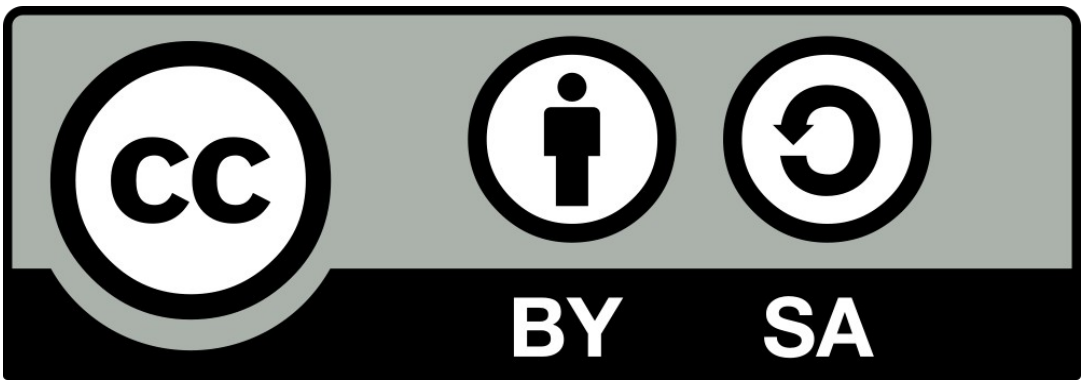
AppInventor

¿Qué pasa si hacemos programas y videojuegos para nuestro móvil o tablet?

:D



Serafín Vélez Barrera – Oficina de Software Libre

Introducción

AppInventor es un entorno en el que disponemos un programa de diseño y otro programa de programación unidos los dos entre sí. Así entonces podemos diseñar una interfaz que es lo que vamos a ver y darle funcionalidad a los programas mediante una serie de piezas.

Instalación

Requisitos

- Ordenador y sistema operativo:
    - GNU/Linux: Ubuntu 8 o superior, Debian 5 o superior. Si tienes otro sistema operativo habla con el profe para que te de directrices.
- Navegador
    - Mozilla Firefox 3.6 o superior. Nota. Si estás usando Firefox con la extensión "NoScript", tendrás que deshabilitarla. Puedes ver más información [aquí] (http://explore.appinventor.mit.edu/content/troubleshooting).
    - Google Chrome 4.0 o superior.
    - Google Chromium.

Pasos

Para instalar AppInventor en nuestro ordenador con Linux necesitamos hacer lo siguiente:

1. **Instalar Java**

Lo primero es que tenemos que tener instalado Java ya que sin esto no podemos continuar, para ello nos dirigimos a [este enlace] (http://www.java.com/en/download/testjava.jsp) para comprobar que lo tenemos instalado. Si está instalado se verá un mensaje que lo diga explícitamente.

Luego tenemos que comprobar que el navegador funciona bien con Java para así poder usar JavaWebStart que es una app de Java para iniciar aplicaciones Java desde el navegador. Esto lo comprobaremos haciendo click [aquí] (http://beta.appinventor.mit.edu/learn/setup/misc/JWSTest/AppInvJWSTest.html).

Si haciendo esto tienes problemas habla con tu profe para ver por qué puede ser o escribe un email a serafin.velez@gmail.com.

2. **Instalar AppInventor**

Para instalar AppInventor necesitamos tener los privilegios de administrador, es decir que necesitamos saber la contraseña del equipo para poder instalar programas. Así que lo que tenemos que hacer es:

- Descargar el paquete de instalación de AppInventor, hazlo desde [aquí] (http://dl.google.com/dl/appinventor/installers/linux/appinventor-setup_1.1_all.deb).

- Una vez descargado lo podemos instalar de varias formas:
    - Usando la terminal. Escribiremos ``` sudo dpkg --install appinventor-setup_1.1_all.deb```
    - De forma gráfica:
        - Instalarlo con el Centro de Software. Esta es la opción más recomendada.
        - O bien instalarlo con GDebi, que deberás tenerlo instalado previamente en tu equipo.
- Cuando hagamos esto en algún momento el ordenador nos preguntará por una contraseña, esa es la del super administrador del ordenador (si sólo hay un único usuario seguramente esa contraseña será la misma que la de este usuario).
- Una vez instalado, podemos comprobar que así es dirigiéndonos a la carpeta en ```/usr/google/appinventor/commands-for-Appinventor```

Empecemos

Cómo se programa con AppInventor

Vamos a aprender cómo se programa con AppInventor haciendo nuestro primer programa.

App Inventor consta de 2 partes que se comunican entre sí:

1. Diseñador (funciona desde tu navegador)
2. Editor de bloques (funciona como una aplicación de Java)

Necesitarás tener estas dos ventanas abiertas para diseñar tus apps.

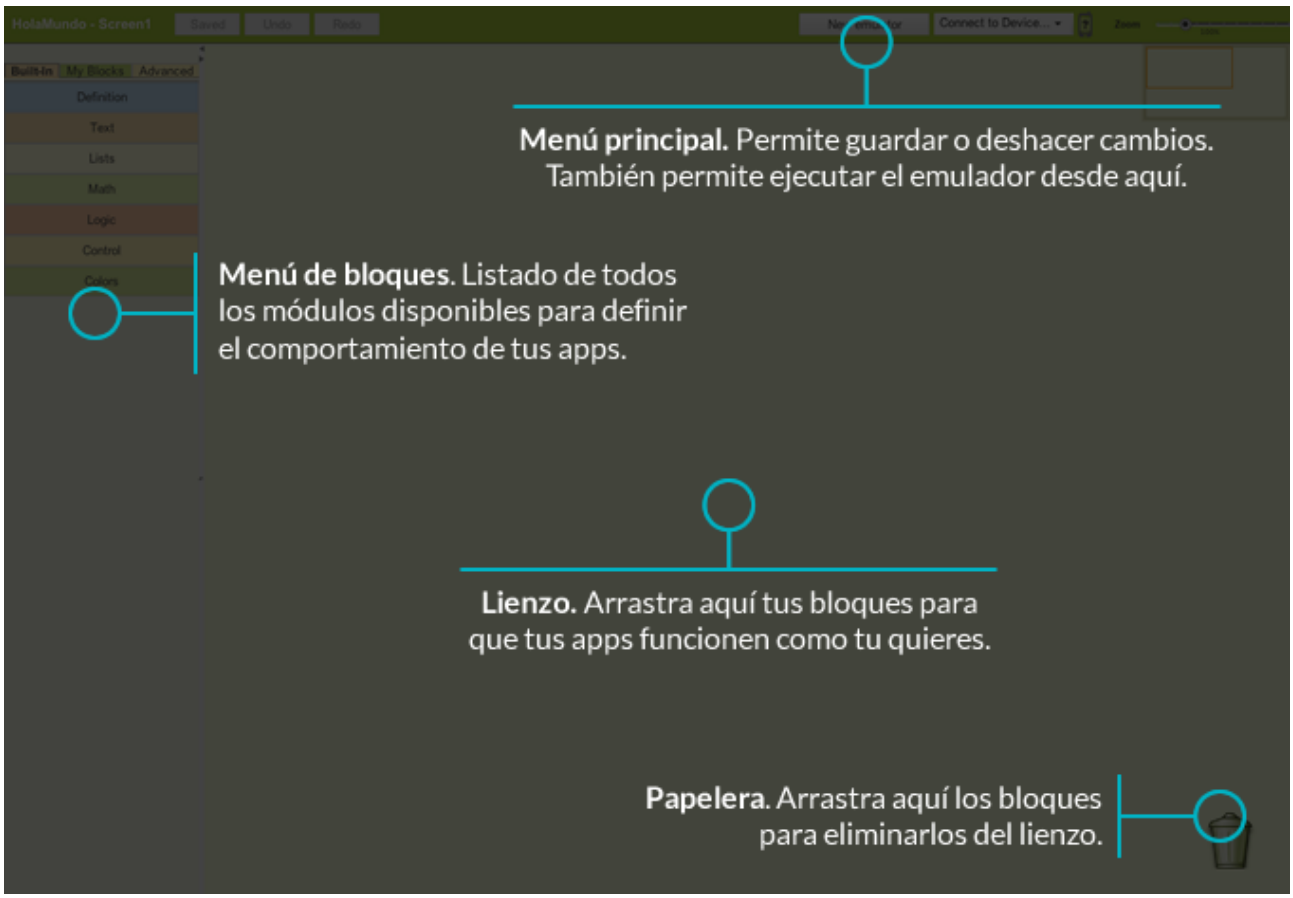
DISEÑADOR

La página de "Diseñador" es el lugar donde seleccionas los componentes de tu app y diseñas la interfaz. Está compuesto por las siguientes áreas:

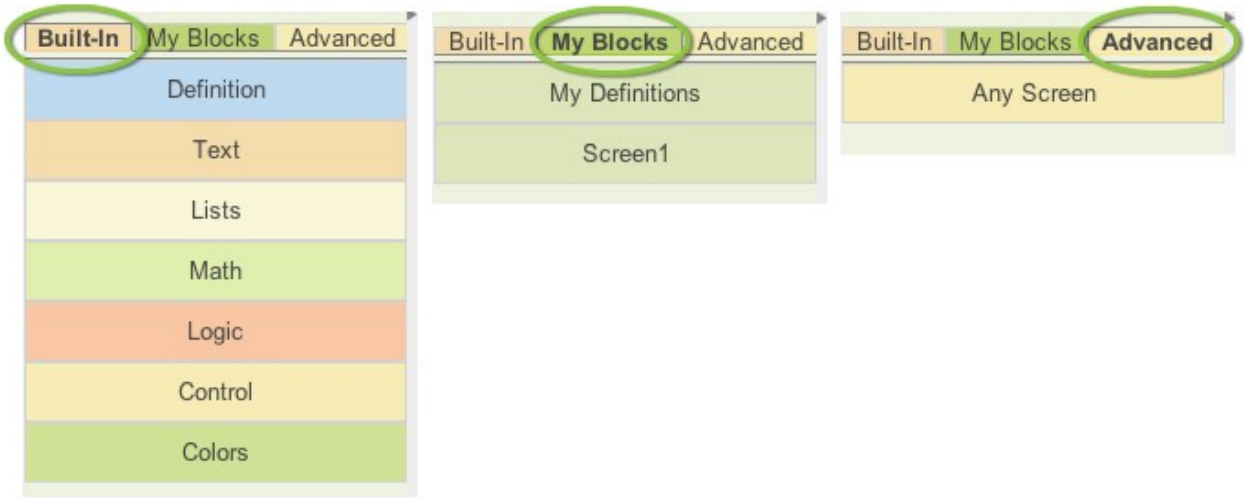
EDITOR DE BLOQUES

El editor de bloques es el lugar donde defines un comportamiento de la app. Está

compuesto por las siguientes áreas:



Los bloques se agrupan en 3 grandes grupos:



1. Built-in. Contiene los elementos básicos para dar comportamiento a tu aplicación.
2. My blocks. Contiene los bloques correspondientes a los componentes que has incluido en la página de "Diseñador".
3. Advanced. Contiene bloques con funcionalidades más avanzadas.

Tres pasos y tres ventanas

Para programar nuestras propias aplicaciones con Appinventor siempre vamos a seguir tres pasos y en cada paso vamos a usar una ventana distinta. Estos son los pasos:

1. Primero diseñamos nuestra aplicación utilizando la ventana del Diseñador de AppInventor (designer). Aquí incluimos los objetos (componentes) que formarán parte de nuestra aplicación.
2. Después programamos la aplicación para que haga lo que queremos usando el Editor de bloques (block editor). Desde aquí definimos cómo la aplicación responde a los usuarios y a los eventos externos (el comportamiento de la aplicación).
3. Por último probamos en la pantalla del emulador, móvil o tablet el resultado para corregir los errores.

Para poder trabajar correctamente debemos de tener el navegador con la web de AppInventor (diseñador) abierta, el editor de bloques también abierto y este a su vez conectado a nuestro dispositivo (móvil o tablet) o al emulador que nos permite simular cómo se vería nuestro programa aunque no tengamos ningún dispositivo conectado.

