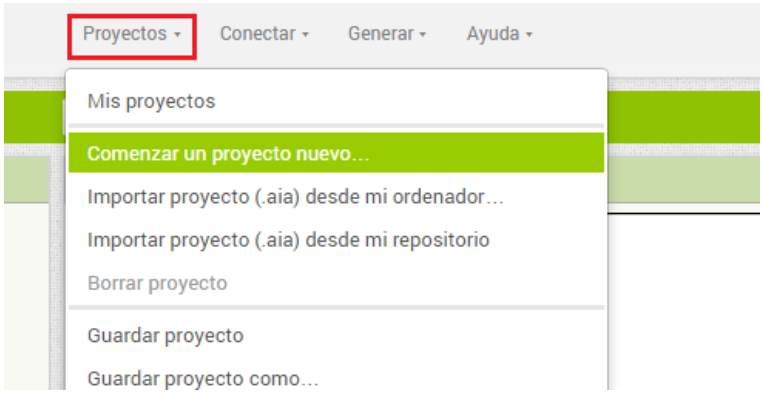
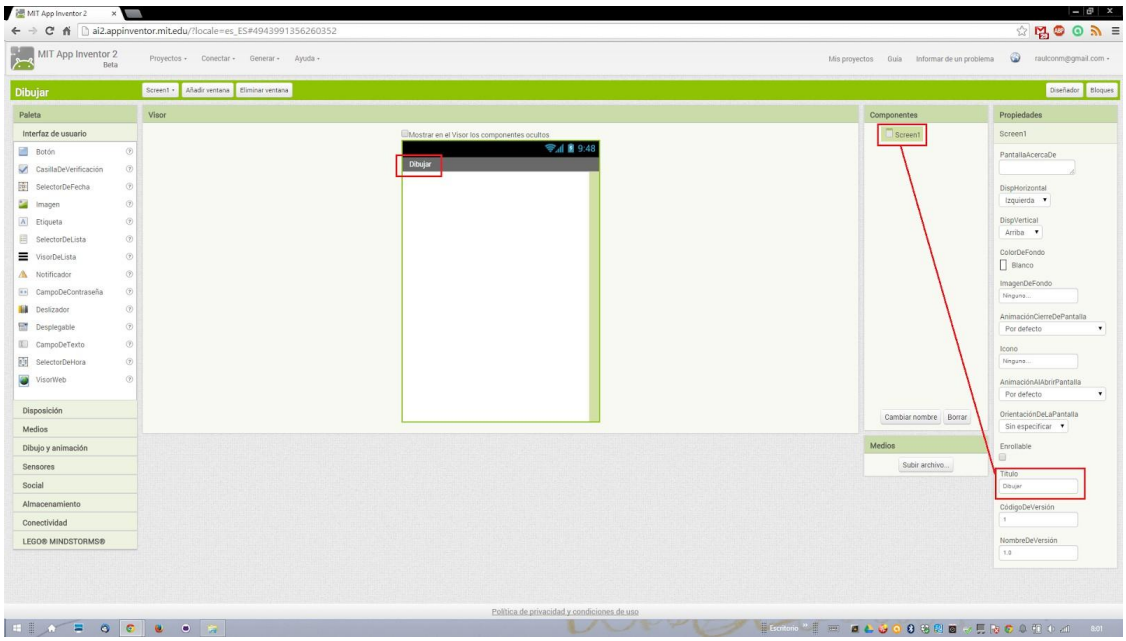
PRACTICA Crear una aplicación para dibujar

Una vez que hemos visto el manejo más básico de las pantallas, el Diseñador, y el Editor de Bloques de App Inventor, ya estamos listos para hacer algo más avanzado. Comenzaremos haciendo clic en el desplegable **Proyectos**, y eligiendo la opción **Comenzar un proyecto nuevo...**



Una vez dentro, cambiaremos el nombre de la pantalla **Screen1** por **Dibujar** (en la propiedad **Título** del componente **Screen1**).

