Trabajando con Canvas

¿ Qué aprenderé ?....2
Empezamos....3
Diseñar los componentes....4
Crear los botones de color....4
Ordenar el contenido de la pantalla....5
Agregar el lienzo....6
Obtener los botones y trabajar con el componente Camera....7
Asignar comportamientos a los componentes....8
Evento para dibujar un punto al tocar la pantalla....8
PRUEBA!!!!....11
Primera Modificación....11
Añadir el evento encargado de dibujar una línea....11
PRUEBA!!!!....14
Segunda Modificación....14
Añadir los controladores de los botones....14
Permitir que el usuario haga una fotografía....15
PRUEBA!!!!....16
Tercera Modificación....16
Cambiar el tamaño del punto....16
Utilizar variables....17
Cambiar el valor de las variables....18
PRUEBA!!!!....19
Variaciones....22
Resumen....22

En este capítulo presentaremos el componente *Canvas*, que se utiliza para generar gráficos sencillos de dos dimensiones (2D). Vamos a construir Bote de Pintura una aplicación para que el usuario pinte con diferentes colores en la pantalla de su dispositivo móvil empleando los dedos. Luego la actualizaremos para que permita abrir una fotografía y pintar sobre ella. Bote de Pintura fue uno de los primeros programas que se desarrollaron para demostrar la capacidad de los PC's

sobre la década de los 70.

Con la aplicación Bote de Pintura podremos:

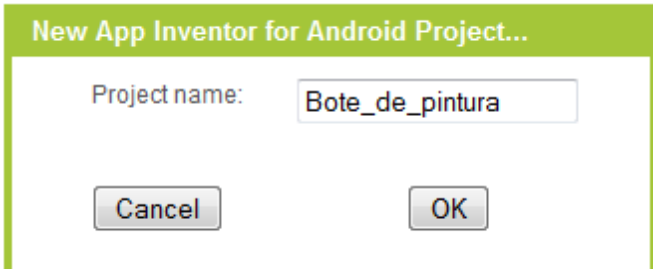
- Sumergir nuestro dedo en un tintero virtual y pintar con ese color.
- Dibujar una línea deslizando el dedo por la pantalla.
- Establecer puntos tocando la pantalla.
- Utilizar un botón para limpiar el lienzo.
- Modificar el tamaño del trazo mediante otros botones.
- Hacer una fotografía con la cámara de nuestro teléfono y pintar sobre ella.

¿ Qué aprenderé ?

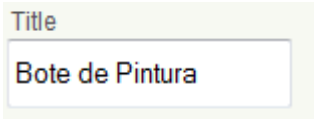
- Uso del componente **Canvas** para dibujar.
- Trabajar con **eventos** encargados de controlar si se toca o si se desliza algún elemento sobre la superficie del teléfono.
- Distribuir los componentes de la aplicación por la pantalla.
- Utilizar **controladores de eventos** para trabajar con **argumentos**.
- Definir **valores** para recordar la información, como el tamaño del trazo que ha seleccionado el usuario.

Empezamos

Inicia un nuevo proyecto desde la opción **Projects-->Start new project** y llámalo **Bote_de_Pintura.**



Abre el **Blocks.** Para empezar ves a **Properties** y modifica el título de la pantalla para que muestre Bote de Pintura. El teléfono deberá enseñar el nuevo texto en la barra de título de la aplicación.



Para saber si estamos cambiando el nombre de un proyecto o el de un elemento de la pantalla, deberemos recordar que hay tres nombres claves en **AppInventor**:

- El nombre que asignemos al proyecto con el que estamos trabajando. Ésta será la denominación que tendrá la aplicación cuando empaquetemos para que funcione en los teléfonos. Podemos utilizar el botón Save As (Guardar como) para que **Component Designer** abra una nueva versión del proyecto. También podemos guardarla con otro nombre.

- El nombre del componente Screen1, ubicado en el panel **Components** (Componentes), que lista todos los elementos de la aplicación. Esta denominación no se puede modificar en la versión actual de **AppInventor**.

- El título de la pantalla, que aparece en la barra de título del teléfono. Su valor inicial será Screen1. Este es el que hemos cambiado.

Diseñar los componentes

Utilizaremos:

- Tres componentes **Button** para seleccionar la pintura roja, azul o verde y uno **HorizontalArrangement** para organizarlos.

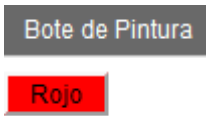
- Un componente **Button** para limpiar el lienzo y otros dos para cambiar el tamaño del trazo.

- Un componente Canvas que será el lienzo de dibujo. **Canvas** tiene una propiedad

- **BackgroundImage** a la que asignaremos el archivo *"Kitty.png"*. Más tarde, aprenderemos a modificar la aplicación para que pueda experimentar con las fotografías que tome el usuario con la cámara de su teléfono.

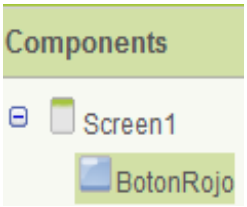
Crear los botones de color

Empezaremos por generar tres botones de color:



1. Arrastre el componente **Button** al visor y cambie su atributo **Text** a Rojo. Luego, modifica su color de fondo para que sea rojo (propiedad **BackgroundColor**).

