Manejo de Sprites

En este capítulo veremos un juego que tiene por protagonistas a una mariquita, una pulga y una rana. Con la aplicación **Ladybug** el usuario podrá:

- Controlar el movimiento de una mariquita inclinando el teléfono.

- Comprobar su nivel de energía gracias a una barra que irá reduciendo su tamaño con el paso del tiempo o bien aumentando el nivel cuando la mariquita coma.

- Dirigir la mariquita para que se coma a la pulga y recupere energía.

- Evitar que la rana se coma la mariquita.

¿ Qué aprenderé ?

Antes de empezar la aplicación, debes de revisar todo el material que se vió durante el desarrollo de **"Dale Fuerte"** en el capítulo 3. Además, este capítulo 5 te enseñará a:

- Utilizar varios componentes **ImageSprite** para detectar choques entre ellos.
- Emplear el componente **OrientationSensor** para detectar el grado de inclinación del teléfono y recurrir a él para controlar un *ImageSprite.*
- Cambiar la imagen que muestra un **ImageSprite**.
- Utilizar el componente **Clock** para controlar varios eventos.
- Emplear **variables** para registrar números (como el nivel de energía de la mariquita).
- Crear y utilizar **procedimientos con parámetros**.
- Recurrir al bloque **and**.

Diseñar los componentes

Esta aplicación tiene un componente **Canvas** que se utilizará como tablero de juego. En él habrá **tres ImageSprite**: uno para la **mariquita**, otro para la **pulga** y un último para la **rana**. También emplearemos un componente **Sound** para *el croar* de este animal. Recurriremos a **OrientationSensor** para medir el grado de inclinación del teléfono y mover la mariquita en consecuencia.

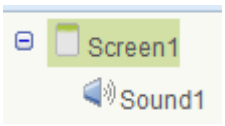
El componente **Clock** lo usaremos para modificar la dirección del movimiento de la pulga. Habrá un segundo **Canvas** que mostrará el **nivel de energía** de nuestra protagonista. Por último, un botón **Reiniciar** permitirá reiniciar el juego cuando la mariquita muera de hambre o sea devorada por la rana.

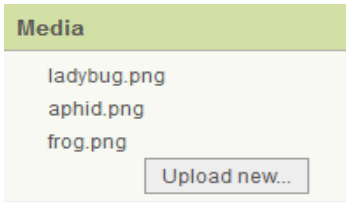
La siguiente tabla contiene todos los componentes que emplearemos:

Tipo de componente	Grupo de Palette	Cómo lo llamaremos	Finalidad
Canvas	**Drawing and Animation**	FieldCanvas	Tablero de juego.
ImageSprite	**Drawing and Animation**	Mariquita	Personaje controlado por el usuario.
OrientationSensor	**Sensors**	OrientationSensor1	Detecta la inclinación del teléfono y controla el movimiento de la mariquita.
Clock	**Sensors**	Clock1	Determina cuándo debe modificar la posición de ImageSprite.
ImageSprite	**Drawing and Animation**	Pulga	La presa de la mariquita.
ImageSprite	**Drawing and Animation**	Rana	El depredador de la mariquita.
Canvas	**Drawing and Animation**	EnergyCanvas	Muestra el nivel de energía de la mariquita.
Button	**User Interface**	RestartButton	Reinicia el juego.
Sound	**Media**	Sound1	Cuando la rana se come a la mariquita, hace que creo.

Empezamos

Conéctate a **AppInventor** e inicia un nuevo proyecto llamado **LadyBug** y cambia el texto de la barra de título por **LadyBug**. Abre **Blocks** y agrega todas las imágenes del capítulo suministradas y el componente **Sound** al panel desde **Media** (Multimedia).





Si las pruebas las vas a realizar en el teléfono, deberás desactivar la rotación automática de la pantalla. En la mayoría de los aparatos, esta opción estará en **Ajustes-> Pantalla**.

Definir el comportamiento de los componentes

Dotar de movimiento a la mariquita

Para mover a nuestra mariquita deberemos inclinar el teléfono. Vamos a crear a la mariquita y el sistema que controlará su movimiento.

- Añade un elemento **Canvas**, que será nuestro tablero de juego. Cambia el nombre a **FieldCanvas**. La propiedad **Width** ponla a **Fill parent** y **Height** a **300 píxels**.

- Coloca un elemento **ImageSprite** desde la sección **Animation**, dentro de **Canvas**. Cambia el nombre a **Mariquita** y modifica su aspecto incorporando la imagen de la mariquita viva dentro de la propiedad **Picture** (imagen).

- El valor de la propiedad **Interval** será **10 milisegundos**. Determinará la frecuencia con la que se moverá el elemento ImageSprite.

- La propiedad **Heading** indica la dirección hacia donde se moverá. Viene definida en grados. Por ejemplo, si tiene el valor 0 se dirigirá a la derecha; el valor 90 indicará que se desplazará hacia arriba; 180 a la izquierda, etc. De momento, no la modificamos.

- La propiedad **Speed** determina el número de píxels que deberá recorrer ImageSprite cada vez que se cumpla el tiempo determinado por Interval. La configuraremos despúes.

El encargado de controlar el movimiento de la mariquita será **OrientationSensor**, que

detectará el grado de inclinación del teléfono. Utilizaremos el componente **Clock** para comprobar la orientación del aparato *cada 10 ms*. También examinaremos si ha habido algún cambio en la dirección de la mariquita (*Heading*). Configuraremos estos comportamientos desde **Blocks**:

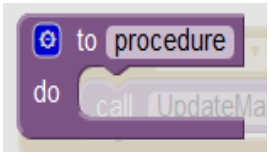
- Agrega un componente **OrientationSensor**, desde **Sensors**,que aparecerá en la sección **Non-Visible-components**.