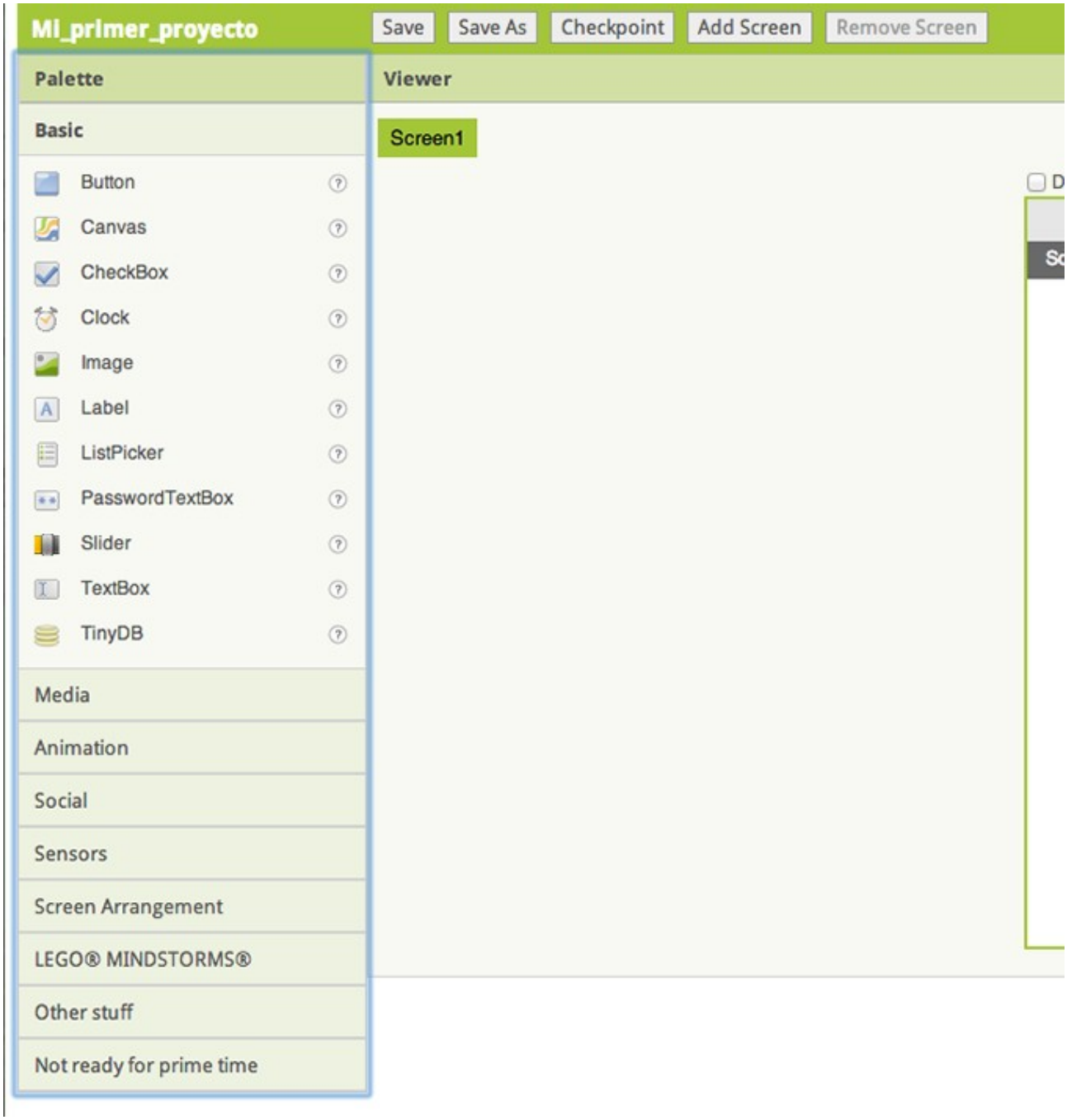
Dentro del Diseñador tenemos un enlace llamado Mis proyectos (My projects) donde podemos guardar y crear proyectos nuevos.

También podemos subir o descargar nuevos proyectos desde nuestro ordenador. Vamos a empezar directamente creando nuestro primer proyecto. Pulsa en New (nuevo) y dale un nombre a tu proyecto, p. ej.: "hola_mundo".

Puedes usar números y letras pero no puedes usar espacios. Cuando termines entrarás automáticamente en la pantalla del Diseñador.

El diseñador de App Inventor

El siguiente paso es diseñar la pantalla principal de nuestro programa (interface de usuario) y elegir los componentes que necesitamos para que funcione.

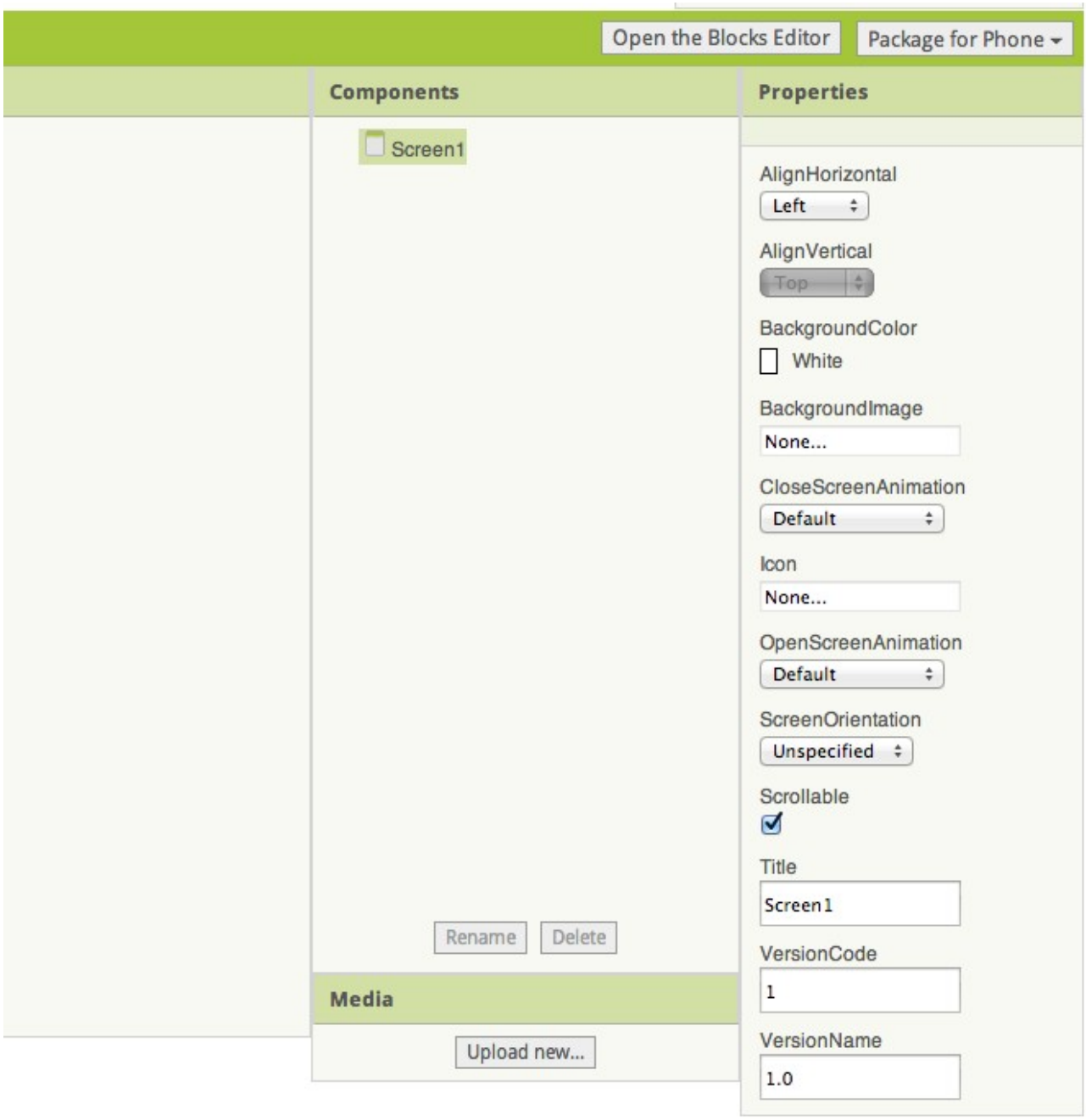


Los componentes son los elementos básicos que utilizamos para hacer programas con AppInventor. Son como los ingredientes de una receta. Cada componente hace algo distinto: mostrar un texto en la pantalla, un botón, una imagen, etc. Otros componentes controlan el sensor de movimientos del teléfono o el GPS. Puedes ver una lista de todos los componentes que puedes usar en tus aplicaciones.

Podemos usar varias veces el mismo componente en nuestro programas, por ejemplo, el componente botón (Button) podemos usarlo tantas veces como botones queramos poner en nuestro programa. Para poder distinguirlos fácilmente usaremos

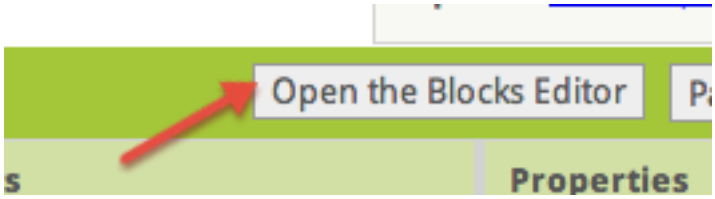
nombres distintos. P. ej.: boton1, boton2, boton3, etc. De esta forma podemos dar instrucciones concretas a cada componente sin confundirnos. Ahora veremos cómo.

Algunos componentes tiene también propiedades que te permiten cambiar cómo se ve el componente en la pantalla o cómo funciona dentro de tu programa. Por ejemplo, el tamaño, el color, etc. Las propiedades de los componentes aparecen aquí:



Hola Mundo

1. Antes de empezar a añadir componentes a nuestro programa abre el Editor de bloques pulsando aquí:



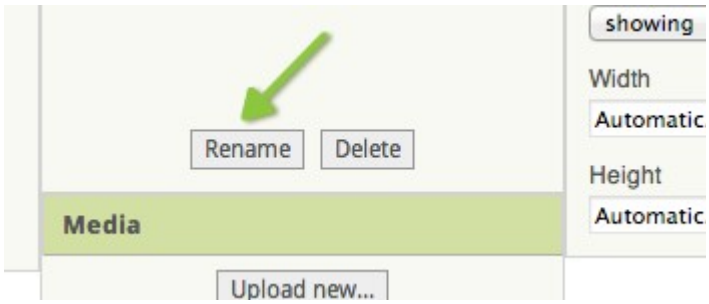
2. Una vez que esté abierto conéctate con tu dispositivo o emulador desde aquí:

    1. Si todo funciona correctamente ya podemos empezar a añadir componentes.

3. Desde la paleta Basic coge el componente Label (etiqueta) y arrástralo al visor (Viewer).

    1. En este caso el componente Label tiene una propiedad llamada Text que contiene el texto que queremos que se muestre en la pantalla.

4. Cambia el texto y escribe "Hola mundo".
5. Cambia también el nombre del componente aquí y escribe, p. ej.: "mi_etiqueta":

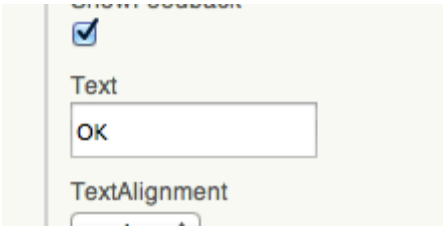


6. Para cambiar el color del texto tienes otra propiedad llamada TextColor. Prueba a cambiar el color.

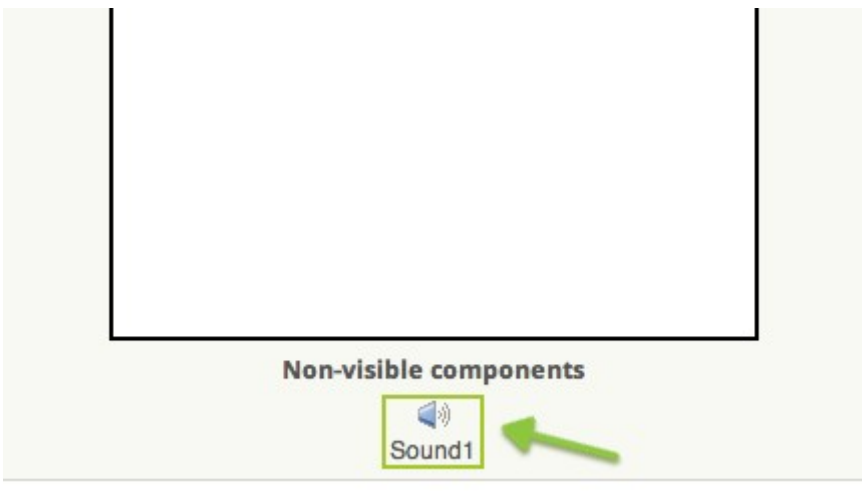
    1. Si todo ha funcionado correctamente deberías de ver en tu emulador o en tu dispositivo el texto que has puesto en tu programa.
    2. Recuerda que cada vez que hagas un cambio este aparecerá automáticamente en tu dispositivo. ¿Funciona?