2. En la sección **Components** (entre Viewer y**Properties**) seleccione el elemento **Button**1 de la lista y, a continuación, haga clic sobre el botón Rename(Renombrar). Cambie su nombre a **BotonRojo**.



Fíjate que no puedes utilizar espacios con las denominaciones de los componentes. Por eso se pone en mayúscula la primera letra de cada palabra.

3. Crea dos botones más para los colores azul y verde de la misma forma que acabas de hacer. Les llamarás **BotonVerde** y **BotonAzul**. Colócalos debajo del botón rojo. Cambia sus propiedades **Text** por Verde y Azul, respectivamente.

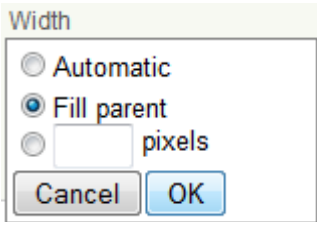
Fíjate que en este proyecto hemos modificado el nombre de los componentes en vez de usar el predeterminado.

Ordenar el contenido de la pantalla

Ahora los alinearemos en la parte superior de la pantalla. Para ello utilizaremos el componente **HorizontalArrangement**:

1. Desde la categoría **Layout** (Distribución de la pantalla) de la Paleta, arrastre el componente **HorizontalArrangement** hasta colocarlo debajo de los tres botones.

2. En el panel**Properties**, modifique el valor Width (ancho) del componente **HorizontalArrangement** a Fill parent (Rellenar el elemento padre). Así, este separador horizontal ocupará todo el ancho de la pantalla.



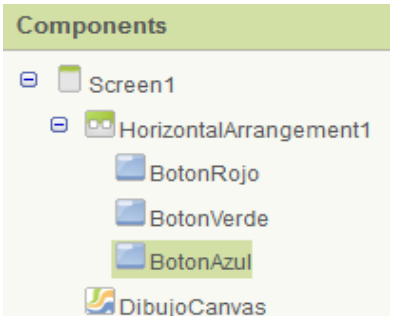
3. Mueve los tres botones hasta ponerlos dentro del componente **HorizontalArrangement**. Una pista: verás una línea vertical azul que te indicará dónde se sitúa el elemento que está arrastrando.

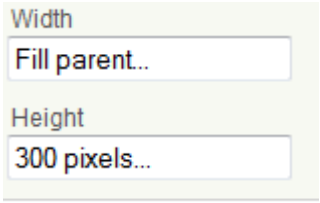
Si observas ahora la lista de los componentes del proyecto, comprobará que los tres botones están dentro de **HorizontalArrangement**. **AppInventor** nos indica que se han convertido en tres subcomponentes. Fíjate que todos ellos se encuentran debajo de **Screen1.**

Agregar el lienzo

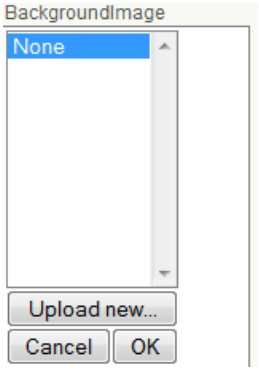
El lienzo será la superficie donde dibujaremos los círculos y las líneas. Vamos a agregarlo y a añadir el archivo *"kitty.png"*, que utilizaremos como fondo (**BackgroundImage**):de **DibujoCanvas**

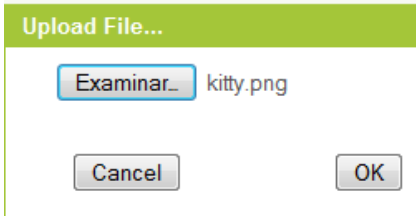
1. Desde la categoría Basic de la sección Palette, arrastra el componente **Canvas** al visor. Cambia su nombre a **DibujoCanvas**. Modifica el valor de su anchura (propiedad width) a *Fill parent* y asígnale una altura de 300 píxels.





2. Usa el archivo *"Kitty.png"* como fondo del lienzo (es decir, asígnalo a la propiedad **BackgroundImage** del componente Canvas). En la sección **Properties** verás que el valor de **BackgroundImage** es None (ninguno). Haz clic sobre ese campo y utiliza el botón **Add** (Agregar) para seleccionar el archivo *"Kitty.png."*





3. Ahora asigna el color rojo a la propiedad **PaintColor** de Canvas. Así, cuando un usuario abra la aplicación y aún no haya elegido ningún color, empezará dibujando con el rojo.

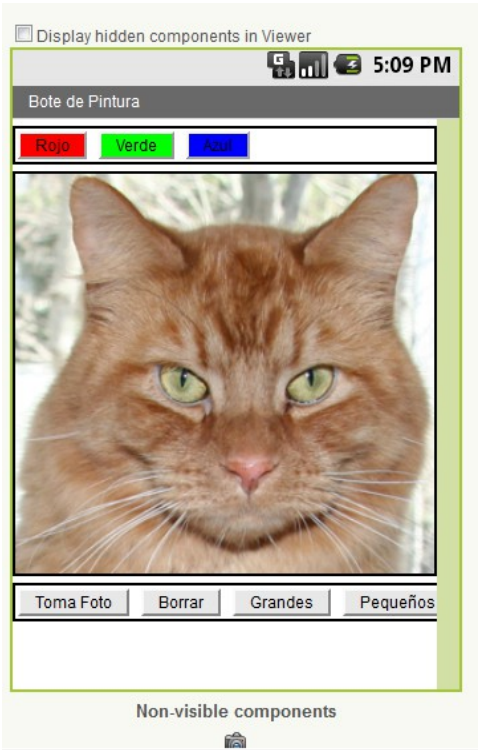
Obtener los botones y trabajar con el componente Camera.