- Añade un elemento **Clock**, desde **Sensors**, que también estará en la sección **Non-Visible-components** y, configura su propiedad **TimerInterval** a 10 ms.

Ahora asignaremos **comportamientos** a los componentes que acabamos de agregar.

Vamos a **Blocks** para crear los **procedimientos UpdateMariquita** y **Clock.Timer** como muestra la figura de abajo:

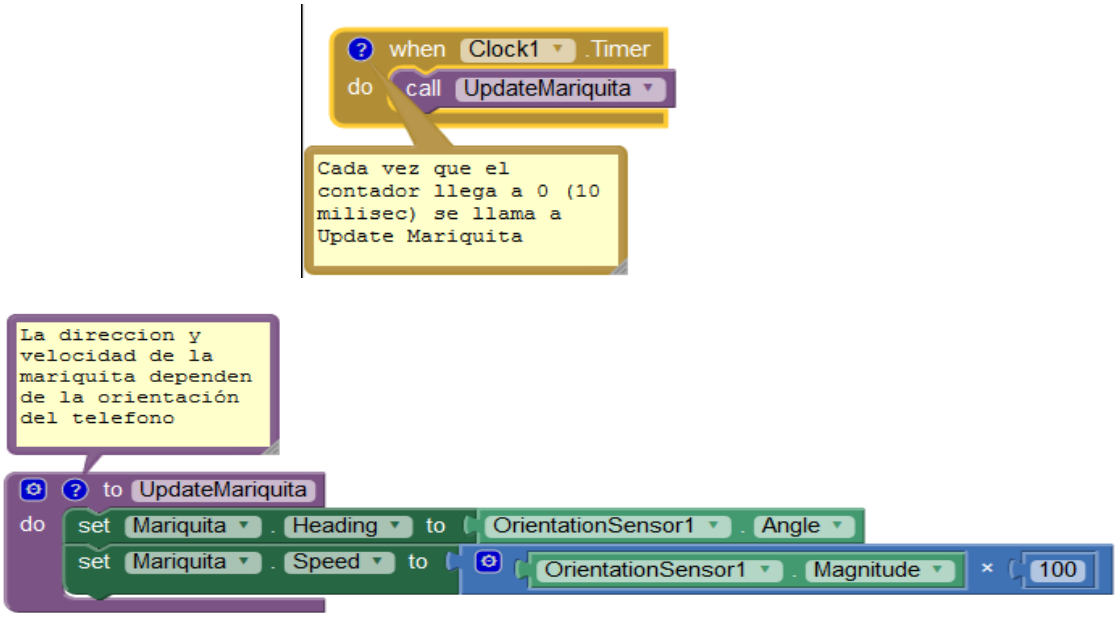
Recuerda que para escribir un procedimiento deberías ir a **Procedures** y arrastrar el bloque de **Procedure** al area de trabajo, luego cámbiale el nombre.



En vez de arrastrar los bloques de los cajones, prueba a escribir directamente sobre el área de trabajo sus nombres (como por ejemplo, **Clock1.Timer**).

Para añadir comentarios haz clic con el botón derecho del ratón y selecciona la opción **Add Comment** (agregar comentario).

Termina los bloques tal que:



El procedimiento **UpdateMariquita** usará dos de las propiedades de **OrientationSensor**:

- **Angle**: Indica la dirección en la que se inclina el teléfono (medida en grados).

- **Magnitude**: Indica la cantidad de inclinación que se aplica al teléfono. Su valor va desde 0 (cuando no se asigna ninguna) hasta 1 (inclinación máxima).

Al multiplicar **Magnitude** por 100, estaremos diciéndole a la mariquita que debe moverse entre 0 y 100 píxels en la dirección que especifique la propiedad **Heading**. Esta operación tendrá lugar siempre que se agote el tiempo definido por **TimerInterval**, que será de 10 milisegundos.

Guarda primero el proyecto con el nombre **Ladybug**, se generará el fichero **"LadyBug.apk"** que es el que debes copiar a tu teléfono. Si ves que el movimiento de la mariquita es lento, aumenta su velocidad. Si se desplaza demasiado rápido, redúcela.

PRUEBA!!!!

Mostrar el nivel de energía

Utilizaremos una **barra de color rojo** para mostrar el nivel de energía de la mariquita. Tendrá un píxel de alto y su ancho indicará el nivel de energía, que irá desde 200 (barra de energía

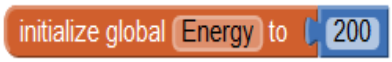
llena) hasta cero (muere la mariquita).

Desde **Component Designer**, generaremos un nuevo elemento **Canvas**, que colocaremos debajo de **FieldCanvas** y bautizaremos como **EnergyCanvas**. A su propiedad **Width** le asignaremos el valor **Fill parent** y en la propiedad **Height** le asignaremos el valor de 1 píxel.

Vamos a crear una variable llamada **energy** desde **Blocks Editor**. Aquí crearemos una variable para la energía que tendrá un **valor inicial de 200** y que guardará información sobre el nivel de energía de la Mariquita:

- Abre **Blocks** y arrastra hasta el área de trabajo un bloque para definir una variable. Renombra el texto **variable** por **energy**.

- Desde Math arrastra el bloque **number** hasta el área de trabajo, colócalo en la ranura de la parte derecha del bloque de definición de la variable **energy**. Pon el valor **200** como inicial para la variable.



