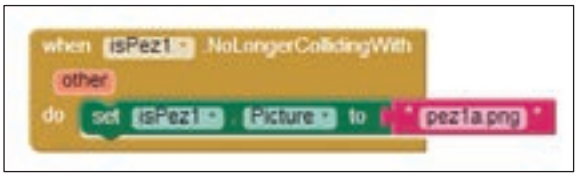
En el evento "CollidedWith" de cada componente *ImageSprite* se debe realizar la comprobación necesaria para que no se contabilicen como acierto las colisiones entre ambos peces.

Por último, y en relación a las colisiones, se deberá establecer de nuevo la apariencia de cada pez a su forma natural tras ser cazado por la foca y contabilizado el acierto por parte del usuario. Para ello se debe añadir un bloque con el método **NoLongerCollidingWith** para cada uno de los peces y establecer, a través de la instrucción **set Picture to,** el valor "pez1a.png" y "pez2a.png" respectivamente. Este sencillo evento detecta cuándo dos componentes se han separado, es decir, cuándo han dejado de chocar.

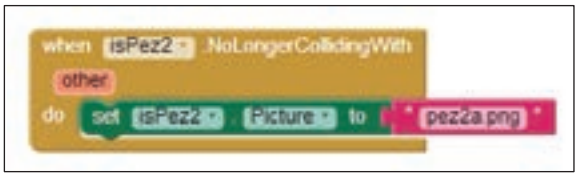
Actividades

4. Investigue y pruebe las diferentes opciones del componente Notifier.

El resultado se puede comprobar en las siguientes imágenes.



Evento "NoLongerCollidingWith" del componente ***ImageSprite isPez1***



Evento "NoLongerCollidingWith" del componente ***ImageSprite isPez2***

Con esto se ha terminado el desarrollo de la lógica de cada una de las pantallas de la aplicación. Ha quedado definida tanto la interfaz gráfica como la funcionalidad de cada pantalla y se puede dar por concluida la aplicación.

Nota

Desde la dirección <http://www.recursosadicionales.com/mit/ud6.apk> se puede descargar el fichero instalador de la aplicación que se acaba de realizar para comprobar en el dispositivo personal el resultado final.

El siguiente paso será probarla para comprobar que todo es correcto y se adapta a las especificaciones iniciales. Si no fuera así, habría que rehacer los pasos erróneos del desarrollo.

5. Resumen

La aplicación consta de dos pantallas, una primera de presentación que se presta a pantalla de bienvenida y otra segunda para la pantalla del propio juego. Cada una de ellas será un **Screen** diferente. Por defecto, el proyecto cuenta con uno y se deberá crear el segundo.

Para crear el proyecto, acceder desde el navegador a la dirección de *MIT App Inventor 2* que, como se aprendió en unidades anteriores, es <http://appinventor.mit.edu/>, elegir la opción **Create** e introducir los datos

de usuario de la cuenta personal de Google. Una vez dentro de la aplicación, hacer clic sobre la opción de menú **Projects** y elegir **Start new projects…** Como nombre para la aplicación puede elegir *unidad6*.

La primera de las pantallas mostrará una pantalla de bienvenida al usuario con una imagen que, al tocarla, dará paso a la pantalla de juego. En la segunda se aprecia un panel que simula la piscina en la que se encuentran la foca y los peces. Además hay dos botones, el primero de ellos para comenzar la partida, que además establecerá los contadores de tiempo y puntuación a cero, y el segundo para pausarla y reanudarla.

El primer paso del desarrollo de la aplicación será la definición del aspecto que tendrá la aplicación. En este se determinarán los componentes que formarán parte de ella y que serán elegidos en función de los requisitos de la misma. En segundo lugar, deberán establecerse las propiedades iniciales de cada componente con el fin de adaptarlo al diseño establecido.

Una vez diseñada la interfaz gráfica se podrá continuar con el desarrollo de la aplicación definiendo los bloques que determinarán el comportamiento de la misma a través de la interacción del usuario con los componentes que se han añadido a la interfaz gráfica. Se recomienda que, cuando se inicie el desarrollo de una aplicación, se establezcan en primer lugar todos los compontes gráficos de la interfaz antes de continuar con el desarrollo de la lógica de la aplicación. Esto evitará tener que hacer y rehacer gran cantidad del trabajo.

Por último, se deberá probar la aplicación con el objetivo de comprobar que todo ha salido acorde a las especificaciones iniciales.