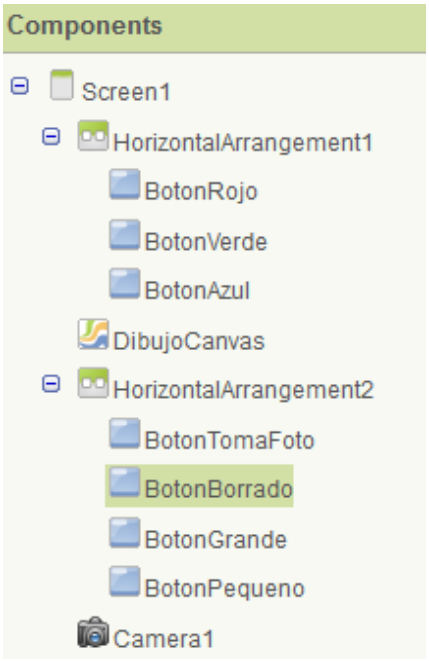
1. Desde **Layout,** arrastra un nuevo complemento **HorizontalArrangement** hasta debajo del lienzo. Configura su priopiedad Width a Fillparent.A continuación, desplaza dos componentes **Button** hasta la pantalla y colócalos dentro del elemento **HorizontalArrangement** que acabas de poner en la parte inferior del visor. Cambia el nombre del primer botón a **BotonTomaFoto.** Altera el valor de su propiedad **Text** por *Toma Foto*. Modifica el nombre del segundo botón a **BotonBorrado** y asigna el valor *Borrar* a su propiedad **Text.**

2. Arrastra dos componentes **Button** más a la distribución horizontal que se encuentra en la parte inferior del visor y colócalos junto al **BotonBorrado**.

3. Cambia el nombre de estos botones por **BotonGrande** y **BotonPequeño** y asígnales los valores "*Grandes"* y "*Pequeños"* a sus respectivos campos **Text.**

4. Desde la **Paleta Media**, arrastra el componente **Camera** hasta el visor. Aparecerá en el área reservada a los componentes invisibles.





Asignar comportamientos a los componentes

Utilizaremos tres bloques para controlar los movimientos que el usuario hará sobre el visor del teléfono (tocar la pantalla y deslizar el dedo por ella), para dibujar y hacer fotografías.

El tipo, **DibujoCanvas** cuenta con los eventos ***Touched*** y ***Dragged***. programaremos el evento **DibujoCanvas**.**Touched** para que llame a **DibujoCanvas**.**DrawCircle**. También estableceremos el evento **DibujoCanvas**.Dragged para que invoque a **DibujoCanvas**.**DrawLine**. Luego, actuaremos sobre los botones para definir el color (con la propiedad **DibujoCanvas**.PaintColor), borrar el contenido del lienzo (con la propiedad **DibujoCanvas**) y cambiar la imagen (**BackgroundImage**) de fondo por la fotografía que hayamos tomado con la cámara.

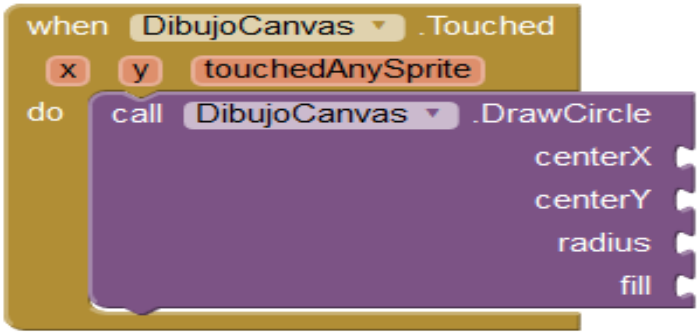
Evento para dibujar un punto al tocar la pantalla

Lo primero que haremos será prepararlo todo para que, cuando el usuario toque la pantalla (el lienzo de dibujo, es decir, **DibujoCanvas**), la aplicación dibuje un punto:

1. Desde **Blocks**, haz clic en **DibujoCanvas** y arrastra el bloque **DibujoCanvas**.**Touched** hasta el área de trabajo. Cuando lo sueltes, Blocks agregará automáticamente tres cajas azules: **X, Y** y **TouchedAnySprite.**

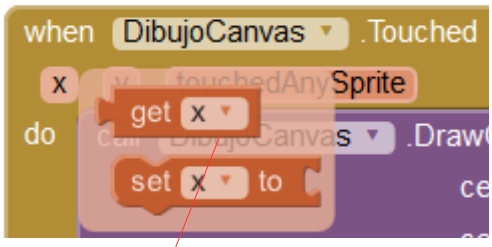
Nota: El evento **DibujoCanvas**.**Touched** indicará las coordenadas (X,Y) del punto donde el usuario ha tocado la pantalla. También señalará si se ha tocado un objeto que se encuentra dentro del lienzo, es decir, dentro de **DibujoCanvas** (en **AppInventor** se llama sprite), pero no lo veremos hasta el próximo capítulo. Las coordenadas X e Y son los argumentos que utilizaremos para saber el punto exacto donde tocó la pantalla el usuario. Así podremos dibujar un punto en esa posición.