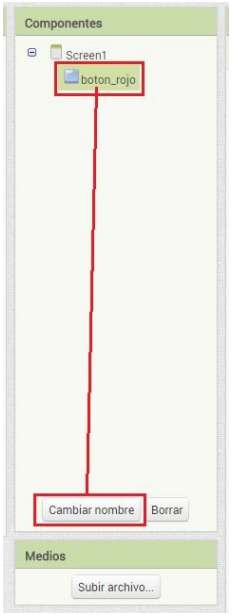


ATENCIÓN

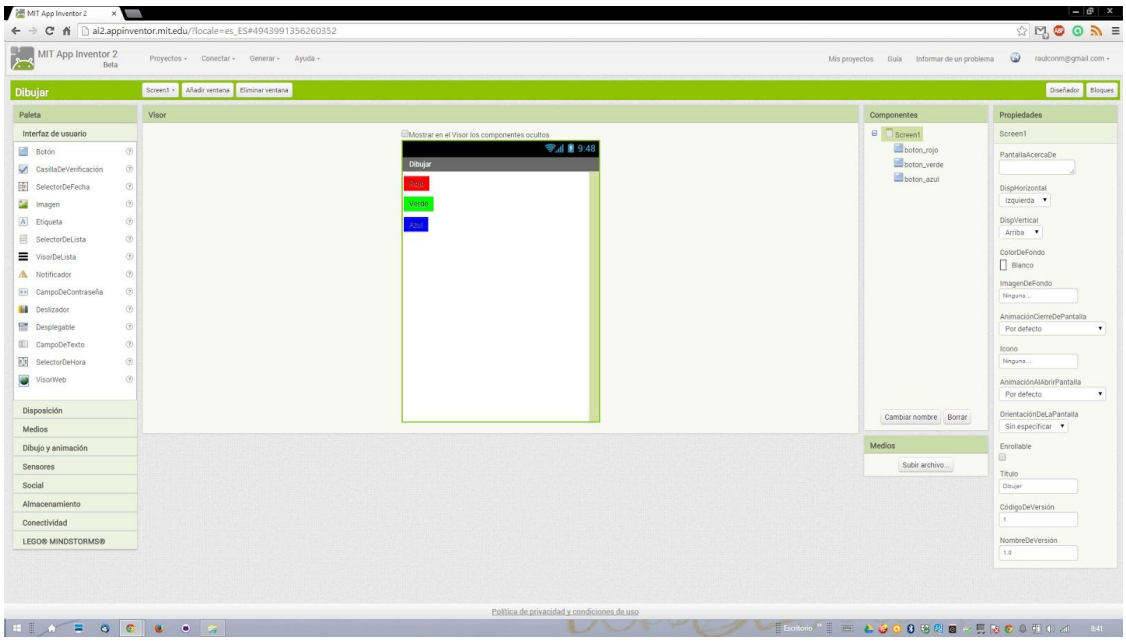
Es importante siempre utilizar nombres descriptivos para los objetos que vamos creando. Al principio, cuando nuestras aplicaciones son pequeñas, es fácil recordar cada objeto, pero a medida que los programas van haciéndose más complejos es fundamental saber para qué sirve cada objeto, cada variable, y sólo podremos hacerlo si le hemos dado un nombre que describe qué es o para qué sirve.

Ahora crearemos por nuestra cuenta un botón, y cambiaremos su propiedad **Texto** por "Rojo", y finalmente pondremos en rojo su propiedad **ColorDeFondo**. Adelante, solamente hay que jugar con las propiedades del botón.

Cambiaremos el nombre del botón **Botón1** por **boton_rojo** en el cuadro de componentes del Diseñador. En lugar de usar un espacio utilizaremos un signo de subrayado, y no emplearemos tildes, porque el sistema entiende que es un carácter especial no válido (esto aún está en inglés ☺).



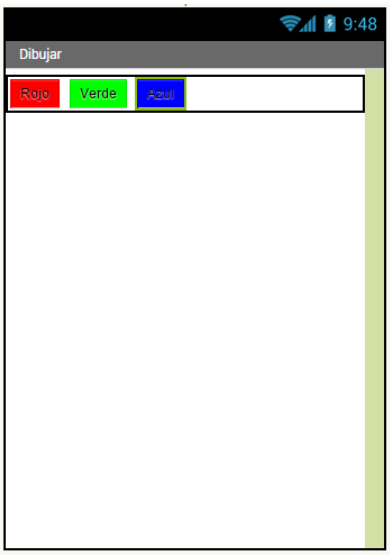
Crearemos después otro botón verde, y finalmente otro azul, y haremos lo mismo que hemos hecho con el rojo, cambiaremos su nombre, el color de fondo, y el texto que aparece en el botón.



Como queremos que los botones queden en una misma línea horizontal, vamos a recolocarlos en la pantalla añadiendo un componente **DisposiciónHorizontal**. Lo arrastramos desde el cajón **Disposición** hasta el Visor. Este objeto aparecerá en el Visor como un cuadro vacío.

Para que se ajuste al ancho de la pantalla del visor ¿qué haremos? En su propiedad **Ancho** elegiremos la opción **Ajustar al contenedor...**

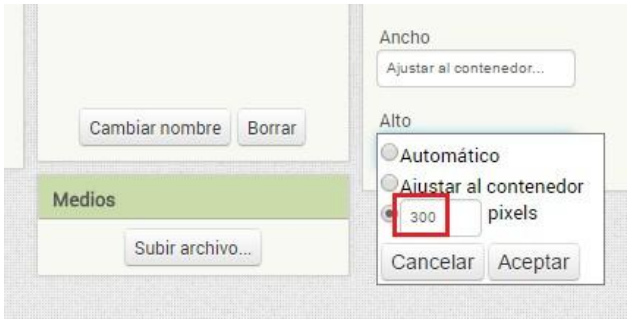
Ahora, en el Visor, arrastraremos los tres botones dentro de **DisposiciónHorizontal1**. Si no nos gusta este nombre podemos cambiarlo por **MarcoHorizontal**, por ejemplo. Para que la altura del marco se ajuste al tamaño de los botones podemos decir que su **Altura** sea automática. Así siempre se adaptará al tamaño del componente más alto de todos los que contiene.