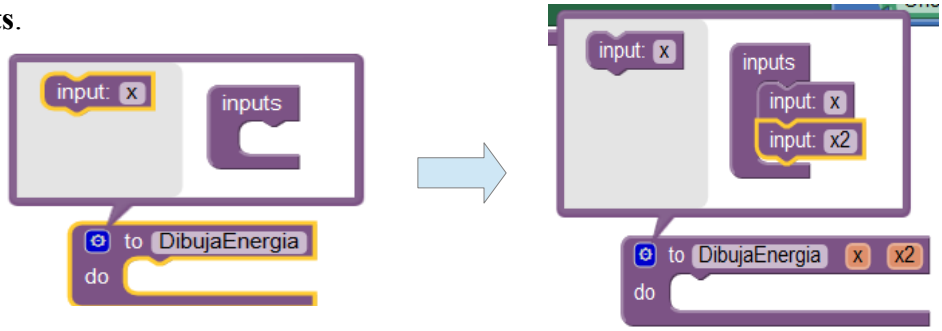
Dibujar la barra de energía

Mostraremos el nivel de energía a través de una barra roja. Para generar el procedimiento **DibujaEnergia** realizaremos lo siguiente:

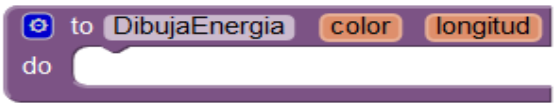
1. Arrastra un bloque **to procedure** al área de trabajo.

2. Cambia el nombre y llámalo **DibujaEnergia**.

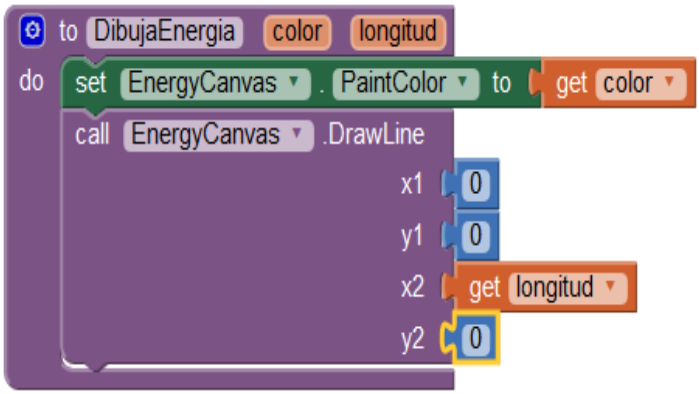
3. Este procedimiento va a tener dos parametros: el *color* y la *longitud*. Para crear parámetros haz clic sobre la rueda del procedimiento y arratra 2 **input:x** al lado derecho, dentro de **Inputs**.



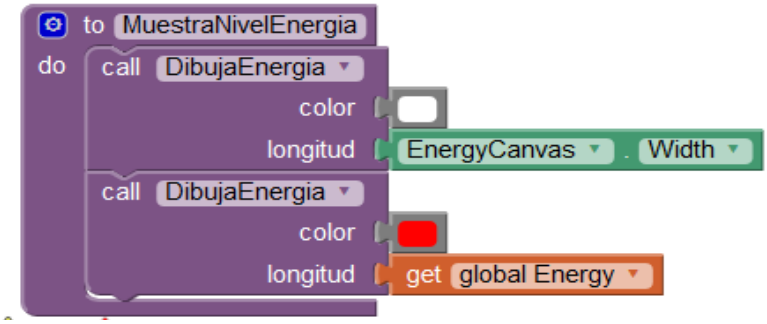
Fíjate como aparecen dos parametros **X** y **X2**, cambia ahora sus nombres por **color** y **longitud.**



4. Completa el resto del procedimiento para que quede como en la figura siguiente:



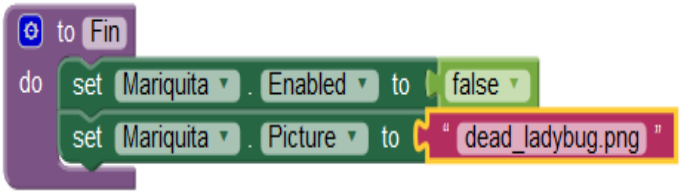
Ahora crearemos un segundo procedimiento llamado **MuestraNivelEnergia** y que se encargará de llamar dos veces a **DibujaEnergia**. Con la primera invocación borrará la línea que mostraba el último estado de energía conocido (dibujará encima de ella otra blanca) y con la segunda enseñará en la pantalla la línea correspondiente al nuevo estado de energía, como en la siguiente figura:



Inanición

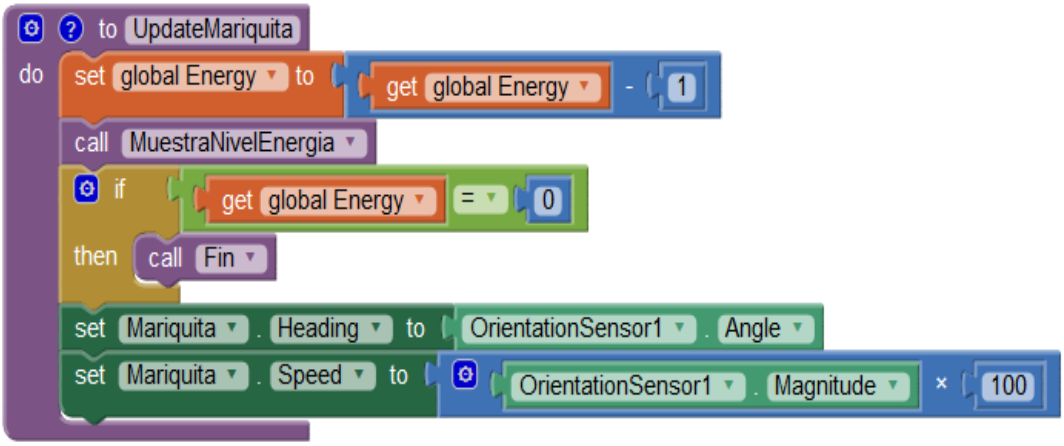
A diferencia de lo que ocurría con las aplicaciones de los capítulos anteriores, este juego puede terminarse. Cuando la mariquita no consiga comer las suficientes pulgas o sea devorada por la rana, el juego finalizará. En cualquiera de los casos, detendremos el movimiento de la protagonista (asignaremos el valor **false** a **Mariquita.Enabled**) y cambiaremos su foto por una en la que aparece muerta. Esto último lo conseguiremos escribiendo el nombre de la nueva imagen