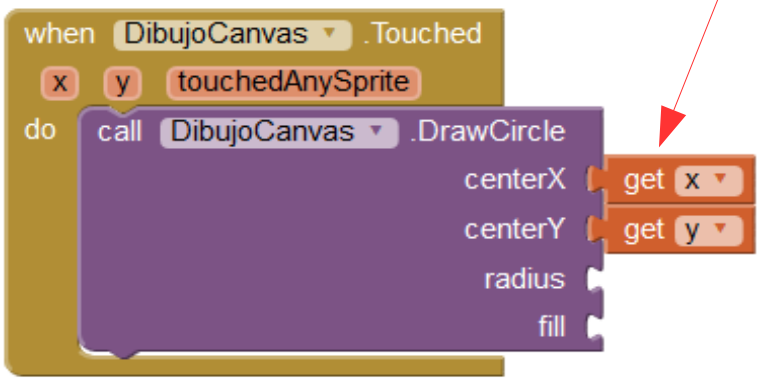
2. Haz clic sobre **DibujoCanvas** y arrastra el comando **DibujoCanvas**.**DrawCircle** hasta el área de trabajo. Suéltalo dentro del controlador de eventos **DibujoCanvas**.**Touched**, como se ve en la figura:



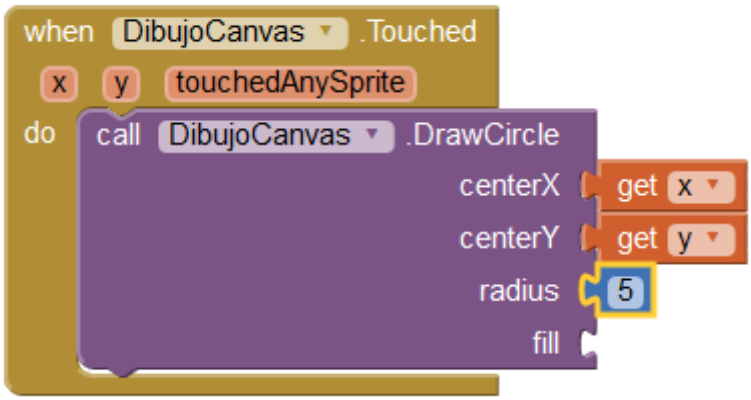
En el lado derecho de **DibujoCanvas**.**DrawCircle** hay tres bloques pequeños reservados para los argumentos X,Y,R. Los dos primeros determinan la ubicación donde se deberá dibujar el círculo; el último establece el radio de dicho círculo. Este controlador puede llegar a ser algo confuso porque el evento **DibujoCanvas**.**Touched** también tiene dos bloques llamados X e Y. Bastará con que recuerdes que los valores X e Y que utiliza el evento **DibujoCanvas**.**Touched** nos dirán el punto exacto de la pantalla que ha tocado el usuario, mientras que las coordenadas X e Y del evento **DibujoCanvas**.**DrawCircle** indican dónde se dibujará el círculo. Como esta aplicación lo trazará en el mismo punto que ha tocado el usuario, usaremos los mismos valores X e Y de **DibujoCanvas**.**Touched** como parámetros X e Y de **DibujoCanvas**.**DrawCircle**.

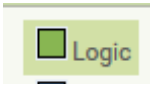
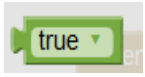
3. Ponte sobre el valor de X del evento y aparecerá una nueva ventana. Son las referencias a los argumentos X e Y del evento. Desplaza los valores X e Y hasta las ranuras correspondientes del bloque **DibujoCanvas**.**DrawCircle**.

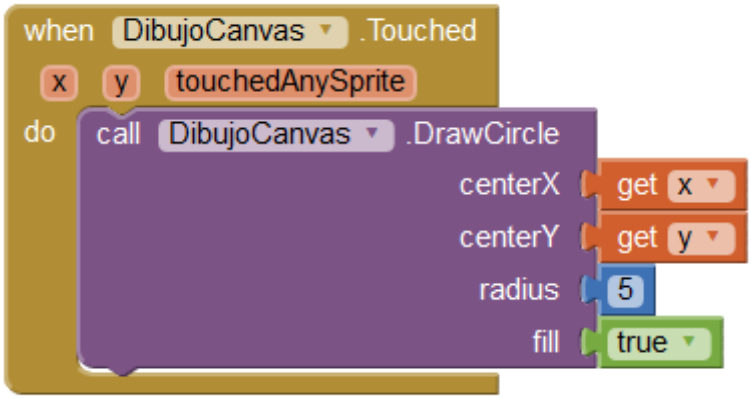




4. También tendremos que determinar el radio (r) del círculo que trazará la aplicación. Como unidad de medida usaremos el píxel, que es el punto más pequeño que se puede dibujar en pantalla. Le asignaremos el valor 5. Pulsa sobre el botón izquierdo **Math** que aparece en el lado izquierdo de **Blocks** y, de la lista desplegable, selecciona el elemento  para crear un bloque numérico. Cambia su valor (a 5) y arrástralo hasta la ranura r. La figura muestra el aspecto del controlador de eventos **DibujoCanvas**.**Touched**:



Por último hemos de rellenar el bloque fill. Se trata de un valor lógico con dos posibles valores True o False, si queremos que el circulo este relleno o no. En este caso elegiremos true. Ponte sobre  y arrastra  al campo fill.