7. Desde la paleta Basic coge el componente button (botón) y arrástralo al visor (Viewer).
8. Cambia el nombre del botón aquí y escribe, p. ej "mi_boton":

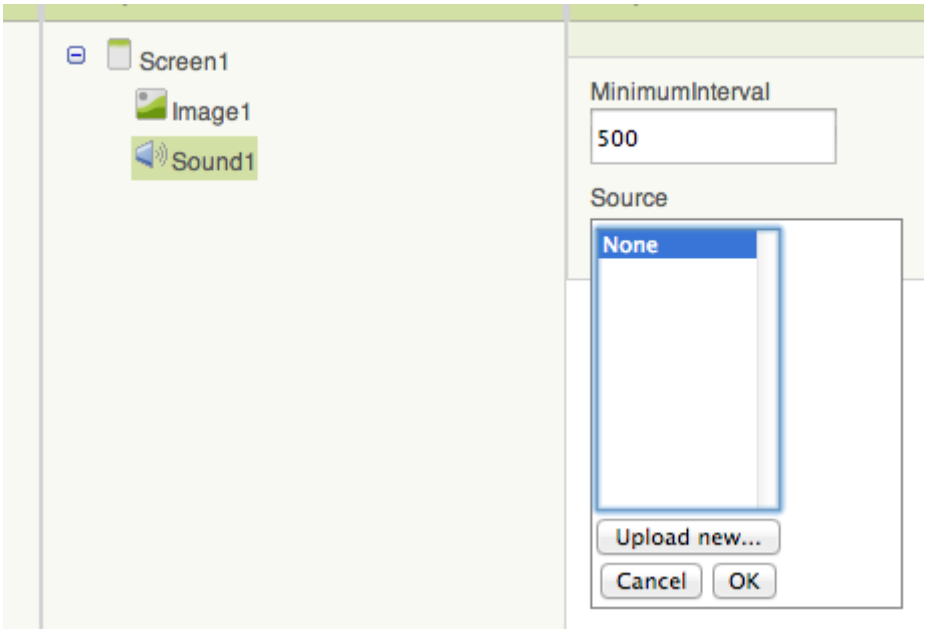
9. Cambia el texto del botón (text en sus propiedades) aquí y escribe, por ejemplo "Ok":



10. Para añadir una imagen a nuestro programa utilizamos el componente Image (imagen) dentro de la paleta basic. Arrastra este componente y colócalo en primer lugar, por encima de los otros dos componentes que ya tenemos.
11. En las propiedades del componente image (imagen) selecciona el campo Picture (imagen) y después selecciona upload new (subir nueva). Cuando subas la imagen a AppInventor verás que aparece en la pantalla de tu dispositivo.
12. Puedes cambiar el tamaño de la imagen jugando con los valores de ancho y largo en sus propiedades. Procura usar una imagen pequeña que no ocupe toda la pantalla.
13. Para añadir un sonido seguimos los mismos pasos que al añadir una imagen. En este caso el componente se llama Sound (Sonido) y se encuentra dentro de la paleta Media. Coge este componente y arrástralo a tu programa igual que hemos hecho con el componente imagen. Verás que el componente de sonido no es visible en la pantalla de tu dispositivo pero aparece aquí:



Al seleccionarlo verás en sus propiedades un campo origen (Source). Si seleccionas este campo puedes elegir el archivo de sonido que quieres usar o subir uno nuevo desde tu ordenador aquí:



Ya has visto lo fácil que es añadir componentes usando el diseñador de AppInventor. El siguiente paso es programar lo que queremos que haga nuestra aplicación. En este caso queremos que al pulsar el botón cambie el texto que teníamos al principio y suene un sonido.

Para hacer esto vamos a usar el editor de bloques.

El editor de bloques

La ventana del editor de bloques está dividida en dos partes. A la izquierda verás todos los bloques que podemos usar en nuestros programas divididos en categorías y a la derecha un espacio en blanco a donde arrastrar los bloques que queremos usar.

Hay tres categorías de bloques: La categoría myBlocks/mis bloques incluye todos los componentes que antes hemos añadido a nuestra aplicación desde el diseñador. Ahí verás los componentes Image1, mi_boton, mi_etiqueta, Screen1 y Sound1. No olvides que esta lista cambiará en función de los componentes que uses en el Diseñador por cada nueva aplicación que hagas.

Los componentes de esta lista tendrán el nombre que les hayamos puesto desde el Diseñador, p. ej.: mi_boton y mi_etiqueta. De lo contario AppInventor les asigna un nombre por defecto que nos permite identificar el tipo de componente, p. ej.: label1, textbox2, imagen1, etc.

También verás un componente llamado Screen1 que corresponde a la pantalla principal de tu programa y que veremos más adelante.

Ya hemos visto que cada componente que usamos en el Diseñador sirve para hacer cosas distintas. Para dar instrucciones (programar) lo que queremos que haga cada componente usamos bloques. Imagina que cada bloque es una orden que le das al componente para que haga algo en concreto. Por ejemplo, en el componente Camera (cámara) tenemos un bloque para hacer una foto (TakePicture), en el componente etiqueta (Label) tenemos un bloque para cambiar el color del texto (TextColor), etc.

Cada bloque representa una acción y juntando varios bloques creamos una secuencia. Verás que los bloques se parecen a piezas de un puzle, de esta forma puedes unir una pieza detrás de otra sin riesgo a equivocarte.

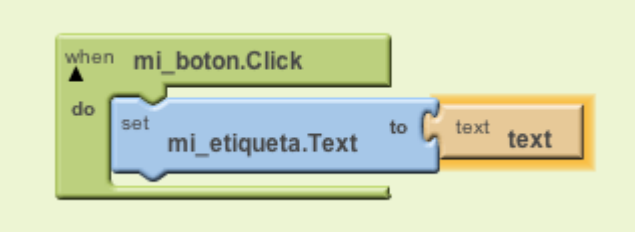
Los bloques se identifican por el nombre del componente, p.ej.: Camara1, seguido de un punto y el nombre del bloque, p. ej.: Camara1.TakePicture, Label2.FontSize, etc.

La categoría de bloques llamada Built-in corresponde a bloques que siempre están disponibles para cualquier aplicación y que te permiten controlar tu programa, hacer operaciones matemáticas, trabajar con cadenas de texto, etc.

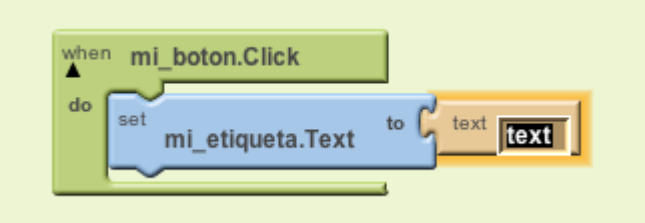
Tu primer programa

1. Dentro de My Blocks selecciona el componente mi_boton. Verás que se despliega una ventana con todas las acciones y propiedades de este componente. Son los bloques que pertenecen a este componente (mi_boton) por ser un componente del tipo botón (button). Más adelante veremos para qué vale cada uno de estos elementos. De momento fíjate que cada uno de estos bloques se llama igual que el nombre que hayamos usado para este componente en el Diseñador seguido de un punto y del nombre de la acción o de la propiedad correspondiente; en este caso mi_boton.
2. El primer bloque de la lista de color verde es "mi_boton.click". Los bloques de color verde controlan los eventos que se producen en tu móvil o tablet. P. ej.: cuando pulsas un botón, cuando mueves el teléfono, cuando recibes un mensaje, etc. Verás que estos bloques tienen escrito "when...do" (cuando...hacer) o dicho de otro modo, "cuando pase esto..haz esto..". En este caso este bloque verde mi_boton.Click se activa si este botón en concreto ha sido pulsado y de ser así activa a su vez los bloques que se encuentren dentro de él. Dicho de otro modo, "cuando este botón se pulse..haz esto". Vamos a verlo con un ejemplo.
3. Selecciona este bloque y colócalo en la pantalla donde te resulte más cómodo.
4. Selecciona ahora el componente mi_etiqueta dentro de la categoría My Blocks que corresponde al componente etiqueta que hemos colocado desde el diseñador. Verás que se despliega nuevamente una ventana con todos los bloques que tiene este componente.
    1. Las propiedades de este componente aparecen de color púrpura. Por ejemplo, el tamaño de la fuente se encuentra guardado en el bloque mi_etiqueta.FontSize, el color del texto en el bloque mi_etiqueta.TextColor, el propio texto de la etiqueta dentro del bloque mi_etiqueta.Text.
    2. Para cada propiedad (bloques de color púrpura) existe otro bloque complementario de color azul para cambiar el valor de esa propiedad. Estos bloques azules que nos permiten cambiar el valor de las propiedades comienzan con la palabra "set" (establecer/poner) y terminan en "to".
    3. Por ejemplo, el texto de nuestra etiqueta mi_etiqueta (recuerda que habíamos puesto "¡Hola mundo !") se encuentra en el bloque púrpura (propiedad) mi_etiqueta.Text . Para cambiar este texto tendríamos que usar el bloque azul complementario "set mi_etiqueta.Text to" y poner el nuevo texto. Vamos a verlo siguiendo con nuestro ejemplo.
5. Busca el bloque "set mi_etiqueta.Text to" y colócalo dentro del bloque verde (evento) mi_boton.Click así:

6. Pincha sobre la categoría de bloques Built-in y dentro de ella, en la categoría Text. El primer bloque se llama "text text". Arrástralo y júntalo por la derecha al bloque azul "set mi_etiqueta.Text" de tu programa así:

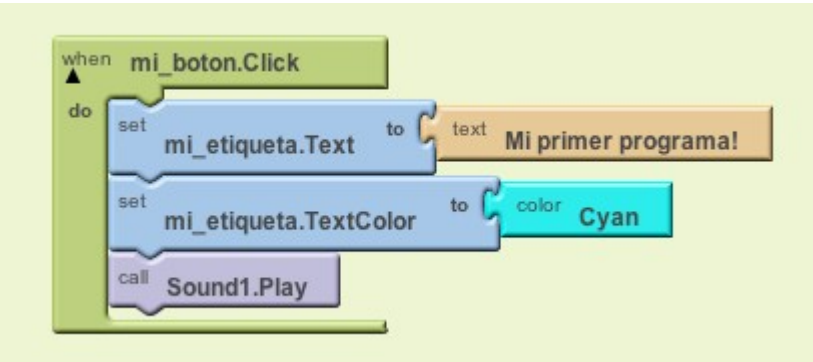


7. Si pinchas encima del texto text en negrita puedes cambiar el texto. Así:



8. Cambia el texto por "Mi primer programa!"
9. Vuelve a la categoría My Blocks y dentro del componente mi_etiqueta coge el bloque "set mi_etiqueta.TextColor to" y colócalo dentro del bloque verde mi_boton.Click de la misma forma que hemos hecho antes.
10. Ahora dentro de la categoría Built-in y dentro de Colors (colores) selecciona el bloque con el color que más te guste y arrástralo a tu programa para unirlo por la derecha al bloque azul "set mi_etiqueta.TextColor to".
11. Por último vuelve a la categoría My Blocks y selecciona el componente Sound1 que corresponde con el sonido que hemos añadido antes desde el Editor de bloques. Verás un bloque morado que se llama "call Sound1.Play" y que sirve para que tu programa toque el sonido. Arrástralo dentro del bloque verde junto con los otros bloques.

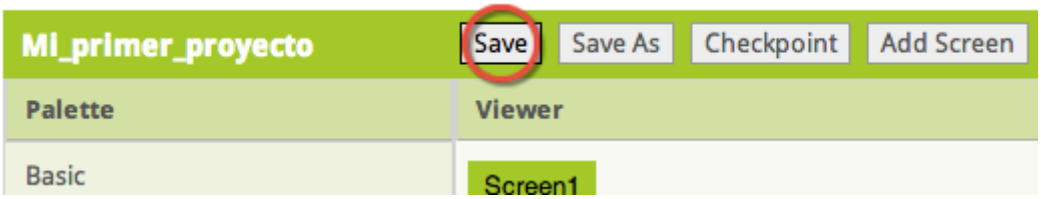
Tu programa debería de quedar así:



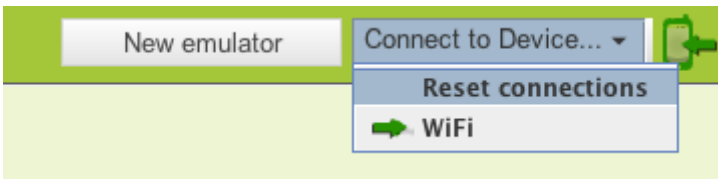
Si todo ha funcionado correctamente deberías de ver los resultados en tu dispositivo. Compruébalo pulsando el botón.

Enhorabuena, ¡acabas de programar tu primera aplicación!

No olvides guardarlo pulsando aquí:



Puedes reiniciar tu programa desde el Editor de bloques volviendo a conectar con tu dispositivo así:



Diseñando la interfaz

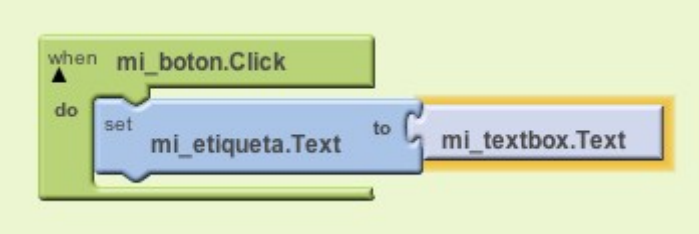
Componentes para pedir información

Textbox

Este componente te permite crear un campo de texto donde el usuario puede escribir un texto o un número que se guardará dentro de la propiedad del componente llamada ".Text".

Lo mismo que hicimos con el botón de nuestro primer programa ahora cogeremos este componente y lo pondremos en la interfaz para ver como queda, renómbralo y quítale el texto que aparece por defecto. A continuación de este componente vamos a poner otro componente, concretamente un Label al que le vamos a modificar lo que muestra como etiqueta.

Luego nos vamos al Editor de Bloques y haremos esto y lo probamos en el emulador:



Listpicker

Este componente nos permite mostrar una lista de opciones para que usuario de la aplicación escoja la que quiera. Volvamos al Diseñador, seleccionemos este componente y añadámoslo a la pantalla del programa entre el componente mi_textbox y mi_boton.

1. Cambia el nombre del componente por "genero" y en la propiedad "ElementsFromString" pon lo siguiente: "mujer,hombre".
2. Cambia la propiedad Text y escribe "género".
3. Vuelve al editor de bloques.
4. Dentro de la categoría Built-in / Text coge un nuevo bloque de texto "text text" y añádelo al final del bloque "make text" que ya tenías antes dentro del bloque mi_boton.Click.
5. Cambia el texto de este bloque y pon: ". Eres ". Fíjate que este bloque "make text" va creciendo uniendo cada bloque que le vamos añadiendo para crear una frase completa. Ahora vamos a usar la propiedad genero.Selection dentro del componente genero en My Blocks que acabamos de crear y que guarda la opción que hayamos elegido de la lista.
6. Selecciona el bloque (propiedad) genero.Selection y añádelo al final del bloque

make text.