dentro de **Mariquita.Picture**. Vamos a crear el procedimiento **Fin** como en la figura:



A continuación, vamos a añadir tres bloques de código al procedimiento ya creado **UpdateMariquita**. Como recordarás, **Clock.Timer** lo llamaba cada 10 milisegundos. La finalidad de estos tres nuevos códigos es la siguiente:

- Reducir el nivel de energía.
- Mostrar el nuevo nivel en la pantalla.
- Finalizar el juego si el nivel de energía es 0.



Guarda primero el proyecto con el nombre **Ladybug2**, se generará el fichero **"LadyBug2.apk "**que es el que debes copiar a tu teléfono.

Verifica si el nivel de energía varía con el paso del tiempo. Agota el nivel de energía para ver si la mariquita se muere.

PRUEBA!!!!

Añadir una pulga

El siguiente paso será añadir una pulga. El insecto deberá revolotear por **FieldCanvas**. Si la mariquita choca con ella, es decir, si se la come, su nivel de energía deberá aumentar y la pulga desaparecer de la pantalla. Al poco tiempo, la aplicación volverá a colocar otra en el tablero de juego. Aunque para nosotros ésta sea el mismo componente **ImageSprite**, para el usuario será un insecto diferente.

El primer paso para añadir la pulga al juego es volver a **Component Designer** para crear otro elemento **ImageSprite**. Asegúrate de no colocarlo encima de la mariquita. Cambia su nombre predeterminado por **Pulga** y configura sus propiedades:

1. Abre la imagen de la pulga "**Aphid.png"** en la propiedad **Picture**.

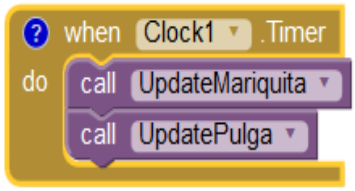
2. Asigna el valor 10 a **Interval** para que se mueva cada 10 milisegundos como la mariquita.

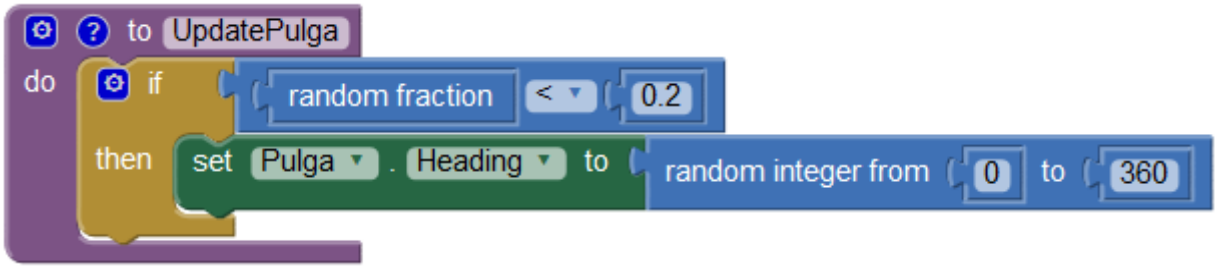
3. Asigna el valor 2 a **Speed** para que no se mueva demasiado rápido y permita que la mariquita pueda cogerla. No te preocupes por las propiedades *X,Y o Heading* porque las configuraremos desde **Blocks** más tarde.

Basándonos en las pruebas que hemos realizado al juego, hemos visto que es conveniente que la **pulga cambie de dirección** aproximadamente una vez cada 50 milisegundos (cada 5 ciclos completos de Clock1). Una forma de conseguirlo sería creando un segundo reloj que tuviese un intervalo (**TimerInterval**) de **50 milisegundos**. Pero hemos preferido utilizar una técnica distinta para que aprendas a usar el bloque **random fraction** que, cuando se invoca, devuelve un número aleatorio *mayor o igual que 0 y menor que 1*. Crea el procedimiento **UpdatePulga** y luego haz que **Clock1.Timer** lo invoque como en la figura que tienes más abajo.

Cada vez que se agota el tiempo (100 veces por segundo), se llama a **UpdateMariquita** (igual que ocurría antes) y a **UpdatePulga**. Lo primero que hace **UpdatePulga** es generar una fracción aleatoria comprendida entre 0 y 1(por ejemplo, 0,15). si este número es menor que 0,20

(algo que pasará el 20 por cien de las veces), la dirección de la pulga cambiará a una cantidad aleatoria de grados comprendida entre 0 y 360. Si la cifra es mayor de 0,20 (algo que ocurrirá el 80 por cien de las veces), la pulga continuará su trayectoria.