PRUEBA!!!!

Vamos a comprobar cómo funciona la aplicación en la pantalla del teléfono. Cuando la toque, deberá dibujar un pequeño círculo. Como asignamos el valor Red a la propiedad

Canvas.PaintColor, aparecerá en color rojo. Si no lo hubiésemos establecido así, los dibujaría de color negro, que es el valor predeterminado o por defecto. Por si acaso guarda el proyecto en tu ordenador con el nombre **BotePintura**, se generará el fichero **"BotePintura.apk"** que es el que debes copiar a tu teléfono y probarlo si no te funciona correctamente ahora.

Primera Modificación

Añadir el evento encargado de dibujar una línea

Ahora vamos a agregar el controlador de eventos que comprobará si el usuario desliza el dedo por la pantalla. Ya que no es lo mismo tocarla que deslizar el dedo por ella:

- Tocamos la pantalla cuando colocamos el dedo en un punto del lienzo y lo levantamos sin más, sin llegar a moverlo por la superficie de la misma.

- Deslizamos el dedo cuando lo ponemos sobre la pantalla y lo movemos sin levantarlo.

En un programa de dibujo, el hecho de deslizar el dedo por la pantalla equivale a utilizar un lápiz con un papel. Al no separarlo de la superficie, dibujamos una línea.

Algo parecido ocurrirá con la aplicación de dibujo porque, en realidad, lo que hacemos aquí es trazar cientos de pequeñísimas líneas rectas para crear el efecto de una línea curva. Cada vez que deslizamos el dedo por la pantalla, hacemos una pequeña línea desde la última posición que tenía el dedo hasta la nueva.

Desde el **Blocks.**

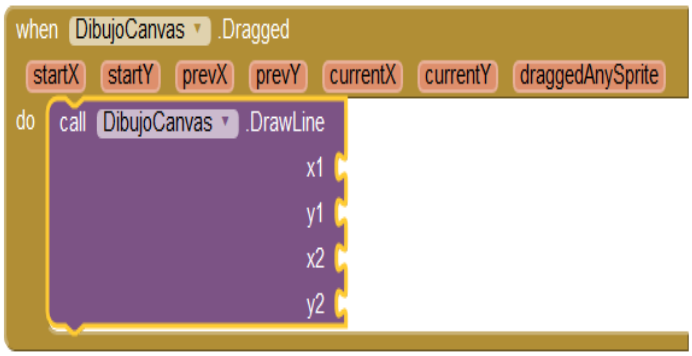
1. Desde **DibujoCanvas** arrastra el bloque **DibujoCanvas**.Dragged hasta el área de trabajo.

El evento **DibujoCanvas**.Dragged tiene siete argumentos.

- *StartX, StartY*: Indican la posición donde se encontraba el dedo en el momento de iniciar su movimiento por la pantalla.
- *CurrentX, CurrentY*: Señalan la posición actual del dedo.

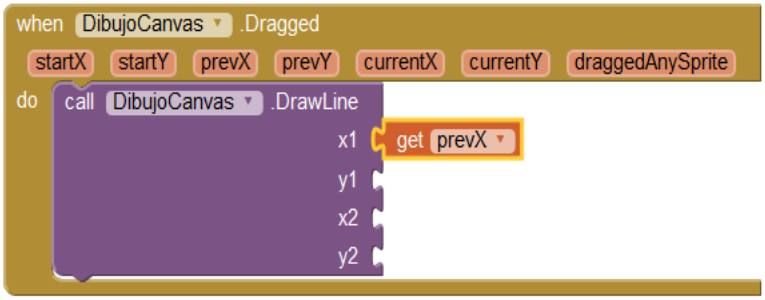
- *PrevX, PrevY*: Determinan la última posición del dedo.
- *DraggedSprite*: Este argumento tendrá el valor true (verdadero) cuando el usuario deslice el dedo sobre la imagen de un **sprite**. No lo utilizaremos en este capítulo.

2. Arrastra el bloque **DibujoCanvas**.**DrawLine** desde **DibujoCanvas** hasta el área de trabajo. Suéltalo dentro del bloque **DibujoCanvas**.**Dragged,** como en la figura:

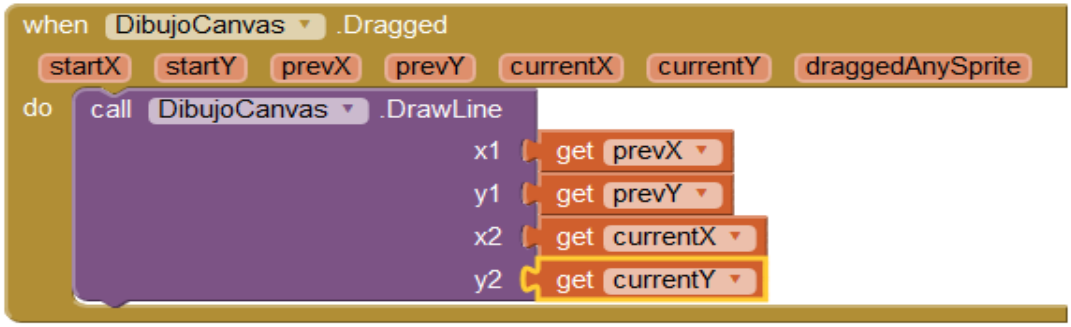


El bloque **DibujoCanvas**.DrawLine tiene cuatro argumentos, dos para cada punto de la línea: (*x1,y1*) es un punto y (*x2,y2*) es otro. ¿Imaginas qué valores deberemos asignar a cada argumento? Recuerda que la aplicación llamará al evento **Dragged** varias veces mientras deslizamos el dedo sobre la pantalla del teléfono (sobre el lienzo). Cada vez que lo movamos, el programa dibujará una pequeña línea que irá desde el punto (*PrevX,PrevY*) hasta el punto (*CurrentX,CurrentY*). Vamos a añadirlo al bloque **DibujoCanvas**.**DrawLine.**