Ahora añadimos el lienzo, arrastrando al Visor el componente **Lienzo**, que se encuentra dentro del cajón **Dibujo y animación** de la Paleta. Lo colocamos justo debajo, fuera del **MarcoHorizontal**. Lo más cómodo es definir que su anchura (propiedad **Ancho**) sea automática, para que se extienda hasta los bordes izquierdo y derecho del Visor. En cuanto a su altura, mejor experimentar con diferentes tamaños hasta que ocupe el espacio que queremos.

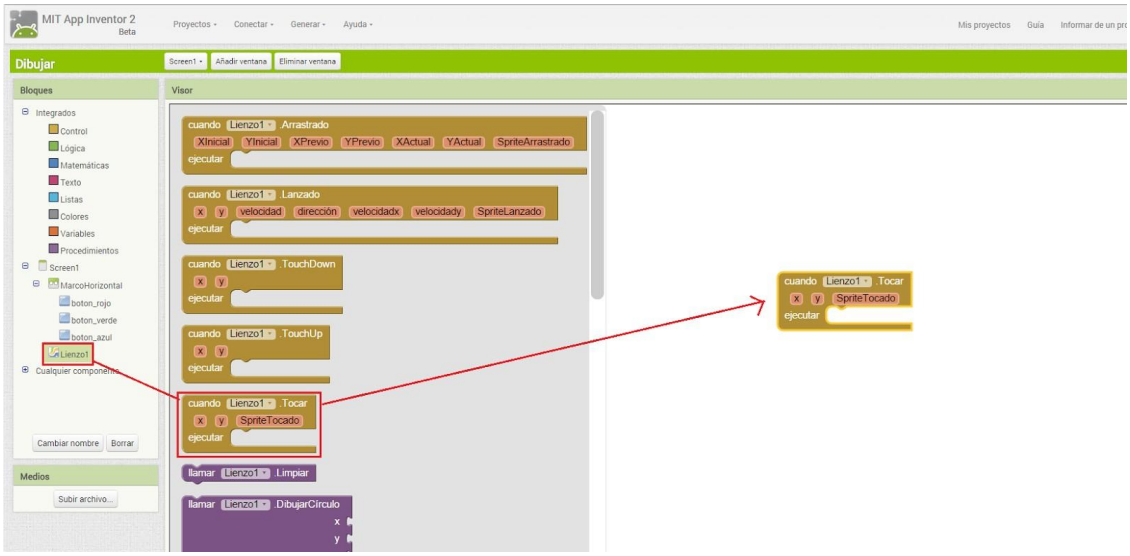
El tamaño se especifica en *pixels*, es decir, en puntos de la pantalla. Cada línea de la pantalla tiene un número de pixels. El número de ellos que haya, en líneas y columnas,

define la resolución de la pantalla. Podemos probar con 300 pixels, y modificarlo más tarde si vemos que no es el mejor tamaño.



Ahora vamos a definir el comportamiento del programa, a decir cómo tiene que funcionar. Vamos entonces al editor de bloques.

Arrastraremos el bloque **cuando.Lienzo1.Tocar** desde el cajón del objeto **Lienzo1** hasta el editor. Esto indica que cada vez que toquemos el lienzo con el dedo tendrá que ocurrir lo que digamos dentro de este bloque mostaza.



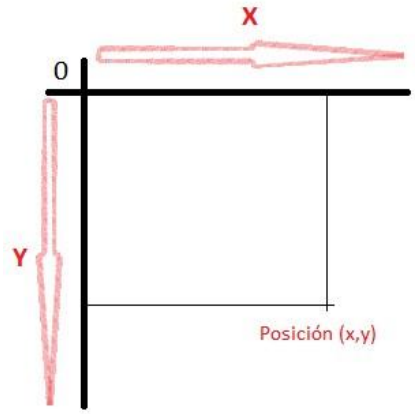
Ahora arrastraremos el bloque morado (de acción) **llamar.Lienzo1.DibujarCírculo** hasta encajarlo dentro del bloque mostaza. Veremos que para funcionar necesita que le asociemos a la derecha tres bloques de información adicional (x, y, r).