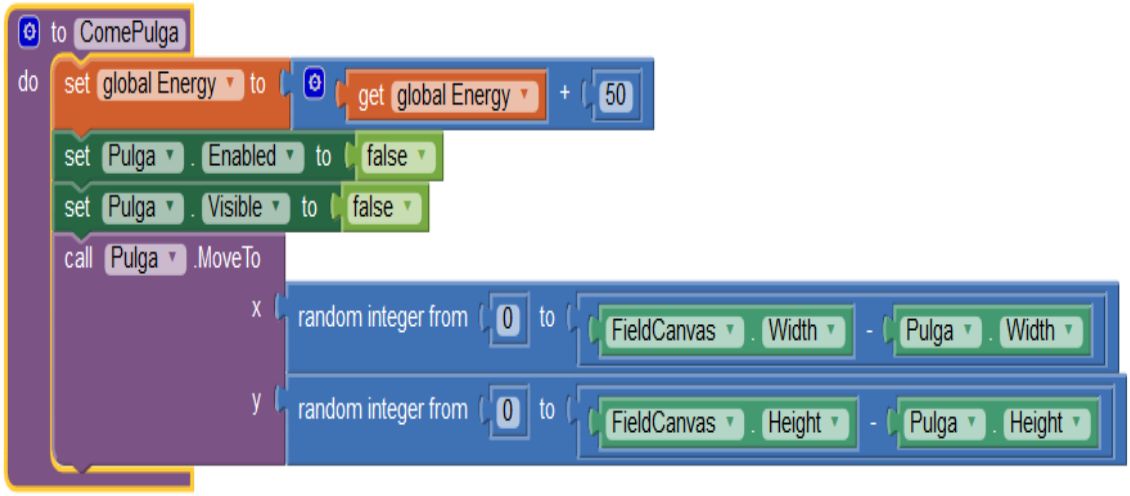
El siguiente paso será conseguir que la mariquita se coma a la pulga cuando impacte con ella (procedimiento nuevo llamado **ComePulga**). Afortunadamente, **AppInventor** cuenta con bloques para detectar las colisiones entre componentes *ImageSprite*. ¿Qué ocurrirá cuando la mariquita choque con la pulga?

- Aumentará el nivel de energía 50 puntos.
- La pulga desaparecerá de pantalla (deberemos asignar el valor **false** a su propiedad **Visible**).
- La pulga dejará de moverse (asignaremos **false** a su propiedad **Enabled**).
- La pulga se moverá hasta una ubicación aleatoria de la pantalla. Aquí, seguiremos el mismo modelo de código que vimos con **DaleFuerte**
- Si consideras oportuno añade un sonido cuando se coma.

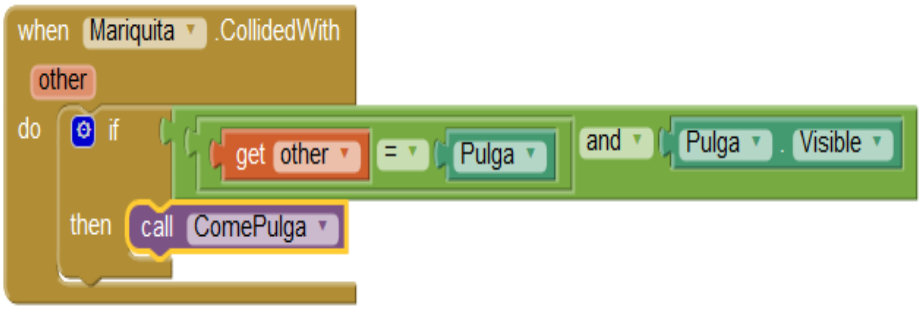
Cada vez que se llame al procedimiento **ComePulga**, se suma **50** a la variable **energy**. Luego, se asigna el valor **false** a las propiedades **Visible** y **Enabled** para que la pulga desaparezca de la pantalla y detenga su movimiento. Finalmente, se generan unas coordenadas **X** e **Y** aleatorias y

se le entregan al procedimiento **Pulga.MoveTo**. De esta forma, conseguimos que el insecto reaparezca en una nueva ubicación, evitando que la Mariquita se lo vuelva a comer de inmediato.

El código que debes generar (sin sonido) es el siguiente:



Para detectar la colisión-choque entre la mariquita y la pulga hemos de añadir el siguiente código:



El anterior código tiene la siguiente explicación:

Cuando la mariquita choca con otro elemento **ImageSprite**, la aplicación llama a **Mariquita.CollideWith**, utilizando el parámetro **other** para hacer referencia al objeto con el que topó el insecto. De momento, el único elemento con el que puede impactar la mariquita es la pulga pero, en breve, añadiremos una rana. Antes de llamar a **ComePulga**, debemos comprobar que la mariquita ha chocado, efectivamente, con la pulga. También debemos examinar que la pulga esté visible en el tablero, ya que si no lo está, la mariquita se la comería antes de que volviese a aparecer

y volvería a subir su nivel de energía.

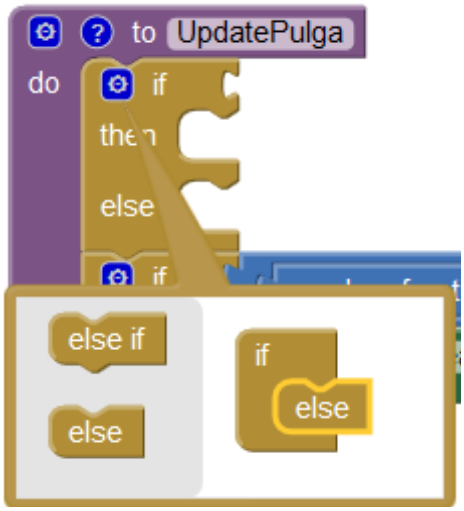
Guarda primero el proyecto con el nombre **LadyBug3**, se generará el fichero **"LadyBug3.apk"** que es el que debes copiar a tu teléfono.