3. Ponte sobre el parámetro **prevX,** aparecerá esta ventana. Arrastra **getprevX** al parametro **X1** de **Drawline**, obtendrás:



Procede de igual forma con el resto de parámetros hasta obtener.



Antes de seguir prueba el funcionamiento de la aplicación. Por si acaso guarda el proyecto en tu ordenador con el nombre **BotePintura2**, se generará el fichero**" BotePintura2.apk"** que es el que debes copiar a tu teléfono y probarlo si no te funciona correctamente ahora. Desliza el dedo por la pantalla y comprueba que se dibujan líneas y curvas. Toca la pantalla para dibujar puntos.

PRUEBA!!!!

Segunda Modificación

Añadir los controladores de los botones

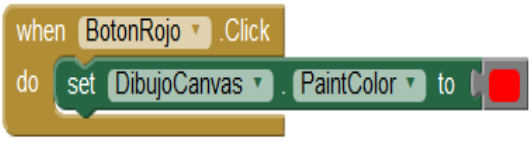
La aplicación que estamos creando permite dibujar con el dedo pero siempre usa el mismo color: **Rojo**. Ahora, vamos a añadir unos controladores a los que asignaremos botones. Con ellos el usuario cambiará el tono del trazo. También emplearemos el controlador **BotonBorrado** para que el usuario borre el contenido de la pantalla y empiece a dibujar de nuevo en el lienzo.

Desde **Blocks:**

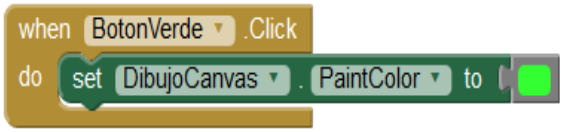
1. Abre el cajón de **BotonRojo** y arrastra el bloque **BotonRojo**.**Click** hasta el área de trabajo.

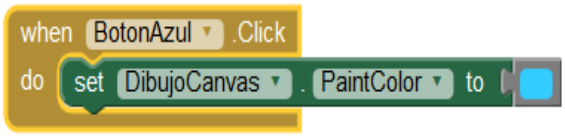
2. Abre el cajón **DibujoCanvas** y arrastra el bloque **Set DibujoCanvas**.**PaintColor to** hasta la sección do de **BotonRojo**.Click.

3. Haz clic sobre la pestaña **Built-In**. Abre el cajón **Colors** y arrastra el bloque del color Rojo hasta colocarlo en la ranura del bloque to de **DibujoCanvas**.**PaintColor.**



4. Repite los pasos del 2 al 4 para añadir botones para el color Azul y Verde.





5. El último botón que configuraremos es **BotonBorrado** (borrará el contenido del lienzo). Abre el cajón **BotonBorrado** y arrastra el elemento **BotonBorrado.Click** hasta el área de trabajo. Abre ahora **DibujoCanvas**, arrastra el elemento **DibujoCanvas**.**Clear** y suéltalo dentro del bloque **BotonBorrado.Click.**

Permitir que el usuario haga una fotografía

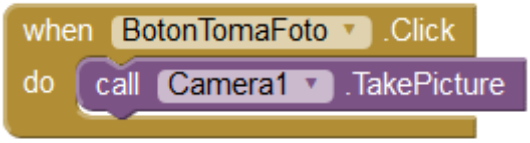
Las aplicaciones de **AppInventor** pueden interactuar con las funcionalidades del dispositivo Android, incluyendo la cámara de fotos. Para mejorar el programa, vamos a hacer que el usuario pueda sacar una fotografía con la cámara de su teléfono y utilizarla como fondo del lienzo.

1. El componente **Camera** tiene dos bloques importantes: **Camera.TakePicture**, que abre la aplicación de la cámara del dispositivo, y **Camera.AfterPicture,** que se inicia cuando el usuario ha hecho la fotografía.

   Añadiremos los bloques al controlador **Camera.AfterPicture** para lograr que **DibujoCanvas**.**BackgroundImage** use la fotografía que acaba de hacer el usuario.

Abre el cajón **BotonTomaFoto.** Arrastra el controlador de eventos **BotonTomaFoto.Click** hasta el área de trabajo.

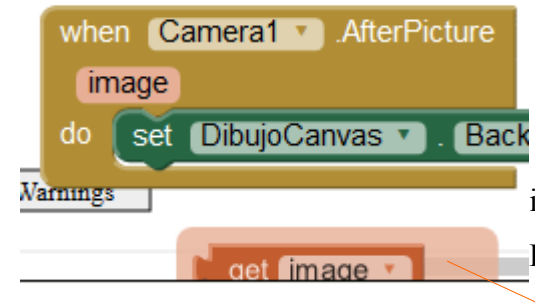
2. Ahora, abre el cajón Camera1 y arrastra el elemento **Camera1.TakePicture** hasta el controlador **BotonTomaFoto.Click**.