Coordenadas x e y

El pensador René Descartes inventó hace tiempo un sistema para definir la posición de un objeto sobre un plano. En su honor, este sistema se conoce como *Coordenadas Cartesianas*. Para conocer más sobre este tema visitemos la página http://es.wikipedia.org/wiki/Coordenadas_cartesianas

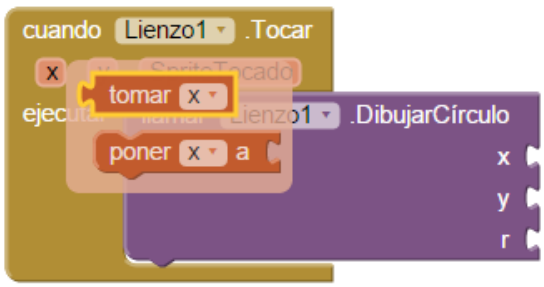
Nosotros vamos a utilizar el sistema de Descartes para referirnos a un punto cualquiera dentro de nuestro **Lienzo1**. Siempre que toquemos con el dedo dentro del lienzo, el

bloque mostaza guardará cuál es el valor de la coordenada x y la coordenada y del punto que estamos tocando, porque seguro que lo necesitaremos. App Inventor utiliza el sistema cartesiano para determinar la posición de un punto determinado dentro de un lienzo. La x aumenta desde la izquierda a la derecha, y la y lo hace desde arriba hacia abajo.



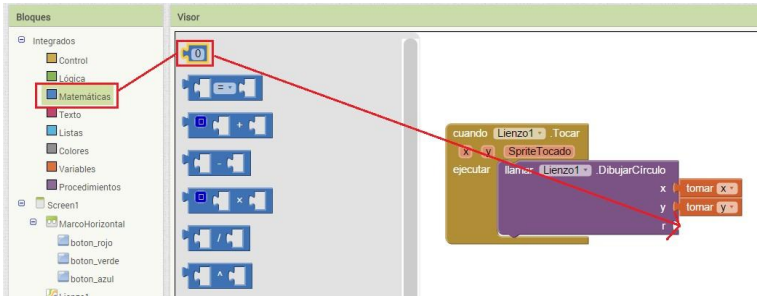
Precisamente para saber dónde dibujar el círculo, el bloque morado deberá saber que saber qué **x** e **y** tiene el punto de la pantalla que estamos tocando. Lo tomaremos del bloque mostaza.

Al dejar quieto el puntero del ratón sobre el icono de la **x** del bloque mostaza nos saldrá una ventanita en la que aparecerá un pequeño bloque **tomarx** de color naranja. Tenemos que arrastrarlo hasta encajarlo con el hueco superior del bloque morado **llamar.Lienzo1.DibujarCírculo**.



Haremos lo mismo con la **y**. Esto hará que el círculo se dibuje sobre la x y la y que estamos tocando, y no en cualquier otro punto de la pantalla.

Nos falta definir la **r**. ¿Qué es? Indica el tamaño del círculo que vamos a dibujar, su radio (de ahí la r). Para definir el tamaño, el radio, abrir el cajón **Matemáticas** de la zona de bloques y arrastrar el bloque azul que indicar el valor 0 (cero) hasta el hueco **r** de nuestro bloque morado.



Queremos que los círculos se vean bien, así que le asignaremos a **r** el valor 10. El radio del círculo será 10.

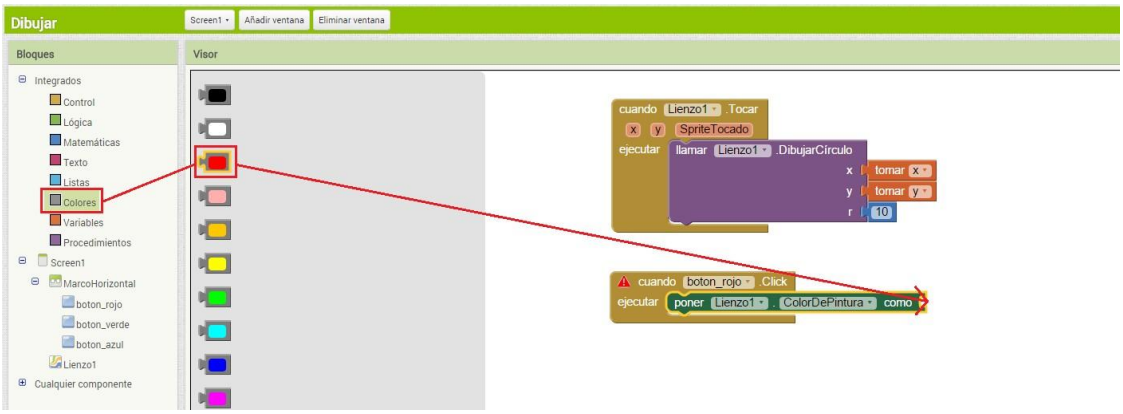
Ahora podemos tocar dentro del lienzo, en la pantalla del móvil o el emulador, para ver cómo se dibujan pequeños círculos. Pero para que los círculos sean rojos, verdes o azules en lugar de negros, hay que decirle al móvil que lo haga creando nuevos bloques. Una pista: hay que utilizar un bloque mostaza del objeto **botón_rojo**. Ya hemos usado este tipo de bloque mostaza en el programa del gato.

Utilizaremos el bloque **cuando.boton_rojo.Clic**. Lo arrastramos hasta el editor de bloques.