PRUEBA!!!!

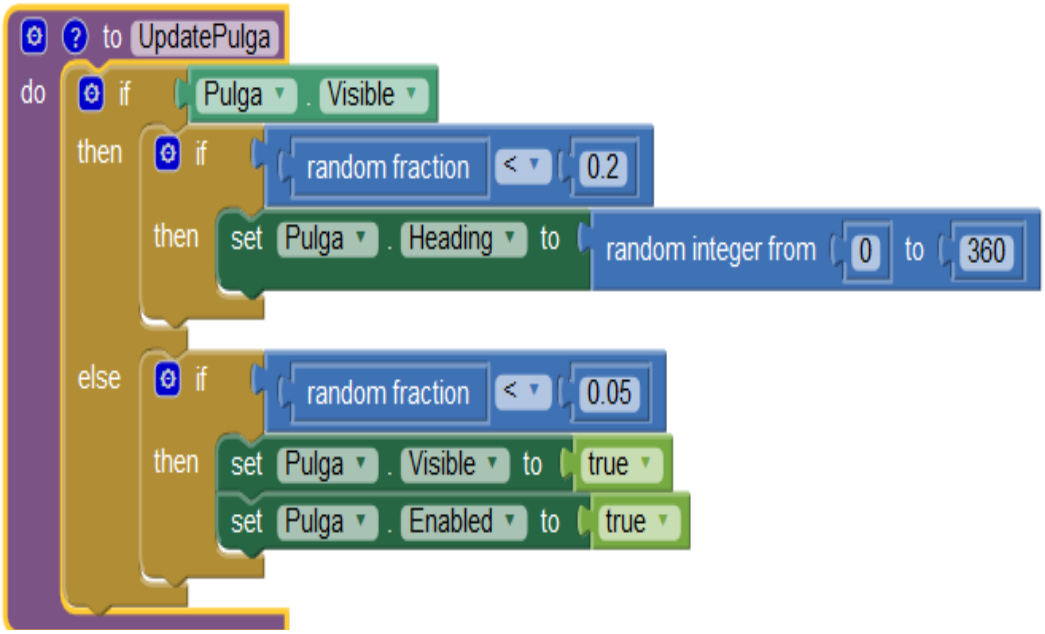
Para que la pulga pueda aparecer en la pantalla deberemos modificar **UpdatePulga**. Tan sólo **se alterará la dirección de la pulga cuando sea visible**. No merece la pena cambiarlo cuando es invisible.

Cuando no aparezca en la pantalla, es decir, cuando la mariquita acabe de comérsela, habrá un 5 por cien de posibilidades (1 entre 20) de que se vuelva a activar de nuevo, es decir, que vuelva a aparecer como presa de la mariquita.

En este caso vamos a emplear una estructura IF THEN ELSE. Este bloque se obtiene insertando un bloque IF THEN y luego desde la rueda se añade un bloque ELSE, arrastrnadolo dentro de la ventana del IF, algo similar a lo que hicimoa para añadir los parametros de un procedure.



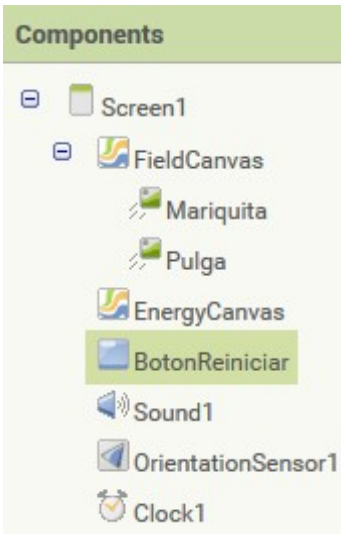
El código modificado de **UpdatePulga** es el siguiente:



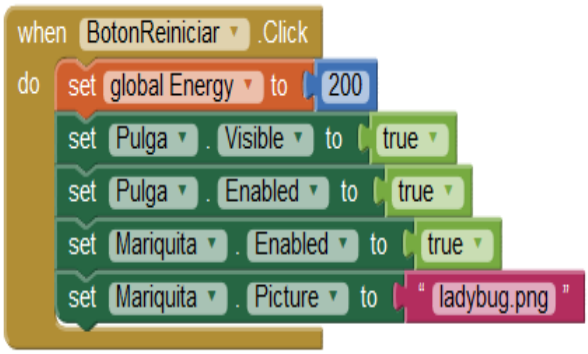
Guarda primero el proyecto con el nombre **LadyBug4**, se generará el fichero **"LadyBug4.apk"** que es el que debes copiar a tu teléfono.

PRUEBA!!!!

Ahora agregaremos un **botón de Reinicio de partida**. Desde **Component Designer**, agregaremos un componente **Button** al área de trabajo y lo colocaremos debajo de **EnergyCanvas**. Lo llamaremos **BotonReiniciar** y en la propiedad **Text** escribiremos **Reiniciar**. Después, en **Blocks,** crearemos el código asociado al pulsar el botón:



- Devolver el nivel de energía a 200.
- Volver a activar la pulga y mostrarla de nuevo en pantalla.
- Reactivar a la mariquita y cambiar su imagen para que aparezca viva.