3. Abre el cajón Camera1 y arrastra el controlador **Camera1.AfterPicture** hasta el área de trabajo.

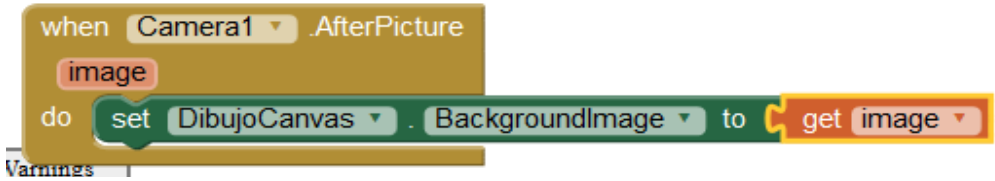
4. Arrastra el elemento **Set DibujoCanvas**.**BackgroundImage to**, que se encuentra dentro del cajón **DibujoCanvas** hasta el controlador **Camera1.AfterPicture.**

5. **Camera1.AfterPicture** tiene un argumento llamado *image*, que es la foto que acaba de tomar el usuario. Situate sobre el parametro *image,* aparecerá esta ventana.



Arrastra el valor **get image** hasta el hueco de la instrucción **Set DibujoCanvas.Background Image to**

obteniendo



Antes de seguir prueba el funcionamiento de la aplicación. Por si acaso guarda el proyecto en tu ordenador con el nombre **BotePintura3**, se generará el fichero **"BotePintura3.apk"** que es el que debes copiar a tu teléfono y probarlo si no te funciona correctamente ahora. Pincha sobre el botón **Toma Foto** (Hacer foto) para hacer una fotografía. Observa que se cambia la imagen del gato por la fotografía que acabas de hacer. Ahora, prueba a dibujar una línea sobre ella.

PRUEBA!!!!

Tercera Modificación

Cambiar el tamaño del punto

El tamaño del trazo está definido por puntos, que se establecen desde **DibujoCanvas**. **DrawCircle**. En la aplicación asignaremos el valor **5** al argumento **R**(radio). Para variar el grosor del trazo, deberemos variar este valor. Prueba a modificar el valor del radio a **10**.

Guarda el proyecto como **Bote_pintura3_punto10** y guárdalo en tu PC como hasta ahora. Ves al teléfono y dibuja. Comprobarás que el trazo ha alterado sus dimensiones.

El problema que tenemos aquí es que sólo podemos emplear el rango que le asignemos al argumento **R**. Pero, vamos a modificar el programa para que el usuario pueda cambiar el tamaño del trazo sin la ayuda del programador. No olvides que el trazo era una sucesión de puntos.

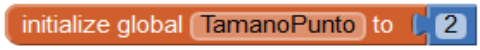
Vamos a hacer que, cuando el usuario toque el botón **BotonGrande** (Puntos grandes), la aplicación varíe el tamaño del punto a 8 píxels. Además, en el momento en que pulse el botón **BotonPequeño** (Puntos pequeños), se le asigne un tamaño de 2 píxels.

Para utilizar distintos valores con el argumento R (radio), la aplicación deberá saber cúal queremos emplear. Tenemos que especificar un valor y guardarlo en algún sitio para que el programa lo recuerde. Cuando hay que acordarse de algo no emplearemos propiedades, sino variables. Una variable es una célula de memoria. Podemos verlo como una caja donde guardamos cosas. Su contenido puede variar (como el tamaño del trazo). Trataremos las variables en capítulos

posteriores. Vamos a empezar por definir una llamada **TamanoPunto:**

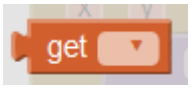
1. Desde **BlockEditor**, en **Built-In,** colocate sobre **variables.** Arrastra hasta el área de trabajo un bloque 

   Sustituye el texto **name** por **TamanoPunto.**

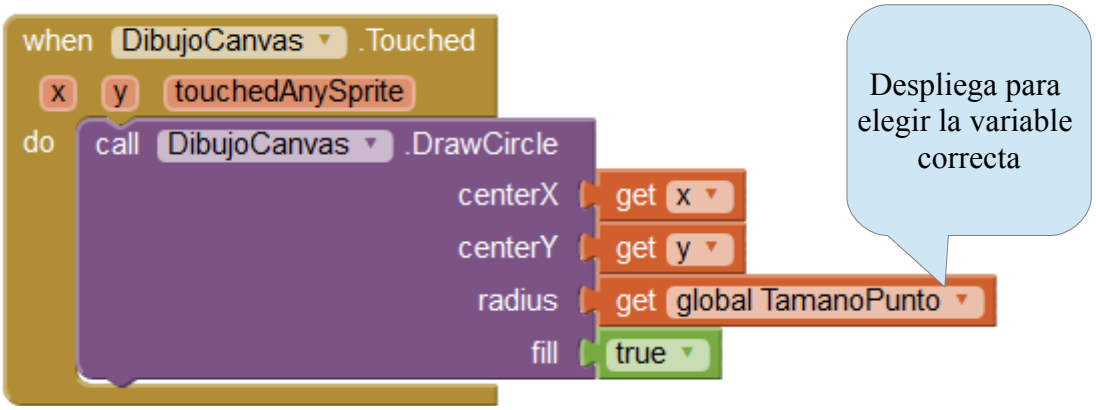
2. Fíjate en el lado derecho del bloque de la variable: tiene una ranura. Es para especificar el valor inicial de la variable o, lo que es lo mismo, determinaremos el rango que tendrá cuando se abra la aplicación. En programación, hablaremos de inicializar una variable. En nuestro programa, asignaremos el valor 2 a **TamanoPunto**. Recuerda que debes ir a **Math** para hacer esto. 