¿Qué tiene que ocurrir cuando hagamos clic sobre el botón rojo? Tiene que cambiarse a rojo el color del pincel que usamos en el lienzo. ¿Dónde tendremos que definir eso? Una pista: el color del pincel es una característica del **Lienzo1**, así que tendremos que buscar la manera de hacerlo usando algún bloque del cajón del objeto **Lienzo**. Otra pista: es un bloque de color verde oscuro.

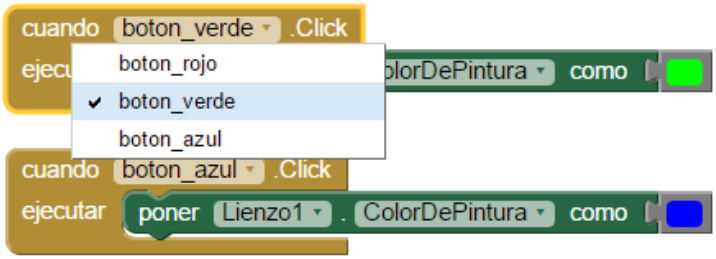
Es el bloque **poner.Lienzo1.ColorDePintura.como.** Lo arrastraremos hasta encajarlo dentro del bloque mostaza **cuando.boton_rojo.Clic**.

Falta indicar que queremos que sea el color rojo. Para ello escogeremos el pequeño bloque que identifica a este color de entre los colores que hay en el cajón **Colores** de la Paleta de App Inventor.



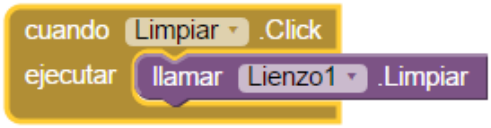
Haremos lo mismo con los botones para color verde y color azul. Lo mejor es hacerlo duplicando el bloque mostaza que hemos creado para el color rojo. Una vez hecho, en el nuevo bloque mostaza, podemos desplegar la lista del nombre de los botones y

elegir el del color verde. Entonces sólo tendremos que cambiar el bloque rojo por el verde, y listo. Lo mismo con el botón para el azul.



Cuando probemos el resultado se nos "ensuciará" el lienzo enseguida, así que hay que poner un botón para dejar el lienzo en blanco de nuevo. ¿Cómo se hará? Varias pistas:

- Crear **DisposiciónHorizontal1** debajo del lienzo
- Meter dentro un nuevo botón y llamarlo **Limpiar**
- Cambiar el texto del botón por "Limpiar"
- Poner los bloques para que al hacer clic sobre **Limpiar** se limpie el lienzo

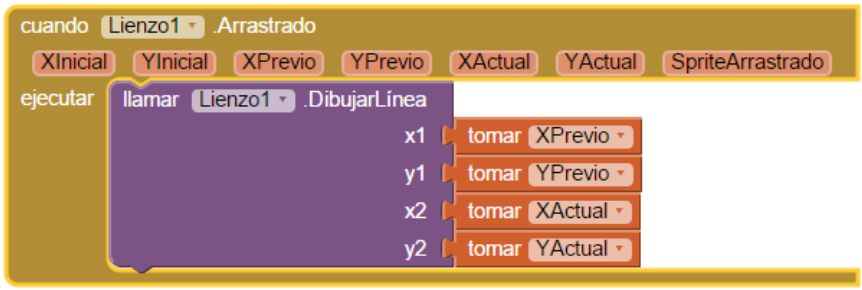


En realidad sería más interesante si en lugar de hacer circulitos, pudieran dibujarse líneas al arrastrar el dedo por la pantalla. ¿Qué bloque mostaza del objeto **Lienzo** habría que usar para que se dibujara según fuéramos arrastrando el dedo por el lienzo? Una pista: deberemos utilizar un bloque naranja del componente **Lienzo1**.

ATENCIÓN
En programación no hay una única manera de hacer bien las cosas, es decir, podemos conseguir el mismo resultado utilizando bloques diferentes. El objetivo crear el programa de la forma más simple y más eficiente, para que nuestro código sea más "elegante".

Podemos usar estos dos tipos de bloques, por ejemplo, para dibujar un círculo por cada punto donde vayamos arrastrando el dedo. **XActual** e **YActual** se refieren precisamente al punto representado por las coordenadas x e y actuales. Veremos como funciona esto al arrastrar el dedo despacio por el lienzo. Si deslizamos el dedo demasiado rápido se verán los puntos separados.