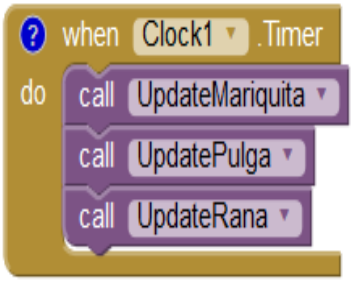
Guarda primero el proyecto con el nombre **LadyBug5**, se generará el fichero **"LadyBug5.apk"** que es el que debes copiar a tu teléfono.

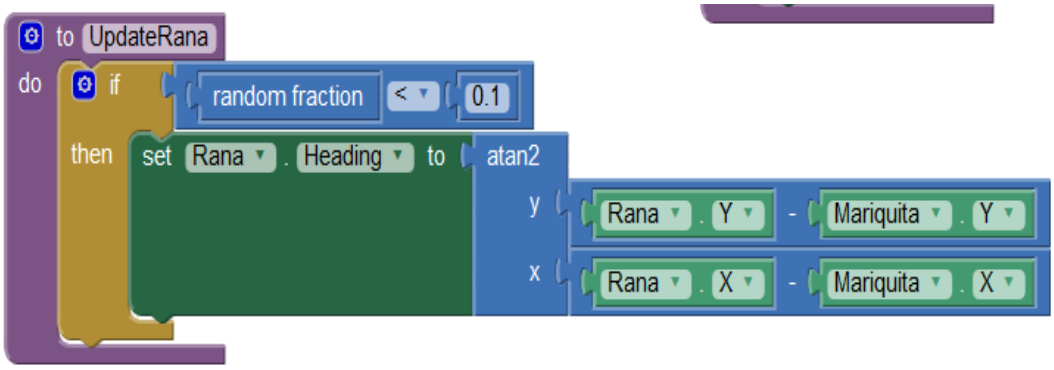
PRUEBA!!!!

Ahora añadiremos a un depredador de nuestra mariquita, la **rana**, que se moverá hacia ella. Si la atrapa el juego finalizará.

Lo primero es colocar la rana en el tablero de juego **FieldCanvas** desde **Component Designer**, arrastrando un **ImageSprite**, al que llamaremos **Rana**. Incorporaremos su imagen en la propiedad **Picture**, a su movimiento le asignaremos un intervalo de **10 milisegundos** y a su **velocidad** le daremos el valor **1**, ya que deberá moverse más lento que el resto de criaturas del juego.

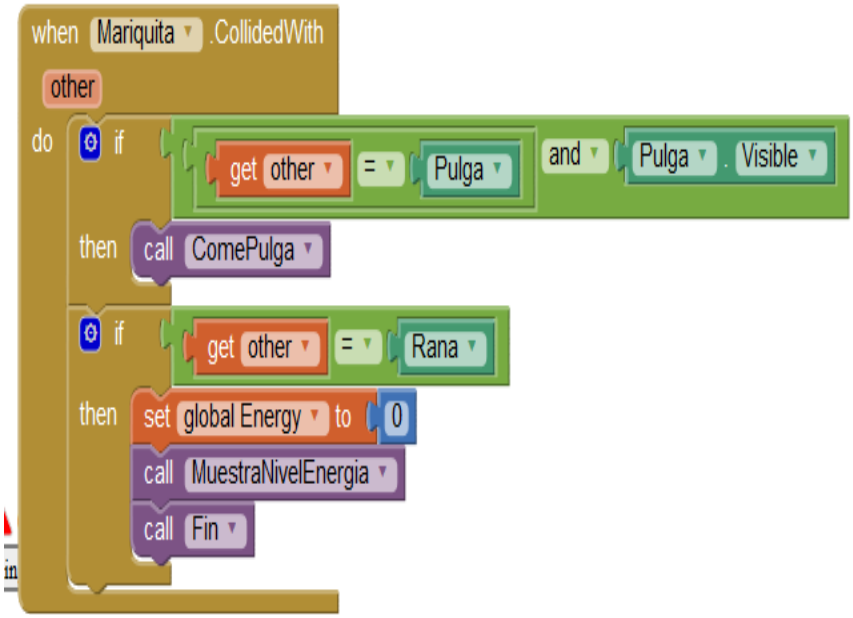
Ahora hemos de crear el procedimiento **UpdateRana** y modificar el ya creado C**lock1.Timer**.





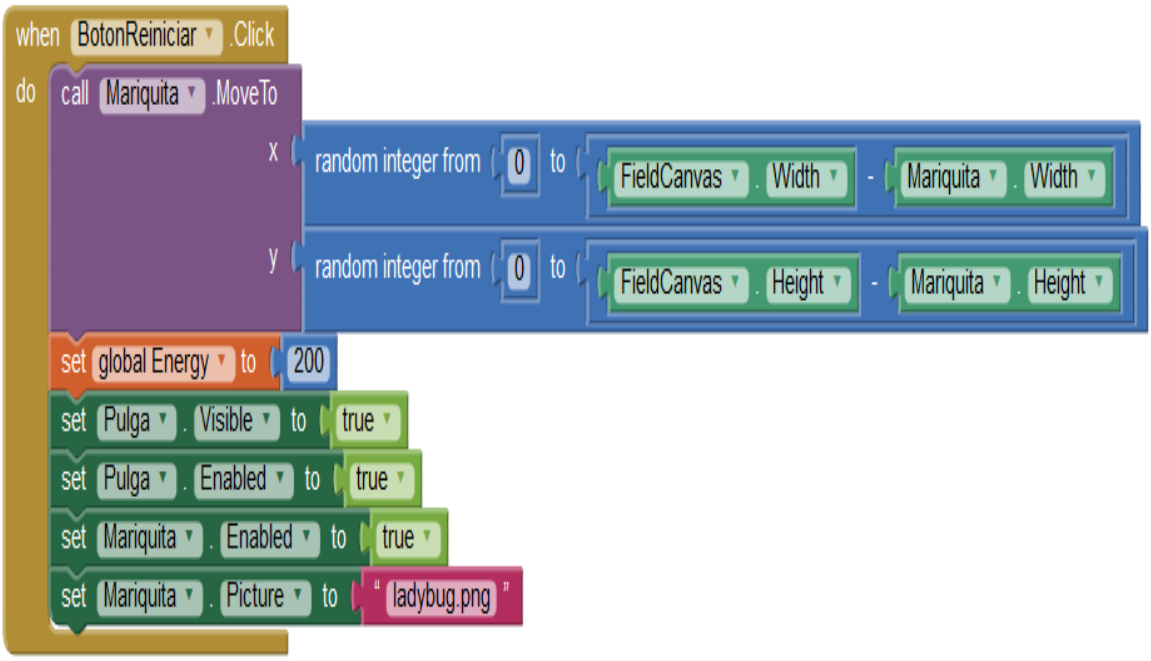
Atan2, es una función matemática de **AppInventor,** que devuelve *el ángulo* correspondiente a los valores X e Y suministrados.