7. En el Diseñador dentro de las propiedades de este componente podemos definir un valor por defecto, eso lo tenemos en la propiedad de “Selection”
8. Prueba tu programa.

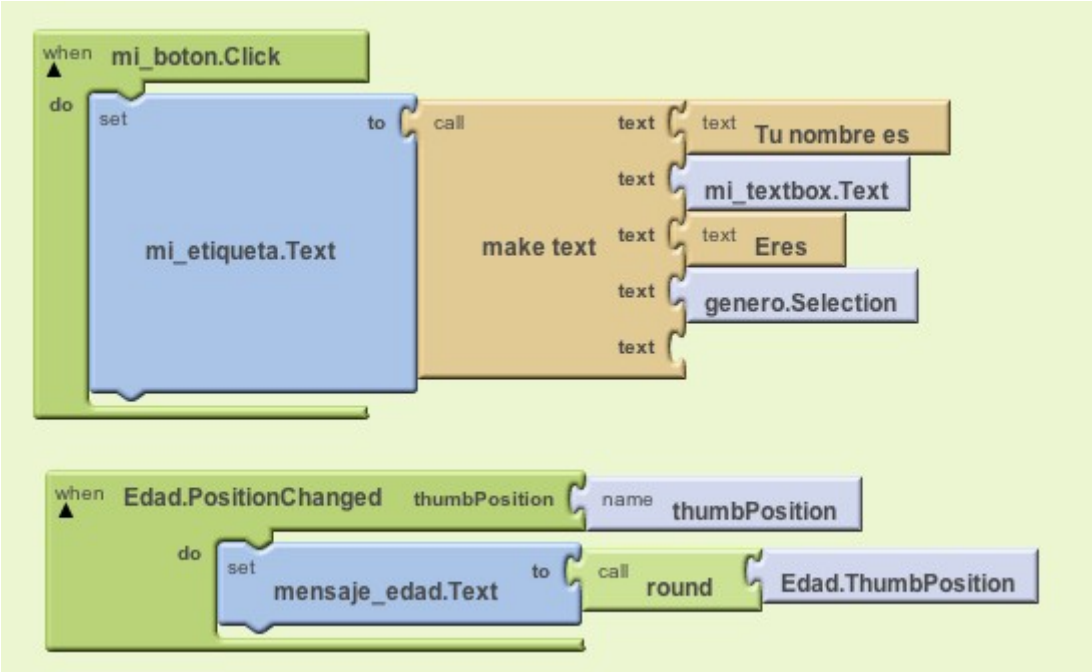
Slider

Este componente permite que el usuario seleccione un valor deslizando un deslizador gráfico entre los extremos de una pista que corresponde a un rango de valores. En nuestro ejemplo vamos a usarlo para introducir la edad.

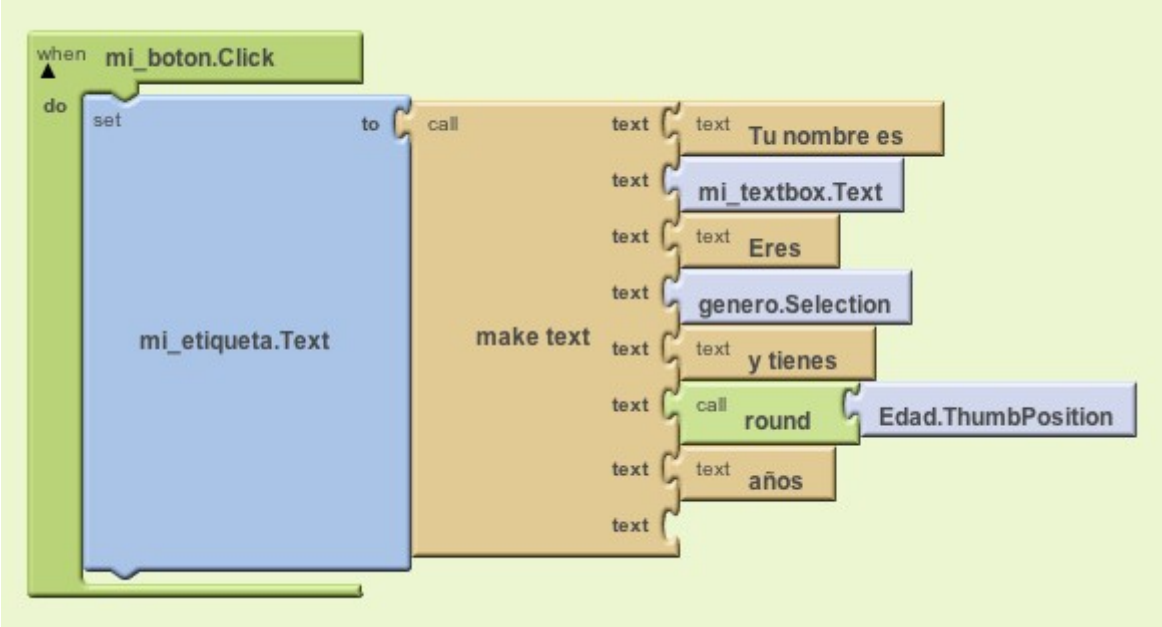
Vamos a usarlo para pedir un número. Debajo del componente anterior ListPicker pondremos un componente HorizontalArrangement en el que dentro de el meteremos 2 Labels, el primer Label quedará a la izquierda y haremos que ponga “Tu edad es: “ y el segundo Label quedará a la derecha dejándolo sin texto ninguno.

Tenemos que modificar los valores límite que admite el Slider, el valor mínimo será 0 y el valor máximo será 100.

Luego tendremos que hacer un bloque como el siguiente:



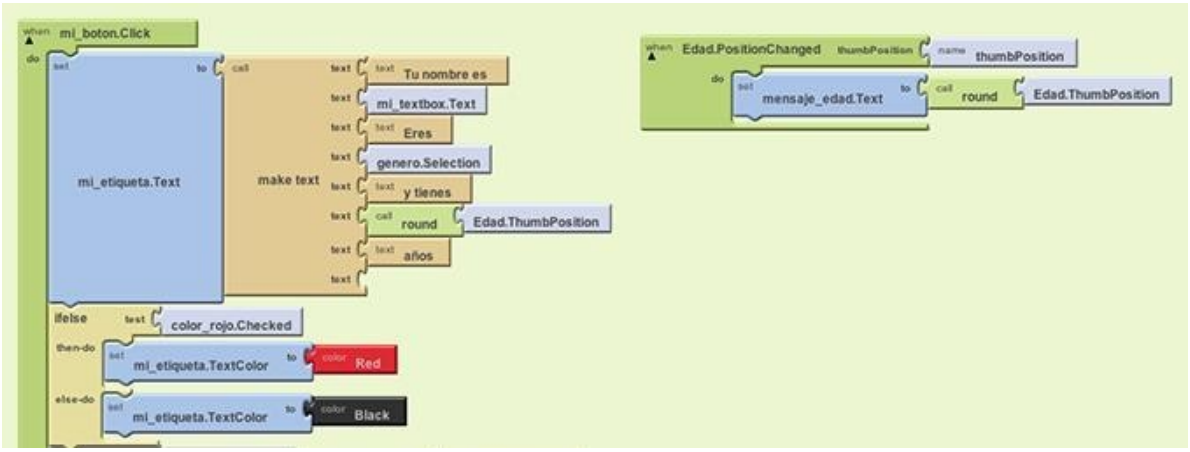
Y a continuación de hacer esto tendremos que dejar el programa como sigue:



Checkbox

Este componente Checkbox se usa para permitir al usuario elegir una opción que puede estar activada o no. Vamos a usarlo para cambiar el color del texto de nuestra frase si el usuario quiere resaltar el resultado.

Introduzcamos un componente de este tipo para hacer que el usuario pueda remarcar o no el texto con otro color. Tendremos que conseguir algo así pero sin los bloques de “TextToSpeech1” y “Sound1”:



Variables

Una variable es un espacio donde podemos guardar información de forma temporal, por ejemplo, podemos crear una variable llamada marcador donde guardar la puntuación de un videojuego mientras el usuario está jugando.

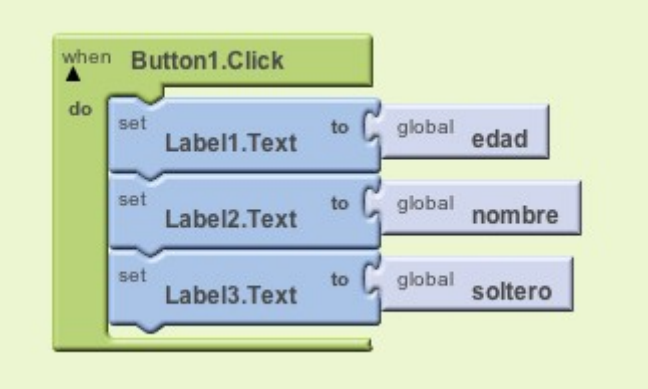
En este capítulo vamos a aprender a usar variables en nuestras aplicaciones.

Creando variables

Vamos a aprender a usarlas con un sencillo ejemplo.

1. Comienza un nuevo proyecto y llámalo "variables".
2. Desde el Diseñador coge tres etiquetas y un botón.
3. Cambia el texto del botón y pon, p. ej.:"ok". Lo demás lo puedes dejar como está. Ahora vamos al Editor de bloques.
4. Para crear variables nuevas usamos el bloque "def variable as" de color azul que encontrarás dentro de Built-in / Definition. Coge tres bloques iguales y ponlos en tu programa.
5. Verás que AppInventor les ha dado un nombre automáticamente pero vamos a cambiarlos. A la primera variable llámala nombre, a la segunda edad y a la última llámala soltero. Ahora vamos a asignar valores a estas variables.
6. Para la primera, nombre vamos a usar una cadena de texto. Pincha en cualquier parte de la pantalla para sacar el menú rápido de bloques y selecciona text / text (texto) Esto crea un bloque nuevo de texto para que lo pegues a la variable nombre. Ahora cambia el texto y pon el nombre que quieras.
7. Para la segunda variable edad vamos a crear un bloque numérico. Haz lo mismo que antes pero selecciona esta vez el bloque math / 123 y pega el nuevo bloque a la variable edad. Cambia el número y pon la edad que quieras.
8. Por último, vamos a crear una variable lógica (booleana). Pincha en la pantalla para abrir el menú rápido y selecciona logic (lógica) y coge un bloque true (verdadero) o false (falso) y pégalo a la variable soltero. Ya tenemos tres variables creadas y hemos asignado un valor a cada una. Vamos a mostrarlas ahora en nuestra aplicación.
9. Desde My Blocks coge el evento button1.Click y ponlo en tu programa.
10. Ahora vamos a cambiar las etiquetas. Coge el bloque "set Label.Text to" de cada una de las tres etiquetas y ponlas dentro del evento .Click del botón,
11. Sólo queda pasarle a cada etiqueta el valor que queremos que muestre y que corresponde a cada una de las variables.

12. Cuando creamos variables en nuestras aplicaciones estas se encuentran siempre dentro de My Blocks/My definitions así que tenemos que cogerlas desde ahí. Verás que las variables que hemos creado aparecen como bloques azul claro llamados "global" más el nombre de la variable. Junto con cada variable aparece otro bloque complementario llamado "set global" más el nombre de la variable. Estos bloques nos sirven para cambiar el valor de la variable. En un segundo veremos cómo usarlos. De momento coge cada una de las variables (recuerda que son las de color azul claro y empiezan por "global") y pégalas a cada bloque "set label.Text to".



13. Prueba a pulsar el botón. Verás que el valor que contiene cada una de las variables aparece en pantalla simplemente usando el bloque con el nombre de la variable. Vamos a ver ahora cómo cambiar el valor de las variables directamente desde nuestro programa.

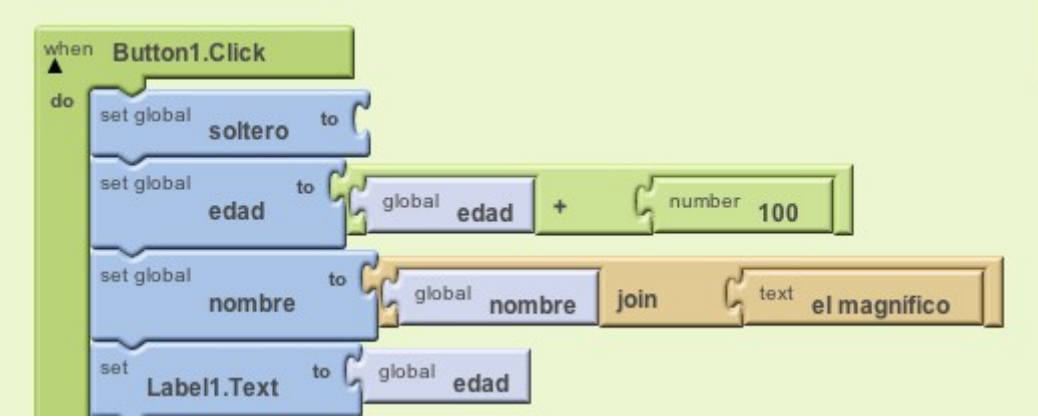
Modificando su contenido

Ya hemos visto que el valor de las variables se puede cambiar usando el bloque "set global" correspondiente a cada variable dentro de My Blocks/My definitions.