Utilizar variables

Ahora vamos a cambiar el argumento de **DibujoCanvas**.**DrawCircle** que se halla en el controlador **DibujoCanvas**.**Touched** para que utilice el valor de la variable **TamanoPunto**. Así, evitaremos que trabaje siempre con un número fijo. Inicialmente **TamanoPunto** tendrá el valor 2 pero cambiaremos el valor de **TamanoPunto** para que modifique el tamaño del trazo.

1. Pulsa sobre el controlador de eventos **DibujoCanvas**.**Touched** y arrastra el bloque **numerico 10** hasta la papelera que se encuentra en el área de trabajo. Sustitúyelo por 

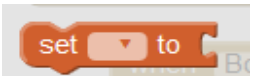
ubicado en el cajón **variables**.



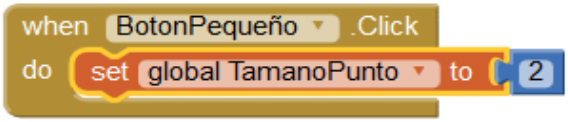
Cuando el usuario toque el lienzo, la aplicación tomará el valor de la variable **TamanoPunto** para establecer el tamaño del trazo (el radio del punto).

Cambiar el valor de las variables

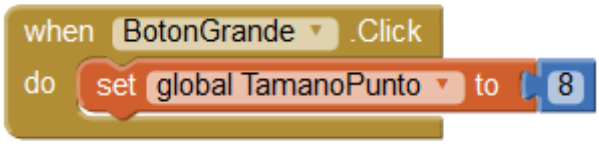
Vamos a ver ahora uno de los puntos más interesantes relacionado con el uso de las variables. Observaremos que la variable **TamanoPunto** permite que el usuario escoja el tamaño del trazo y que el controlador encargado de pintar la pantalla empleará este nuevo rango. Para implementar este comportamiento programaremos dos controladores de eventos: **BotonPequeno.Click** y **BotonGrande**.**Click**

1. Abre el cajón **BotonPequeno** y arrastra hasta el área de trabajo el controlador de eventos **BotonPequeno.Click**. A continuación, pulsa en **variables** y desplaza 

hasta **BotonPequeno.Click.** Por último, crea un bloque numerico 2 y conéctalo al **Set TamanoPunto to**



2. Ahora, crea un controlador de eventos similar para **BotonGrande**.**Click** pero asigna el valor 8 a **TamanoPunto**.



Nota: La palabra **"global"** del bloque **set global TamanoPunto to** indica que la variable la pueden utilizar todos los controladores de eventos del programa (es decir, que se puede emplear de forma global) Algunos lenguajes de programación permiten definir variables que son locales y que sólo se puede recurrir a ellas en una parte concreta del programa. **AppInventor** sólo trabaja con variables globales.

Antes de seguir prueba el funcionamiento de la aplicación. Guarda el proyecto en tu ordenador con el nombre **BotePintura4**, se generará el fichero **"BotePintura4.apk"** que es el que debes copiar a tu teléfono y probarlo si no te funciona correctamente ahora. Haz clic sobre los

botones que controlan el tamaño del trazo y prueba a dibujar en el lienzo. ¿Aparecen los círculos con diferentes tamaños? ¿Y las líneas?

PRUEBA!!!!

Las dimensiones de la línea no deberían cambiar porque sólo hemos programado el bloque **DibujoCanvas**.**DrawCircle** para que use la variable **TamanoPunto**. Basándose en esto, ¿Sabrías modificar los bloques para que los usuarios también pudiesen alterar el tamaño de las líneas?

Recuerda que Canvas tiene una propiedad llamada **LineWidth**. Cuando lo logres guarda el proyecto como **BotePintura5.** La solución esta abajo, mírala después de haberlo intentado, en caso de no haberlo logrado.

La solución consiste en asignar a la propiedad **LineWidth** del objeto **DibujoCanvas** el valor de la variable **TamanoPunto**. Esto lo haremos para el **BotonPequeño** y para **BotonGrande**.

Variaciones

Te propongo unas cuantas variaciones para que las explores :

- La interfaz del usuario no proporciona demasiada información sobre la configuración de la aplicación. Por ejemplo, la única forma de saber el tamaño del punto es dibujando algo. modifica el programa para que la configuración aparezca en la pantalla.

- Permite que el usuario determine el tamaño del punto que quiera utilizar. Deberás hacerlo con un componente **TextBox**. De esta manera, podrás recurrir a cualquier valor y no ceñirte sólo a *2 y 8 píxeles*.

Resumen

Este capítulo te ha enseñado cómo se puede utilizar el componente **Canvas** para crear un programa de dibujo sobre un lienzo, detectar cuándo el usuario toca la pantalla y desliza su dedo por ella, programar controladores de eventos relacionados y utilizar variables. También se puede emplear para programar animaciones como las que aparecen en los juegos 2D. Ampliaremos más este tema en otros capítulos.