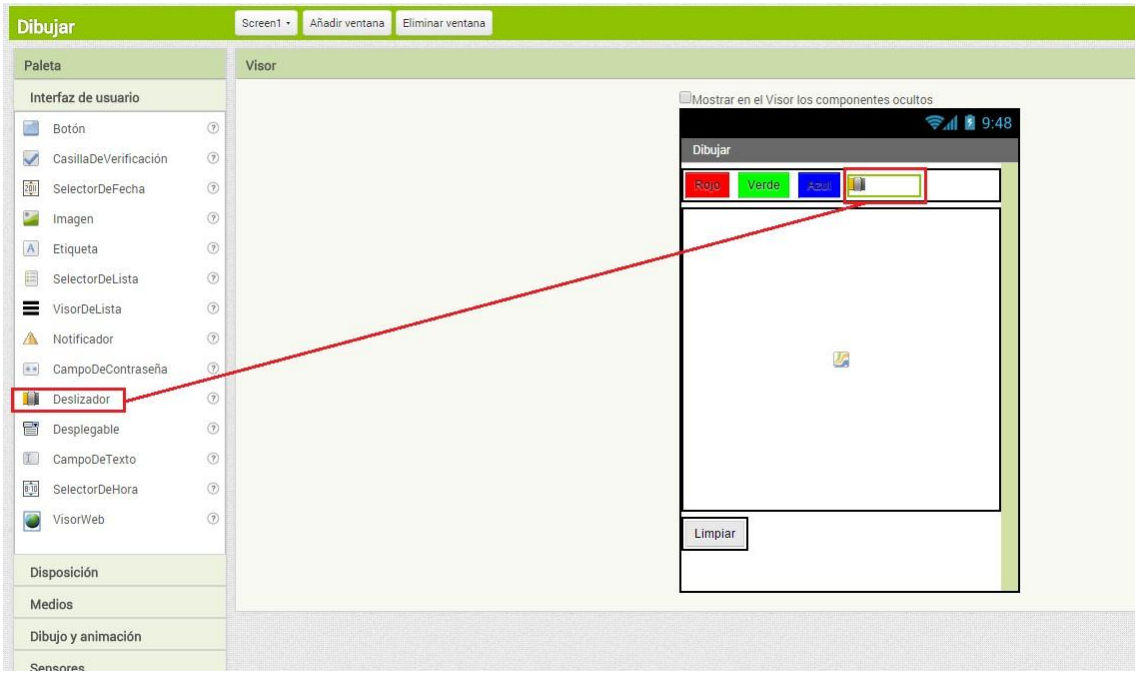
Más apropiado puede ser entonces utilizar este otro bloque morado, que se basa en dibujar líneas, y no círculos. Para dibujar una línea hace falta especificar el punto inicial de la línea y el punto final. Por ello para usar este bloque hay que indicarle dos coordenadas x y dos y. El primer (x1, y1) define el punto inicial de la línea, y el segundo par (x2, y2) define el punto final de la misma. Al hacer clic sobre el lienzo, sin levantar el dedo, el programa guardará en su memoria el valor de la x e y del punto que hemos tocado (determinado por XPrevio e YPrevio). Según vayamos deslizando el dedo el programa irá detectando el cambio de posición y dibujará una línea desde ese punto guardado hasta el punto que estamos tocando actualmente (determinado por XActual e YActual). Esto se repetirá todo el tiempo, muy rápidamente, mientras sigamos arrastrando el dedo por la pantalla. Se dibujarán por lo tanto muchas líneas muy cortas, una detrás de la otra, dando la sensación de ser una única línea continua.



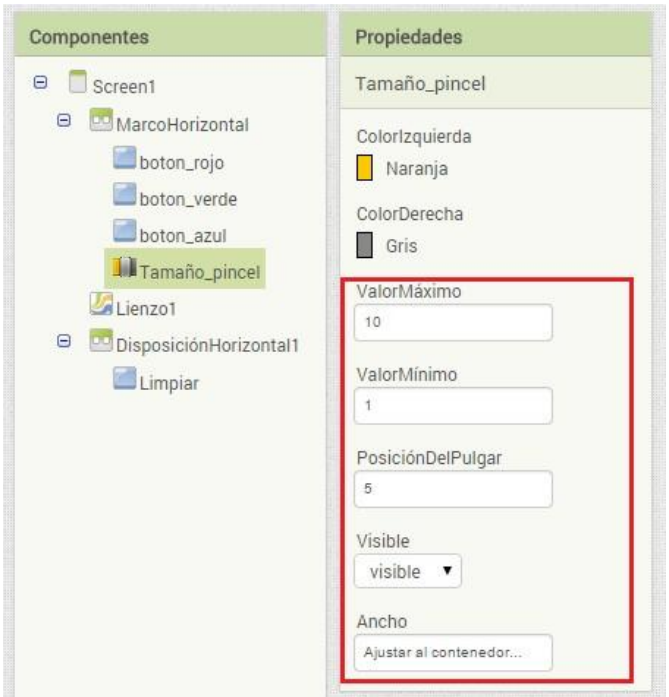
Como hemos decidido utilizar el bloque **cuando.Lienzo1.Arrastrado** podemos eliminar el bloque naranja **cuando.Lienzo1.Tocar** que había utilizado anteriormente, y que dibujaba círculos, a no ser que queramos mantener ambos botones, par que nuestra aplicación haga dos cosas distintas dependiendo de la acción que tomemos: si sólo tocamos el lienzo se dibujará un círculo, y si tocamos el lienzo y deslizamos el dedo se dibujará una línea.