1. Coge el bloque "set global" de la variable nombre y ponlo dentro del evento .Click justo antes de los bloques "set label.Text" al principio.
2. Haz lo mismo con el bloque "set global" de la variable edad. Ponlo también al principio.
3. Por último, haz lo mismo con el bloque "set global" de la variable soltero. Ponlo también al principio.
4. Ahora pincha en la pantalla para sacar el menú rápido de bloques y selecciona text / join. Este bloque te permite unir dos cadenas de texto. Pégalo al bloque "set global nombre".
5. Para la primera cadena de texto vamos a usar la variable nombre (ya sabes dónde encontrarlo) y para el segundo bloque de texto crea un nuevo bloque de texto y cambia el texto por " el magnífico".
6. Ahora vamos con la edad. Saca el menú rápido de bloques y selecciona math y la

suma (+).

7. Pégalo al bloque "set global edad".
8. Para el primer valor vamos a usar la variable edad y para el segundo crea un nuevo bloque numérico (math/123) y añade por ejemplo 100 así:



9. Por último, crea un bloque lógico nuevo y pégalo al bloque "set global soltero". Cambia el valor por el contrario al que usaste al principio.
    1. Verás ahora que aunque habíamos asignado a cada variable un valor inicial (el nombre, la edad y el estado) ahora hemos modificado el valor de las variables dentro de nuestro programa, añadiendo una cadena de texto al nombre, 100 años más a la edad y cambiando el estado.
    2. Esta es la forma de crear, usar y manipular variables dentro de nuestros programas.
10. ¿Qué pasa si pulsamos el botón más veces?

Componentes con variables

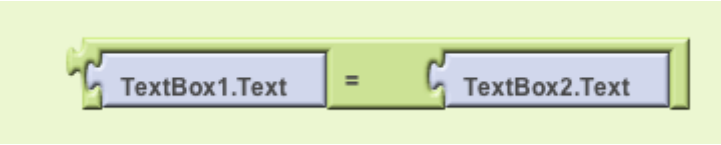
Muchos componentes de la pantalla como los botones, las etiquetas y las cajas de texto guardan información en variables. Por ejemplo, cuando asignamos un texto a una etiqueta usando el bloque "set label.Text" este texto se guarda en realidad dentro de una variable llamada "label.Text" que encontrarás dentro del componente.

El tamaño del texto o la fuente también se encuentran guardados dentro de variables. De esta manera, no siempre es necesario crear variables y basta con que uses las variables que tienen los propios componentes de la interfaz de usuario.

Bloques de control

Comparando números

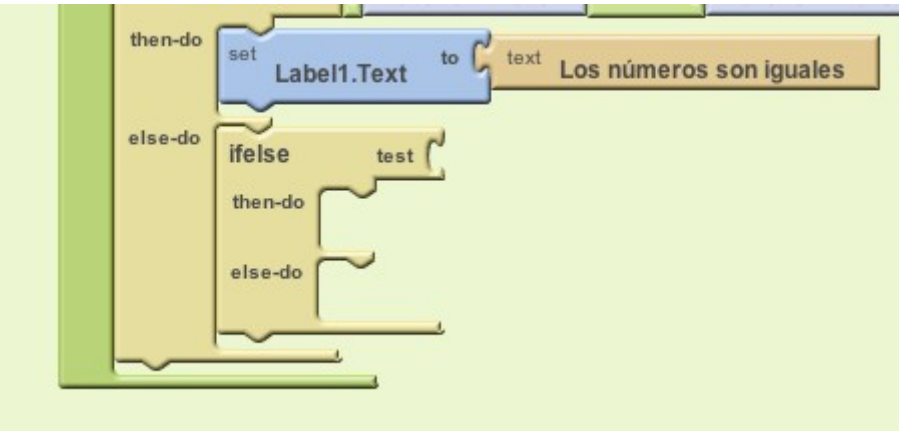
1. Vamos a hacer una pequeña aplicación que nos pregunte por dos números y nos diga si el primer número es igual, mayor o menor que el segundo y para hacerlo vamos a aprender a usar bloques de control. Empieza un proyecto nuevo y llámalo "comparando_numeros".
2. Desde el Diseñador añade dos componentes textbox, después añade debajo una etiqueta label y por último un botón. Puedes quitar los textos de estos componentes y dejarlos vacíos. Cambia también si quieres el texto del botón.
3. Ahora vamos al Editor de bloques.
4. Coge el evento button.Click y ponlo en tu programa. Cuando pulsemos el botón queremos comprobar en primer lugar si el primer número es igual que el segundo y si es así mostrar el mensaje en la pantalla. Si no fuese igual continuaremos con la comprobación para saber si es mayor o menor. Vamos a ver cómo hacer estas comprobaciones.
5. Dentro de la categoría Built-in / Control coge el bloque amarillo "ifelse then-do else-do" y ponlo dentro del evento.
6. Verás que este bloque tiene un campo llamado test. Este es el campo que tendrá la expresión lógica "pregunta" que queremos hacer. En nuestro caso: ¿es el primer número igual que el segundo?
7. Como estamos comprobando dos números, abre el menú rápido de bloques y coge el bloque Math/ = que nos permite preguntar si dos números son iguales.
8. En el primer espacio pon el bloque (variable) TextBox1.Text que contiene el número que haya puesto el usuario en el primer campo (recuerda que son de color azul claro).
9. En el segundo espacio haz lo mismo con el segundo componente TextBox2.Text. Así:



10. Ahora pega este bloque al campo test del bloque de control ifelse. Ya tenemos montada la pregunta, ahora tenemos que decidir qué vamos a hacer si la respuesta es correcta (verdadera) o no (falsa).
11. Dentro del bloque de control ifelse verás un primer espacio llamado then-do (entonces haz esto). Aquí pondremos lo que queremos que ocurra si la respuesta

es verdadera (los dos número son iguales). Coge un bloque "set label1.Text to" y ponlo dentro.

12. Ahora crea un bloque de texto y pon dentro "Los número son iguales" y pégalo al bloque "set label1.Text".
13. Ahora en el espacio llamado else-do (sino haz esto) pondremos los bloques que queremos activar si los números son distintos.
14. Ya hemos preguntado si los números son iguales, si la respuesta es falsa (son distintos) ahora vamos a comprobar si el primer número es mayor o menor que el segundo. Vuelve a coger otro bloque de control "ifelse" como el anterior y colócalo dentro de este espacio. Así:



15. Vuelve a crear un bloque Math pero esta vez > que nos permite preguntar si un número es mayor que otro.
16. En el primer espacio pondremos el bloque (variable) TextBox1.Text y en el segundo el bloque (variable) TextBox2.Text. Pégalo ahora al segundo bloque de control ifelse en el campo test.
17. Recuerda que si la respuesta es verdadera (el primer número es mayor) entonces se activarán los bloques del primer espacio then-do, así que dentro haremos lo mismo que antes cambiando el valor de la etiqueta. Selecciona el bloque azul "set Label1.Text to" que usamos antes y duplícalo.
18. Coge la nueva copia y ponla en este espacio.
19. Ahora cambia el texto y pon "El primer número es mayor".
20. Si la respuesta es negativa (falso) quiere decir que el segundo número es menor que el primero. Vuelve a duplicar el bloque "set Label1.Text" anterior y ponlo en el espacio else-do.
21. Ahora cambia el texto y escribe "El segundo número es mayor".

Procedimientos

Un procedimiento es una parte del programa que realiza una acción específica que hemos definido con anterioridad y a la que podemos llamar tantas veces como queramos usando un identificador (su nombre) como hacemos cuando usamos variables. En programación a los procedimientos también se les conoce como rutinas, subrutinas o subprogramas.

Utilizar procedimientos en nuestros programas nos ahorra tiempo, hace los programas más sencillos de entender y nos permite reutilizar los procedimientos que ya hemos hecho en otros programas.

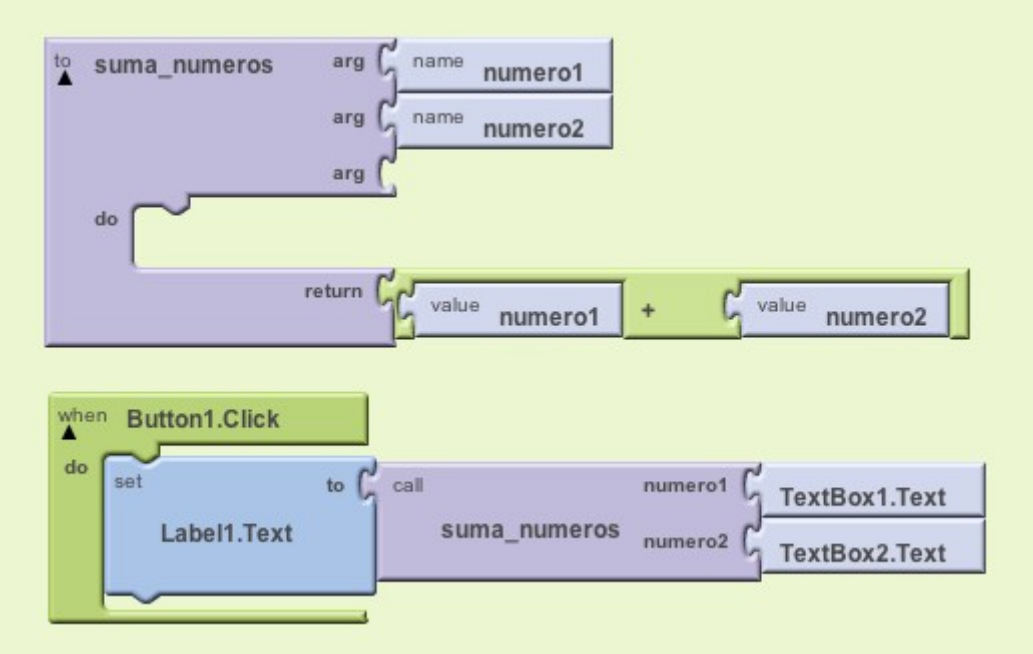
Sumando números

Vamos a hacer una pequeña aplicación para demostrar la utilidad de los procedimientos en nuestros programas. En este ejemplo vamos a sumar dos números llamando a un procedimiento al que vamos a llamar suma_numeros usando dos parámetros con los números que queremos sumar. El procedimiento nos devolverá la suma de ambos.

1. Crea un proyecto nuevo, llámale "suma_numeros" y añade desde el Diseñador los siguientes componentes: Dos componentes textbox, un botón y una etiqueta.
2. Ahora vamos al Editor de bloques.
3. Crear y usar procedimientos es muy similar a crear y usar variables. Dentro de Built-in/Definitions coge el bloque "to procedureWithResult" (procedimiento con resultado) y ponlo en tu programa.
4. Este bloque nos permite crear procedimientos que nos devuelven un valor al terminar. Cambia el nombre del procedimiento y llámalo suma_numeros.
5. Para nuestro procedimiento vamos a usar dos argumentos. Cada uno corresponde a los números que queremos sumar. Para definir los argumentos dentro de Built-in/Definitions tenemos un tipo de bloque llamados "name name". Coge dos de estos bloques y pégalos al procedimiento que acabamos de crear como argumentos en arg.
6. Cambia el nombre del primer argumento y llámalo numero1, al segundo llámalo numero2. Ahora vamos con la suma. Fíjate que el procedimiento suma_numeros tiene un campo llamado return. En este campo pondremos el resultado de las operaciones que queremos devolver cuando llamemos a este procedimiento. En este caso, el resultado de la suma. Como lo que queremos hacer es muy sencillo podemos hacer la operación directamente dentro de return. Vamos a ver cómo.
7. Crea un bloque math + para sumar dos números.
8. En el primer espacio pon el bloque value numero1 y en el segundo el value

numero2. Recuerda que los encontrarás ahora dentro de My Blocks/My definitions.