Ahora hemos de modificar el código para controlar el choque de la mariquita con cualquier otro elemento del tablero. Ahora, si la mariquita se topa con la rana, la barra y el **nivel de energía** deberían de descender hasta **0** y el juego terminaría. El valor de la variable **energy** descenderá hasta 0. Se llamará a la función **MuestraNivelEnergia** para que borre la línea de energía en la pantalla y dibuje una nueva que estaría vacía. Además, la aplicación llamará al procedimiento **Fin** para que detenga el movimiento de la mariquita y cambie su imagen por un insecto muerto:



Un último detalle....

Hemos de tener en cuenta **el retorno de la mariquita** en el botón **Reiniciar**. Si no movemos la mariquita cuando reiniciamos el juego, volvería a aparecer en la boca de la rana. Aunque vamos a utilizar para ello la función **random integer** ya conocida, la probabilidad de que aparezca la mariquita encima de la rana es tan baja, que no merece la pena complicar más el código:



Guarda primero el proyecto con el nombre **LadyBug6**, se generará el fichero **"LadyBug6.apk"** que es el que debes copiar a tu teléfono.

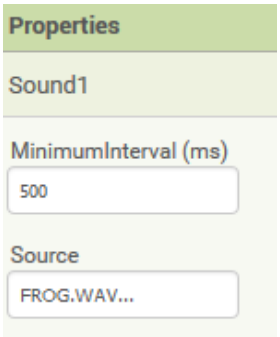
PRUEBA!!!!

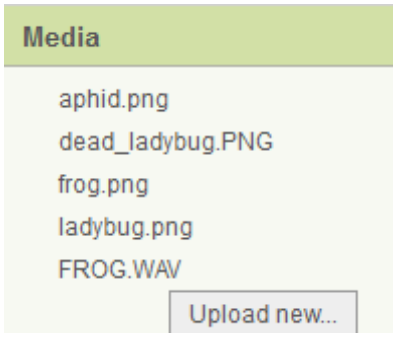
Vamos a añadir efectos sonoros. Para ello utilizaremos el componente **Sound1** que añadimos al principio de la práctica (compruébalo primero). Ahora, desde **Blocks Editor** haz lo siguiente:

- Haz que el teléfono vibre cuando la mariquita se coma a la pulga. Para ello, deberás agregar el bloque **Sound1.Vibrate** a **ComePulga** y asignarle un argumento de **1000 milisegundos**. Ponla como primera instrucción de este bloque.

- Haz que la rana croe cuando se coma a la mariquita. Subelo desde Media(FROG.WAV).

Modifica el **sound1** desde sus propiedades en Designer. Añade una llamada **Sound1.Play** en **Mariquita.CollideWith** justa antes de que se invoque a la función **Fin**.





Guarda primero el proyecto con el nombre **LadyBug7**, se generará el fichero **"LadyBug7.apk"** que es el que debes copiar a tu teléfono.

PRUEBA!!!!

Variaciones

Algunas ideas para mejorar el juego:

- Ahora, la **rana y la pulga** siguen moviéndose incluso después de terminar el juego. Para evitarlo, deberías asignar el valor **false** a las propiedades **Enabled** que se encuentran en la función **Fin** y el valor true en **Reiniciar.Click**.

- Muestra un **marcador** que indique la **cantidad de vida** que le queda a la mariquita. Para ello, puedes crear una etiqueta que incremente su valor con **Clock1.Timer**.

- Aumenta la altura de la barra de energía para que sea más llamativa. Recuerda que puedes hacerlo incrementando el valor de la propiedad **Height** de **EnergyCanvas** hasta **2** y dibujando dos líneas, una sobre otra, en **DibujarEnergia**.

- Añade una imagen de fondo para mejorar el entorno y nuevos efectos de sonido, como los de la naturaleza o algún aviso de cuando el nivel de energía de la mariquita sea demasiado bajo.

- Haz que el **juego** sea cada vez **más difícil**. Para ello, puedes incrementar la **velocidad** de la **rana** o disminuir el valor de la propiedad **Interval**.

- Técnicamente, la mariquita debería desaparecer cuando sea devorada por la rana. Cambia el juego para que esto ocurra cuando sea comida pero permanezca en la pantalla cuando muera de hambre.

- Sustituye las imágenes de la mariquita, la pulga y la rana por otras que sean de tu gusto, como por ejemplo, un hobbit, un orco y un mago diabólico.