Cambiar el tamaño del pincel con un Deslizador

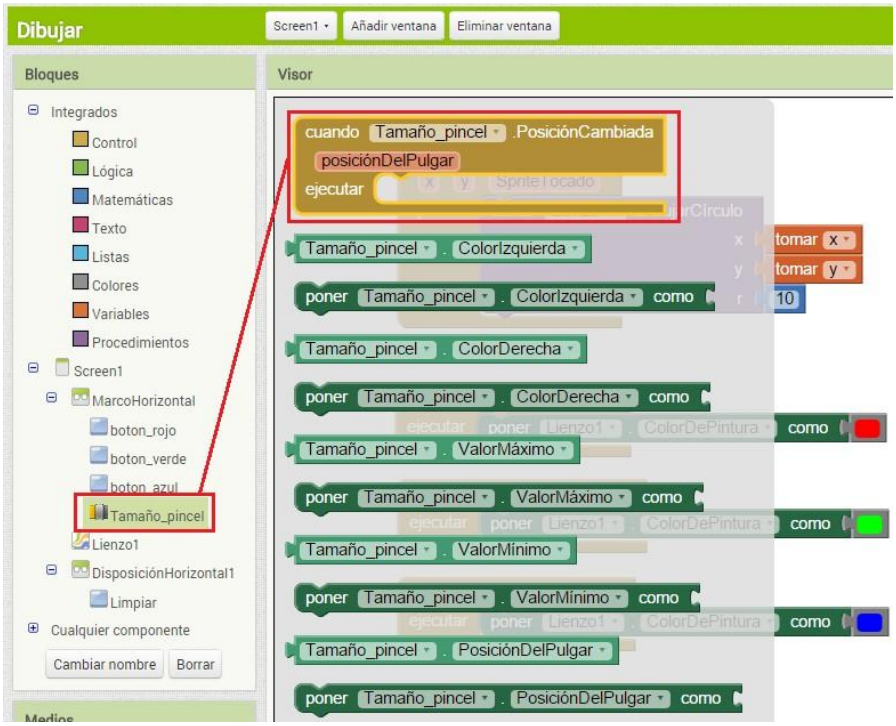
Un objeto **Deslizador** es perfecto para modificar el valor numérico de un parámetro sin tener que escribirlo. Para incluir un deslizador en nuestro interfaz de usuario tenemos que arrastrarlo desde la Paleta. En nuestro caso lo llamaremos **Tamaño_pincel**.



Tenemos que definir el valor mínimo y máximo que podrá tomar el deslizador, y el valor que tendrá inicialmente (**PosiciónDelPulgar**). También es conveniente especificar la anchura del Slider para que se ajuste al espacio restante (**Ajustar al contenedor**).

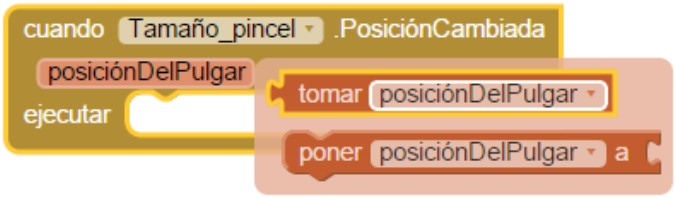


Tenemos que indicarle ahora a nuestro programa que modifique el ancho de la línea de nuestro dibujo según variemos la posición de **Tamaño_pincel**. Para ello, utilizaremos el bloque mostaza **cuando.Tamaño_pincel.PosiciónCambiada**, que se ejecuta cada vez que se cambia la posición del selector dentro del deslizador.

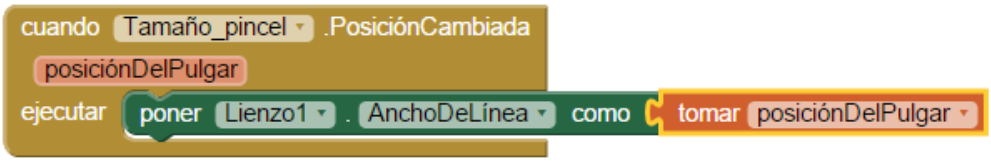


Este bloque incluye una variable, **posiciónDelPulgar**, que contiene el valor actual de la posición, y que puede utilizarse para nuestro propósito.

Si dejamos momentáneamente el puntero del ratón sobre la casilla **posiciónDelPulgar** del bloque mostaza, se mostrará un desplegable que nos permitirá elegir el bloque que define el valor de esa variable (**tomar**), o definir un nuevo valor para ella (**poner**).