9. Finalmente junta la suma al argumento return del procedimiento. Como este procedimiento es muy sencillo no hemos necesitado hacer nada dentro del espacio do del procedimiento y ha sido suficiente con hacer la suma directamente en el campo return.
10. Ahora coge el evento .Click de nuestros botón.
11. Vamos a mostrar el resultado en la etiqueta así que coge el bloque "set label1.Text to" y ponlo dentro del evento .Click.
12. Como lo que queremos mostrar es el resultado de sumar los dos números y ese resultado es el que nos devuelve nuestro procedimiento, podemos unirlos directamente. Coge el bloque "call suma_numeros" dentro de My Blocks/My definitions y pégalo al bloque "set label1.Text".
13. Ahora únicamente necesitamos pasarle los datos que queremos sumar a nuestro procedimiento que están en "textBox1.Text" y "TextBox2.Text" respectivamente. Te tiene que quedar así:



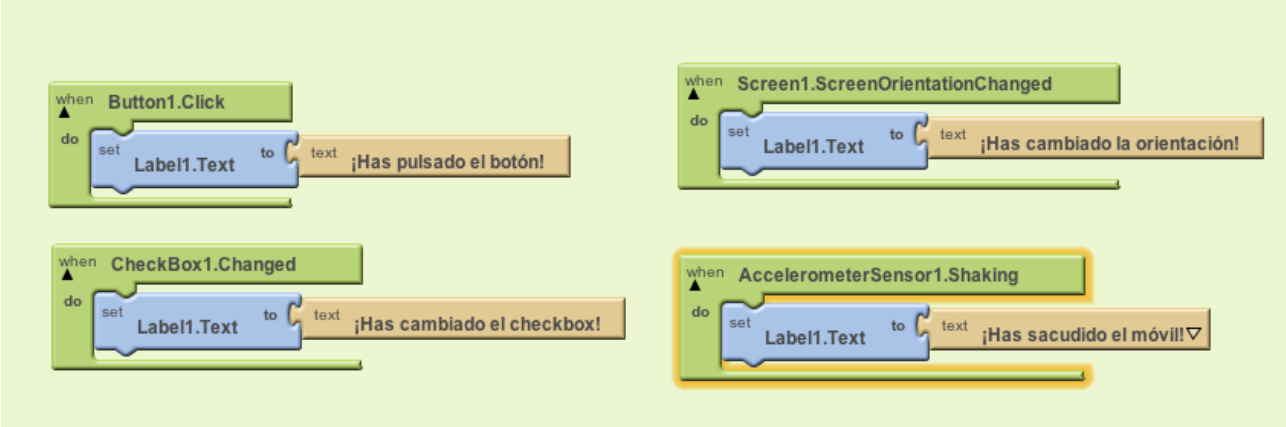
A partir de este momento si quisieses sumar otros números dentro de tu programa sólo tienes que llamar a tu procedimiento desde donde quieras. Como el procedimiento nos devuelve un número (en este caso la suma) podemos pegarlo en cualquier sitio tal y como si fuese un bloque numérico cualquiera. En este caso lo estamos pegando a una etiqueta para que nos muestre el valor en pantalla.

Eventos

Nuestros ordenadores son capaces de hacer muchas cosas a la vez (multitarea).

Podemos navegar por Internet mientras recibimos mensajes de correo electrónico, podemos empezar a ver vídeos on-line mientras el resto del video todavía se está descargando de Internet, podemos escuchar música mientras el navegador nos da instrucciones para llegar a nuestro destino, etcétera.

Todo esto es posible gracias a los procesadores que llevan pero también gracias a cómo están diseñadas estas aplicaciones. Vamos a ver un ejemplo en que hay 1 botón y 1 CheckBox además de la pantalla de por sí:



De aquí lo que no nos hará falta será programar el acelerómetro así que tendremos que tener 3 de estos 4 bloques.

Prueba a ejercutar todo esto.

Tipos de Eventos

Eventos de la *Interfaz Gráfica de Usuario*

Generalmente la mayoría de eventos son producidos por el usuario cuando usa el interface de tu aplicación. Ya hemos visto algunos de estos eventos, por ejemplo, cuando el usuario pulsa un botón se produce el evento "button1.Click", etcétera.

Eventos de *componentes no visibles*

Otro tipo de eventos son producidos por algunos componentes que usamos en nuestros programas y que no son visibles. Por ejemplo, el sensor de movimiento dispara un evento cuando sacudimos nuestro dispositivo tal y como hemos visto también en nuestro ejemplo anterior.

Eventos del *reloj / cronómetro*

Todos nuestros dispositivos cuentan con un reloj interno que funciona también como un cronómetro. Una forma de usar este reloj/cronómetro es para hacer que nos

avise (generando un evento) cada cierto tiempo que nosotros queramos. Esto es muy útil cuando queremos hacer algo que dependa de un determinado tiempo. Por ejemplo, mostrar un mensaje durante un tiempo determinado, comprobar algo cada cierto tiempo, etc.

Vamos a ver un ejemplo siguiendo el ejemplo anterior:

1. Vuelve al diseñador y usa el componente clock (reloj) arrastrándolo a tu programa. Verás que este componente no es visible. Ahora selecciona el componente y vamos a fijarnos en sus propiedades.
2. La propiedad más importante se llama TimerInterval y nos permite definir por cuánto tiempo queremos que funcione nuestro cronómetro. El tiempo se mide en milisegundos por lo tanto para que cuente, por ejemplo, cinco segundos, tendremos que escribir 5000. Hazlo ahora y pon 5000. Cuando el intervalo de tiempo se cumpla nuestro reloj volverá a contar desde cero. Imagina el sonido del tambor de una banda de música; el tiempo que pasa entre cada golpe del tambor es el intervalo.
3. La propiedad TimerAlwaysFires nos permite decidir si queremos que el reloj cuando llegue al final del intervalo de tiempo active o no un evento. Mientras esta propiedad esté activa, al cumplirse el intervalo se generará un evento.
4. Para nuestro ejemplo déjalo activo.
5. Por último la propiedad TimerEnable nos permite decidir si el reloj estará contando desde el momento de arrancar nuestro programa o lo activaremos desde el Editor de bloques después. Déjalo también activo. Ahora vamos al editor de bloques.
6. Dentro de la categoría My Blocks verás el componente Clock1. Si lo seleccionas verás que tiene un evento de color verde llamado Clock1.Timer. Este es el evento que se activará cada vez que nuestro reloj cuente el número de segundos que hemos decidido antes. En nuestro caso 5 segundos. Arrástralo a tu programa. Ahora sólo nos queda decidir qué queremos que pase cuando esto ocurra. Vamos a añadir un sonido y a actualizar la etiqueta como hemos hecho con los otros eventos.
7. Vuelve al Diseñador y coge el componente Sound.
8. En sus propiedades verás que puedes seleccionar el sonido que quieres usar en Source (origen). Puedes usar el sonido beep.mp3 que encontrarás en la librería de sonidos.
9. Volvemos al editor de bloques donde aparecerá nuestro componente de sonido Sound1. Si lo seleccionas verás que tiene un bloque llamado "call Sound1.Play". Coloca este bloque dentro del evento del reloj "when Clock1.Timer".
10. De esta forma cada vez que el cronómetro cuente tres segundos se activará este evento y sonará una alerta. Finalmente vamos a actualizar la etiqueta. Duplica

cualquiera de los bloques (set label1.Text) que has usado antes y cambia el texto por: "¡Se ha activado el reloj!"

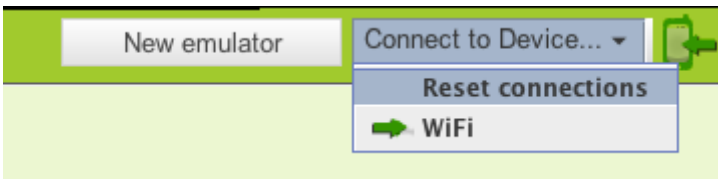
Eventos del componente *pantalla*

Recuerda que todos nuestros programas tienen un componente llamado screen1 que corresponde a la pantalla principal de nuestra aplicación. En nuestro ejemplo anterior hemos usado este componente para detectar cuando el usuario cambia la orientación del dispositivo usando el evento screen1.ScreenOrientationChanged.

Este componente también tienen un evento llamado .Initialize. Este evento se activa cuando la pantalla principal se inicia, es decir, cuando nuestra aplicación arranca por primera vez, que es el momento en que nuestra aplicación inicia la pantalla principal. Podemos usar este evento para preparar cosas dentro de nuestro programa, mostrar un mensaje de bienvenida, etc.

Vamos a aprender a usar este evento para crear un mensaje de inicio o bienvenida a nuestros programas. Para empezar arrástralo a tu programa.

1. Vuelve al Diseñador y coge el componente Notifier (notificador) dentro de la categoría de componentes Other stuff (otras cosas) y arrástralo a tu programa. Verás que este componente no es visible y que tampoco tiene propiedades.
2. Este componente nos permite mostrar mensajes en la pantalla de nuestro dispositivo. Vamos a usarlo en nuestro ejemplo para mostrar un mensaje de bienvenida a los usuarios. Vuelve al Editor de bloques.
3. Como siempre, verás que el componente Notifier1 aparece dentro de la lista My Blocks. Si lo seleccionas verás un bloque llamado "call Notifier1.ShowMessageDialog" arrástralo dentro del evento "when Screen1.Initialize do" que hemos usado antes. Ahora sólo queda crear tres bloques de texto que corresponden al título (title) que queremos darle a la ventana con el mensaje, al mensaje propiamente dicho (message) y al botón que aparecerá junto con el mensaje (button).
4. Por ejemplo: "Bienvenido", "Espero que mi aplicación te guste" y "Continuar" respectivamente. Pégalos al bloque "call Notifier1.ShowMessageDialog" donde corresponden.
5. Para comprobar que funciona sólo nos queda reiniciar nuestra aplicación porque de lo contrario este evento nunca se activará. Para reiniciar nuestra aplicación basta con que selecciones de nuevo la conexión con tu dispositivo aquí.



Nos preguntará si queremos reiniciar la conexión y le decimos que sí. Si no hacemos esto no nos funcionará nada de lo que hemos hecho.

Dibujemos

Ahora haremos lo siguiente para dibujar en nuestra pantalla.

Distribuir componentes

Para organizar de forma más fácil los componentes en la pantalla de tu aplicación existe una categoría llamada Screen Arrangement (Organización de Pantalla).

Dentro de ella encontrarás tres componentes que te permiten distribuir fácilmente los elementos siguiendo una disposición horizontal, vertical o como una tabla.

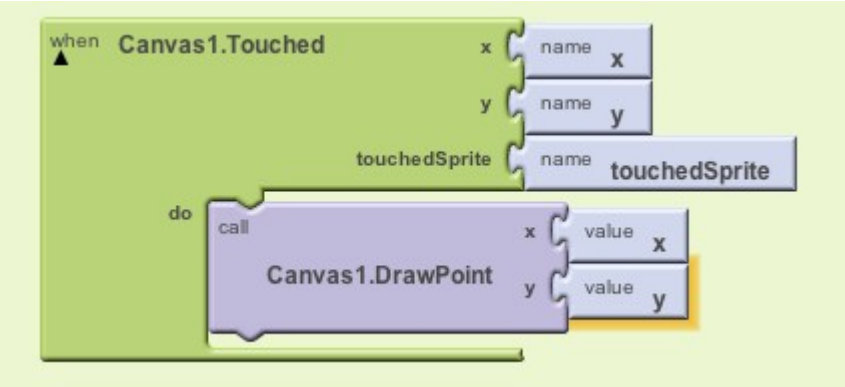
Dibujar

Ya hemos visto que cada aplicación que hacemos con AppInventor tiene un pantalla principal (componente Screen) dentro de la cual colocamos otros componentes como botones, texto, etc.

El componente Canvas (Lienzo de Dibujo) es otro componente más que podemos incluir dentro de nuestra pantalla principal y que nos va a permitir dibujar y hacer animaciones.

Otra característica del componente Canvas (Lienzo) es que es un espacio sensible al tacto; es decir, el usuario podrá usar los dedos para interactuar con nuestra aplicación, por ejemplo, para dibujar o mover objetos usando gestos como tocar, arrastrar, etc. Vamos a ver varios ejemplos.

1. Empieza un nuevo proyecto y llámalo “Canvas”.
2. En las propiedades de la pantalla (screen) cambia la propiedad AlignHorizontal por Center (centrado).
3. Desactiva la propiedad Scrollable y ahora selecciona el componente Canvas (Lienzo) y colócalo en la pantalla de tu programa.
4. Cambia el color de fondo y las propiedades Width (ancho) y Height (largo) a "fill parent". Verás como el componente Canvas (Lienzo) ocupa ahora toda la pantalla.
5. Por último añade un botón por encima o por debajo del lienzo. Cambia el nombre del botón por "boton_borrar" y cambia el texto del botón por "Borrar".
6. Vamos al Editor de Bloques.
7. Programaremos el evento “.Click” del botón que hemos puesto y cuando el canvas los tocamos con el evento “.Touched”
8. Vamos a pintar puntos en el Canvas teniendo un bloque como el siguiente:

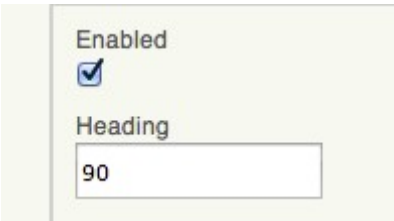


9. También podremos cambiar el color para dibujar, pintar círculos y más.
10. Para dibujar líneas es sencillo también, haremos esto:
    1. Selecciona de nuevo el componente Canvas1 y coge el bloque evento Canvas1.Dragged.
    2. Este evento nos dice si el usuario ha arrastrado (drag) el dedo por el lienzo. Verás que este evento nos pasa nuevos parámetros. Los dos primeros (startX y startY) nos indican dónde ha empezado el usuario a tocar el lienzo aunque ahora no las vamos a usar. Nos interesan las propiedades prevX y prevY y las propiedades currentX y currentY. Este conjunto de propiedades nos dicen desde dónde y hasta dónde está el usuario arrastrando el dedo. Vamos a verlo dibujando una línea.
    3. Vuelve al componente Canvas1 y coge el bloque Canvas1.DrawLine (Dibujar línea) y arrástralo dentro del evento Canvas1.Dragged.
    4. Este bloque nos pide las coordenadas del inicio y final de la línea que queremos dibujar. Nuevamente esta información ya la tenemos porque nos la está facilitando el evento.
    5. Desde My Blocks / My definitions coge las propiedades prevX y prevY y pégalas al bloque Canvas1.DrawLine X1 e Y1 respectivamente. Estas son las coordenadas de inicio de la línea que corresponden a las coordenadas donde el usuario ha empezado a arrastrar el dedo.
    6. Ahora desde My Blocks / My Definitions coge también las propiedades currentX y currentY y pégalas al mismo bloque en X2 e Y2. Estas serán las coordenadas donde termina la línea y que corresponden a donde el usuario tiene el dedo en este momento. Pruébalo y verás como lo entiendes mejor.
    7. Ahora que entiendes cómo funciona el evento Canvas.Dragged puedes probar a cambiar el bloque .DrawLine por .DrawPoint
    8. Recuerda que el evento .Touched también funciona así puedes ver la diferencia entre tocar y arrastrar el dedo por la pantalla.

Animaciones - Un viaje a la Luna

Para crear animaciones con AppInventor utilizamos sprites. Los sprites son imágenes que puedes mover por la pantalla siempre dentro de un lienzo (Canvas) Estas imágenes reaccionan a eventos como tocar (touch) o arrastrar (drag) y se pueden mover por el lienzo a una velocidad y dirección determinadas en sus propiedades. Vamos a verlo con un ejemplo.

1. Empieza un proyecto nuevo y llámalo "viaje_luna".
2. En las propiedades de la pantalla (screen) asegúrate de que la propiedad Scrollable está desactivada. Además en la propiedad ScreenOrientation selecciona Portrait (vertical).
3. Ahora añade los siguientes componentes a tu programa: En primer lugar una etiqueta (label), después un lienzo (canvas) y por último el componente OrientationSensor.
4. Selecciona la etiqueta y cambia su nombre por marcador. En la propiedad texto escribe 0.
5. Ahora selecciona el componente canvas y cambia la imagen de fondo (BackgroundImage) y selecciona la imagen estrellas.png que encontrarás aquí.
6. Cambia también las propiedades Width (ancho) y Height (largo) a "fill parent".
7. Ahora vamos a añadir las imágenes para nuestra primera animación. El componente que vamos a usar se llama ImageSprite y lo encontrarás dentro de la categoría Animation (animaciones). Vamos a necesitar dos, así que coge dos componentes y ponlos dentro del canvas.
8. Selecciona el primer componente ImageSprite y cambia su nombre a cohete. Vamos a ver en detalle sus propiedades.
9. La primera propiedad que nos interesa se llama Heading (ángulo de dirección) y nos indica la dirección hacia la que queremos mover nuestra imagen. Podemos utilizar un valor desde 0 hasta 360. 0 indica una dirección de izquierda a derecha, 90 de abajo hacia arriba, 180 de derecha a izquierda y 270 de arriba hacia abajo. Así:



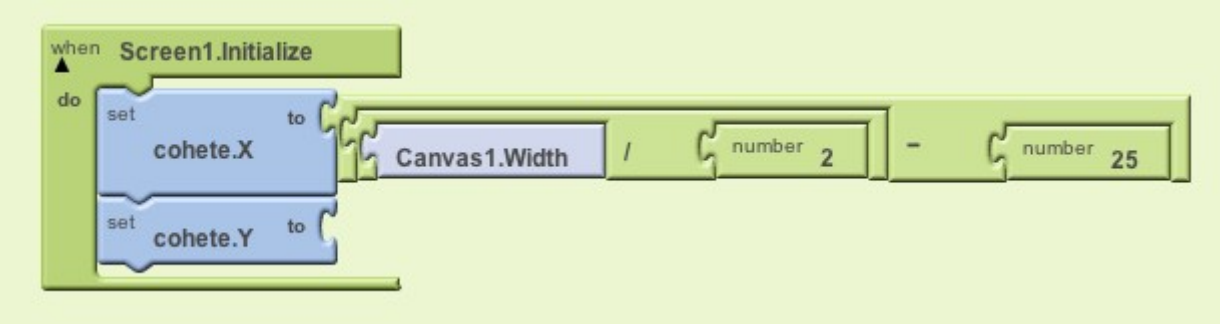
10. En nuestro caso queremos que nuestro cohete vaya subiendo por la pantalla de abajo hacia arriba por lo tanto pondremos en esta propiedad 90.
11. La siguiente propiedad se llama Interval (intervalo) y representa cada cuanto tiempo vamos a mover la imagen. Para nuestro ejemplo vamos a empezar

poniendo 1000. Este dato se representa en milisegundos por lo tanto 1000 =1 segundo.

12. En la propiedad Picture (imagen) vamos a seleccionar nuestra imagen para el cohete. Puedes encontrar la imagen que necesitamos aquí se llama cohete.png.
13. Desactiva la opción Rotates.
14. La siguiente propiedad que nos interesa se llama Speed (velocidad) y nos indica el número de pixeles que se va a mover nuestra imagen en el intervalo que hemos definido antes. Para empezar pon 1.
15. Por último, cambia el ancho (width) y largo (height) de nuestro cohete por 50 x 50. Si todo ha funcionado correctamente nuestro cohete se debería de estar moviendo por la pantalla pero muy despacio. La razón está en las propiedades Interval y Speed que hemos usado. Ahora vamos a entender mejor cómo usarlas.
16. Antes de nada, si tu cohete llega al margen superior de la pantalla se detendrá. Para volver a ponerlo en su posición original basta con que lo muevas un poco desde el Diseñador. De esta manera volverá a su posición original. Después veremos cómo manejar esto desde el Editor de bloques.
17. Hasta ahora le hemos pedido a nuestro cohete que cada segundo (Interval = 1000) se moviera 1 pixel (Speed = 1) Si cambias la propiedad interval y vas poniendo valores cada vez más pequeños, el cohete avanza más rápido porque se mueve cada menos tiempo. Si aumentas el número de pixeles que avanza nuestra imagen (Speed) también avanzará más rápido por que los saltos serán mayores. Recuerda que la velocidad depende de la distancia y del tiempo. Prueba a cambiar estos valores hasta que encuentres el que más te guste dependiendo de tu dispositivo. Un intervalo de 25 y una velocidad de 5 suelen funcionar bien.
18. Ahora vamos al Editor de bloques. En primer lugar vamos a colocar nuestro cohete en la posición inicial que queremos. Como se trata de la posición inicial lo lógico es hacerlo nada más arrancar nuestro programa, por lo tanto vamos a usar el evento Screen1.Initialize que se activa al inicio como hemos visto ya. Desde el componente Screen1 coge el evento Screen1.Initialize y ponlo en tu programa.
19. Para colocar un sprite en un punto en concreto de la pantalla podemos usar sus coordenadas como hemos visto antes. Si seleccionas el componente cohete verás los bloques azules "set cohete.Y to" y "set cohete.X to" cógelos y ponlos dentro del evento .Initialize.
20. Queremos poner el cohete en el margen de abajo de la pantalla, pero tenemos el inconveniente de que nuestra aplicación puede funcionar en muchos tipos distintos de dispositivos (móviles, tablets, etc) y cada uno de ellos tendrá un tamaño de pantalla distinto, por lo tanto tenemos que buscar una manera de saber siempre cual es el tamaño de la pantalla en la que está funcionando nuestra aplicación. Para ello tenemos dos bloques dentro del componente Canvas que se llaman Canvas.Height (alto) y Canvas.Width (ancho) que nos dicen el alto y ancho

del lienzo en cada momento. Esto es importante también si el usuario cambia la orientación de la pantalla en algún momento puesto que cambiará también el tamaño del lienzo. En nuestro caso hemos desactivado la posibilidad de que el usuario lo haga, pero esta sería la forma de hacerlo.

21. Como queremos colocar nuestro cohete centrado en la pantalla, lo primero que tenemos que hacer es fijar su propiedad cohete.X a la mitad del ancho del lienzo (Canvas.Width / 2). Primero coge un bloque numérico de división "/" y pégalo al bloque "set cohete.X to".
22. Ahora coge un bloque Canvas.Width y ponlo como primer argumento de la división. El segundo argumento será un 2. De esta manera conseguimos centrar el cohete.
23. Como el ancho de la imagen del cohete (sprite) es 50. Su centro sería 25. Para centrar con precisión el cohete en la pantalla habría que restar al cálculo anterior estos 25 pixeles adicionales. Es decir el bloque "set cohete.X to" sería igual a ((canvas.Width / 2) - 25) así: No lo hagas ahora si no quieres, pero tenlo en cuenta en tus futuros proyectos.

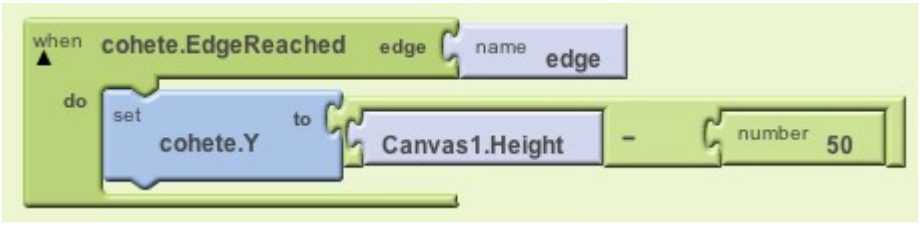


24. Una vez centrado en el eje horizontal ahora toca colocar el cohete en el margen inferior de la pantalla por lo que vamos a utilizar un truco parecido al anterior. Como sabemos el tamaño vertical del lienzo, porque lo tenemos en la propiedad Canvas.Height (Alto), basta con decirle al cohete que su coordenada Y Así lo colocamos siempre al final del lienzo, sea cual sea su tamaño.
25. Ya tenemos nuestro cohete colocado en la posición inicial, ahora sólo queda que cuando llegue al borde superior de la pantalla el cohete vuelva a su posición original. Para detectar si un sprite ha llegado al borde de la pantalla tenemos un evento llamado EdgeReached (Borde alcanzado) Este evento lo encontrarás dentro de cada sprite que uses en tu programa. En nuestro caso vamos a usar el del cohete. Selecciona el componente cohete y coge el evento cohete.EdgeReached.
26. Verás que aparece una propiedad (argumento) llamado edge (borde) que nos indica el borde al que ha llegado el cohete. En este caso no lo vamos a usar.
27. Ya sabemos que este evento se va a activar cuando nuestro cohete llegue al borde de la pantalla. Sólo nos queda que cuando esto ocurra volvamos a situar el cohete

en su posición inicial, tal y como hemos hecho anteriormente en el evento .Initialize. Coge un bloque "set cohete.Y to" y ponlo dentro de este evento.

28. Ahora tenemos que tener en cuenta un detalle importante. Al colocar de esta manera el cohete lo que ocurre es que lo estamos poniendo en el mismo borde. Por lo tanto el evento cohete.EdgeReached se va a activar de nuevo. Es decir, el cohete siempre estará tocando el borde, el evento se activará de forma continua y el cohete no se moverá porque entraremos en bucle infinito. Para que entiendas esto mejor, haz lo siguiente: Dentro del componente Canvas coge el bloque Canvas.Height y pégalo al bloque "set cohete.Y". ¿Ves qué ocurre? El cohete no se mueve, ¿verdad? ¿Comprendes lo que está ocurriendo? Vamos a repasarlo de nuevo.

29. El cohete está en el borde, por lo tanto el evento cohete.EdgeReached se activa. Dentro del evento hemos dicho que si llega al borde lo coloque de nuevo en la posición canvas.Height que es el tamaño máximo del lienzo y es, por lo tanto, el borde, con lo cual el evento se vuelve a activar. Al activarse volvemos a colocar el cohete en el borde y así indefinidamente. ¿Cómo lo solucionamos? Basta con que coloquemos el cohete un poco antes del borde, de esta manera el evento no se activará. Como el tamaño del sprite del cohete es 50x50, basta con que restemos el tamaño del cohete (50) del borde. Es decir "set cohete.Y to" = (canvas.Height - 50). Así:



    1. De esta forma estamos colocando el cohete 50 pixeles antes del borde inferior y el evento no se activará más en este caso. Sin embargo, al llegar al borde superior, el evento se vuelve a activar, poniendo el cohete en su posición correcta. Haciéndolo así tu cohete debería de despegar y volver a su posición original de forma continua.
    2. Aunque podríamos saber cual es el bloque contra el que está chocando (parámetro edge del evento .EdgeReached) así nos ahorramos un poco de trabajo.

30. Ahora vamos con la luna. Vuelve al Diseñador y selecciona el componente de la luna.

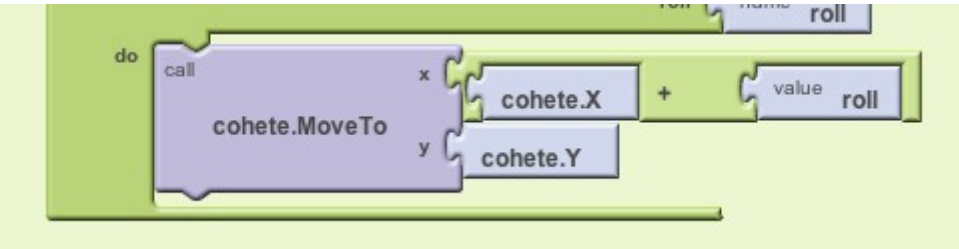
31. Vamos a cambiar sus propiedades. Para empezar cambia el nombre del componente y llámalo luna.

32. Ahora cambia el valor de la propiedad Heading y pon 0. Recuerda que un valor de 0 significa que queremos que la imagen se mueva desde la izquierda hacia la derecha.

33. Cambia el intervalo de tiempo (Interval) a 100 y para la imagen (Picture) selecciona la imagen luna.png aquí. Cambia la velocidad (speed) por 10 y el tamaño del sprite (imagen) por 50x50.
34. Ahora sitúa la imagen de la luna con el ratón arriba de la pantalla.
35. Vuelve al Editor de bloques.
36. Ahora la luna se mueve por la pantalla desde la izquierda hacia la derecha y se detiene al llegar al borde. Lo que vamos a aprender ahora es a hacer que una imagen rebote cuando llega a un borde. Para empezar necesitamos el mismo bloque evento que usamos antes para detectar cuando un sprite toca un borde. Selecciona el componente luna y coge el evento luna.EdgeReached.
37. Ahora el bloque que necesitamos para que un sprite rebote se llama "call luna.Bounce" (rebotar). Cógelo y ponlo dentro del evento luna.EdgeReached.
38. Recuerda que el evento .EdgeReached nos dice cual es el borde que ha tocado nuestra imagen a través del parámetro edge. Así que tenemos que usar ese parámetro también en el bloque .Bounce. Como siempre lo tienes dentro de My Blocks/My definitions pero ten cuidado y no te confundas porque tenemos dos parámetros edge y edge1. El primer lo usamos antes, ahora necesitamos el edge1. Pega este parámetro al bloque luna.Bounce y tu luna debería de rebotar de un lado a otro de la pantalla. ¿Funciona?
39. Ya tenemos dos sprites moviendose por la pantalla, ahora vamos a añadirle interactividad; es decir, la posibilidad de controlarlos de alguna manera. Como nuestros dispositivos tienen sensores de movimiento, vamos a controlar nuestro cohete inclinando nuestro móvil o tablet. Para esto hemos añadido el componente OrientationSensor antes. Selecciona este componente y coge el evento OrientationSensor.OrientationChanged.
    1. Este evento nos dice si hemos movido el dispositivo y en concreto nos dice si lo hemos inclinado en alguna dirección. Aunque lo veremos en detalle más adelante de momento nos interesa el parámetro roll que es el que nos indica la inclinación del dispositivo y la dirección. Si inclinamos hacia la izquierda el valor de roll será negativo y si lo inclinamos hacia la derecha el valor de roll será positivo. Cuanto más inclinamos en una dirección u otra el valor de rol será mayor o menor. Vamos a ver cómo usarlo.
40. Como queremos mover el cohete inclinando nuestro dispositivo lo primero que necesitamos es el bloque cohete.Move. Cógelo y ponlo dentro del evento OrientationChanged que acabamos de usar.
41. Ahora tenemos que decirle al cohete cuales son las coordenadas a las que queremos que se mueva. Recuerda que las coordenadas actuales del cohete, es decir, su posición en todo momento, la tenemos en las propiedades cohete.X y cohete.Y. Como el cohete se mueve por sí mismo de abajo hacia arriba, la coordenada Y no la vamos a tocar y vamos a dejar la que tenga en cada momento.

Busca la propiedad cohete.Y y ponla en el bloque .MoveTo.

42. Ahora la coordenada X, es decir, la posición horizontal del cohete es la que queremos manejar inclinando el dispositivo. Basta con que sumemos a la coordenada actual cohete.X el valor de roll. Pega al valor de x del bloque .MoveTo la suma de la propiedad cohete.X más roll. Así:



43. Ahora deberías de poder controlar tu cohete inclinando el dispositivo. ¿Funciona?

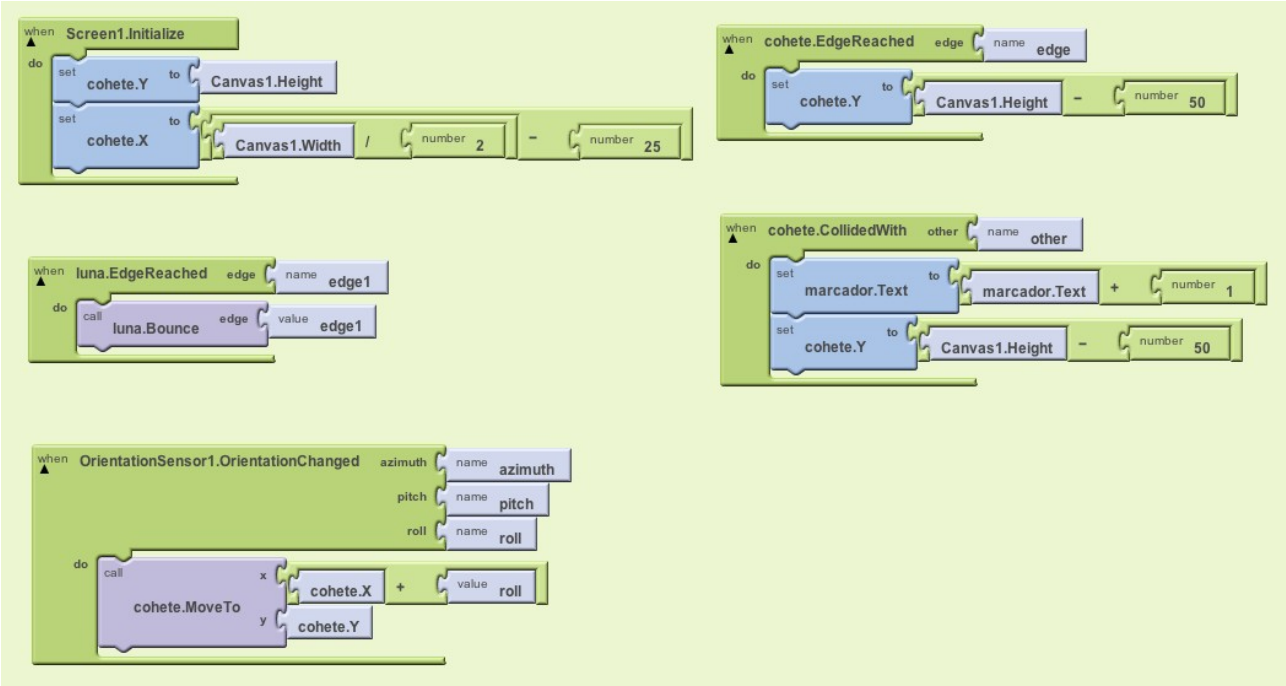
44. Por último queremos que cuando nuestro cohete toque la luna sumemos un punto a nuestro marcador. Es decir, cada vez que consigamos llevar el cohete a la luna ganamos un punto. Para detectar cuando un sprite toca a otro tenemos un evento llamado .CollideWith (colisiona con..) Para nuestro ejemplo podríamos usar el evento CollideWith tanto de la luna como del cohete puesto que el uno toca al otro indistintamente. Vamos a usar el evento del cohete. Busca el evento cohete.CollideWith y ponlo en tu programa.

45. Verás que el evento nos pasa un parámetro llamado other que nos dice contra qué otro sprite nos hemos tocado. Para este caso no vamos a usarlo puesto que ya sabemos que sólo podemos haber chocado contra la luna. Ahora dentro de este evento vamos a sumar un punto a nuestro marcador. Selecciona el componente marcador y coge el bloque "set marcador.Text to".

46. Como en el Diseñador habíamos puesto un valor de 0 a nuestro marcador, ahora sólo tenemos que sumarle un punto más. Coge un bloque suma y como primer parámetro usa el bloque marcador.Text porque es donde tenemos guardado el valor de nuestro marcador.

47. Como segundo valor de la suma pon un 1. Así vamos sumando un punto más cada vez que nuestro cohete llegue a la luna.

48. Finalmente tenemos que volver a poner el cohete en su posición original, de lo contrario nuestro cohete al tocar la luna la atravesaría sin más. Puedes duplicar el bloque "set cohete.Y to" que usamos dentro del evento cohete.EdgeReached para hacer esto. El programa final quedaría así:



Y ya por último si te atreves, incluye sonidos y pruébalo.

¿Y si hacemos un videojuego?

Conviértete en un *Caza mosquitos*

Vamos a hacer un juego muy chulo, nos vamos a convertir en caza mosquitos, esos bichos feos que te pican cuando menos te lo esperas y sobre todo te pican de noche incluso.

Como ya sabemos usar las variables, procedimientos y muchas cosas más ahora nos toca reunir todos esos conocimientos.

La idea será que tendremos un mosquito en pantalla que se irá moviendo cada cierto tiempo y el objetivo será tocar la pantalla de manera que si tocamos al mosquito lo mataremos, luego tendremos un número de intentos para matarlo y un tiempo de partida. Piensa que corre prisa matar al mosquito así que hazlo rápido por que no durará toda la vida la partida.

Pensemos como lo podemos hacer:

1. Interfaz de usuario.
    1. Primero crea un proyecto nuevo con el nombre que quieras.
    2. A continuación vamos con la puntuación, usa el componente HorizontalScreenArrangement para poner una etiqueta que será para la puntuación y la otra que será para los intentos de matar al mosquito.
    3. Debajo pon un canvas con un fondo por ejemplo de un cielo. Al canvas le añadiremos la imagen del mosquito y la imagen del mosquito muerto. Estas imágenes en principio deberán estar ocultas, así que ojo.
    4. Añade un Notifier para la ventana de puntuación, luego un reloj para mover al mosquito (se moverá cada 1 segundo, es decir un valor de 1000) y otro reloj para controlar el tiempo de la partida (haremos que dure 1minuto, que eso será un valor de 60000).
2. Moviendo el mosquito.
    1. Para asegurarnos de que el reloj reloj_mover_mosquito está activo al arrancar nuestro programa, vamos a activarlo nada más arrancar.
    2. Ahora vamos a por el evento del reloj que se activará cuando se cumpla el intervalo. "when reloj_mover_mosquito.Timer" y dentro vamos a mover el mosquito usando "imagen_mosquito.MoveTo".
    3. La coordenada X será un número aleatorio entre 1 y el ancho del canvas (canvas.Width) y la coordenada Y será otro número aleatorio entre 1 y el largo de canvas (canvas.Height). Con esto nuestro mosquito debería de moverse ya.

3. Cazándo al dichoso mosquito.
    1. Si tocamos con el dedo un sprite se activa el evento .TouchDown. Vamos a usar este evento para saber si el usuario ha tocado el mosquito.
    2. Cuando pulsamos el sprite del mosquito tendremos que parar el reloj que hace que se mueva el mosquito.
    3. Cuando toquemos al mosquito y se pare el reloj se tendrá que ocultar la imagen del mosquito, luego la imagen del mosquito muerto se tendrá que mover a donde está la imagen del mosquito que hemos matado y finalmente esta imagen de mosquito muerto se tendrá que visualizar.
4. Contando la puntuación.
    1. Vamos a llevar una puntuación sencilla, para ello lo que haremos será que el objetivo sea un valor cuanto más pequeño mejor.
    2. Si cuando pulsamos la pantalla no matamos al mosquito la puntuación se incrementará en 10 puntos, si transcurridos por ejemplo 3 intentos no hemos matado al mosquito la puntuación se incrementará en 20 y si tras ya 6 intentos no hemos matado al mosquito se irá incrementando la puntuación en 30 puntos.
    3. También por cada vez que pulsamos la pantalla el número de intentos se tendrá que incrementar en 1, para ello usaremos el bloque de “set global intentos = global intentos + 1”.
5. Fin de la partida
    1. Si matamos al mosquito en menos de 3 intentos el score será entre 0 y 30, si no será entre 0 y 90 y si no ya será entre 0 y 270puntos. Es decir que si lo intentamos matar y hemos fallado 6 veces o más significa que somos un poco malos para matarlo :)
    2. Cuando pase 1 minuto el juego terminará, así que los relojes se tendrán que parar tras este tiempo.
    3. Tras esto usaremos el bloque “Notifier1.ShowChooseDialog”. Como lo hemos añadido desde el Diseñador, lo encontrarás dentro de My Blocks.
    4. Configura el título con el parametro “title”, luego al parámetro “button1Text” le pasaremos el valor “Nueva partida” y al parámetro “button2Text” le pasaremos el valor de “Salir”. Al final al parámetro “cancelable” le pasamos el valor “false”.
    5. El evento Notifier1.AfterChoosing nos da un parámetro (choice) con la opción que ha elegido el usuario. Como los botones que hemos usado antes para esta ventana son "Salir" o "Nueva partida", este parámetro será uno de los dos. Por lo tanto únicamente tenemos que comprobar uno de ellos para saber cuál ha sido el que ha elegido. Coge un bloque de control ifelse y ponlo dentro del

evento Notifier1.AfterChoosing. Vamos a comprobar si el usuario quiere salir de la aplicación, de lo contrario, ya sabemos que quiere jugar otra vez.

6. Al parámetro test le vamos a pasar una expresión lógica (value choice = "Salir") Recuerda que el parámetro que nos pasa este evento lo tienes dentro de My Blocks/My definitions.
7. Si la respuesta es verdadera (el usuario quiere salir) coge un bloque "call close application" que encontrarás dentro de Built-in/Control. Este bloque termina la aplicación pero únicamente funcionará si instalas la aplicación directamente en tu dispositivo. Mientras estés conectado a través del Editor de bloques no es posible cerrar la aplicación.
8. Si la respuesta es negativa (el usuario quiere volver a jugar) lo primero que vamos a hacer es volver a poner los contadores a cero. "set global intentos = 0" y "set global puntos=0" y después cambia las etiquetas. Puedes duplicar los bloques que has usado antes "set etiqueta_intentos.Text" y "set etiqueta_puntuación.Text".
9. Ahora volvemos a activar los dos relojes "set reloj_mover_mosquito.TimerEnabled=true" y "set reloj_final_partida.TimerEnabled=true".