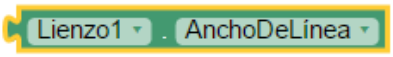
Elegiremos en este caso el bloque **tomar**, y lo combinaremos con el bloque verde del objeto **Lienzo** que define el ancho de línea de dibujo, para indicar al programa que cambie el ancho de la línea cada vez que arrastremos el puntero del deslizador **Tamaño_pincel** a un nueva posición.



Igual que ocurre con los bloques de color naranja **tomar** y **poner** que apareccen dentro de algunos bloques mostaza, y que sirven para poner o tomar un valor (en este caso para **posiciónDelPulgar**), en muchos componentes existen bloques de color verde, que nos permiten tomar o poner valores a sus propiedades.

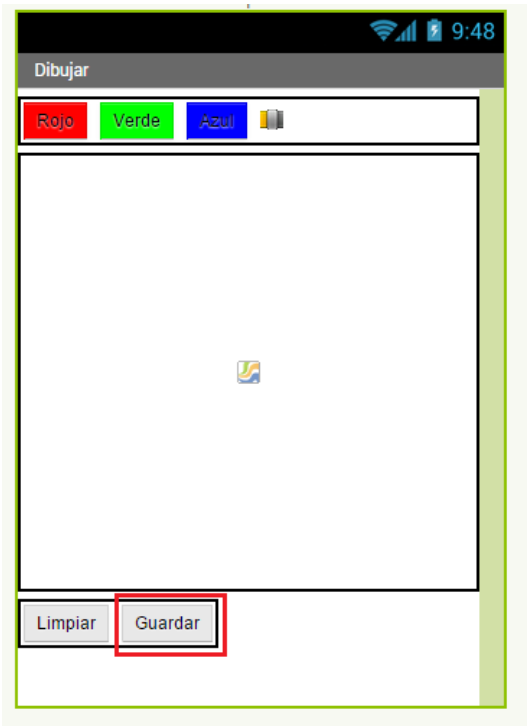
Por ejemplo, el siguiente bloque nos permite **poner**, asignar, definir el valor del ancho de línea del componente **Lienzo1**. La propiedad **AnchoDeLínea** de **Lienzo1** tendrá entonces el valor del bloque que encajemos a la derecha del bloque verde oscuro.

Y este otro sirve para **tomar** el valor actual del ancho de línea del componente **Lienzo1**. Es decir, cuando encajemos este bloque verde claro a la derecha de un bloque verde oscuro, éste tomará el valor de la propiedad **AnchoDeLínea** de **Lienzo1**.

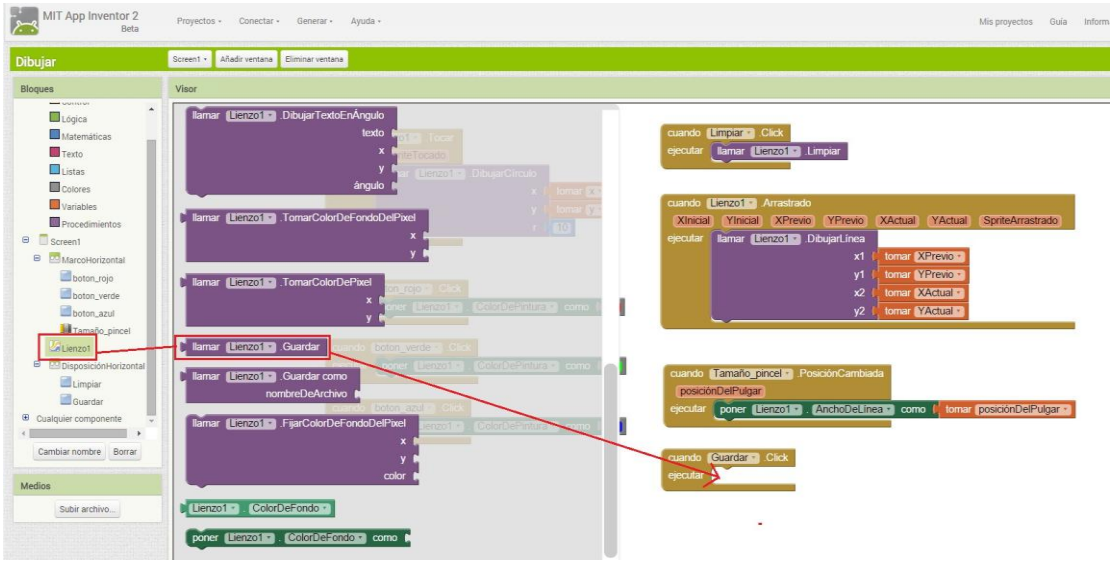


Guardar un archivo con el dibujo que hemos hecho

Cuando el usuario termine de hacer un dibujo interesante, seguramente querrá conservarlo en la memoria permanente del teléfono. Para ofrecerle esa posibilidad, tendremos que indicarle a nuestro programa cómo almacenar en la memoria el contenido actual del lienzo. En primer lugar, hay que incluir en el interfaz un botón **Guardar**, que al ser pulsado ejecutará el código correspondiente.

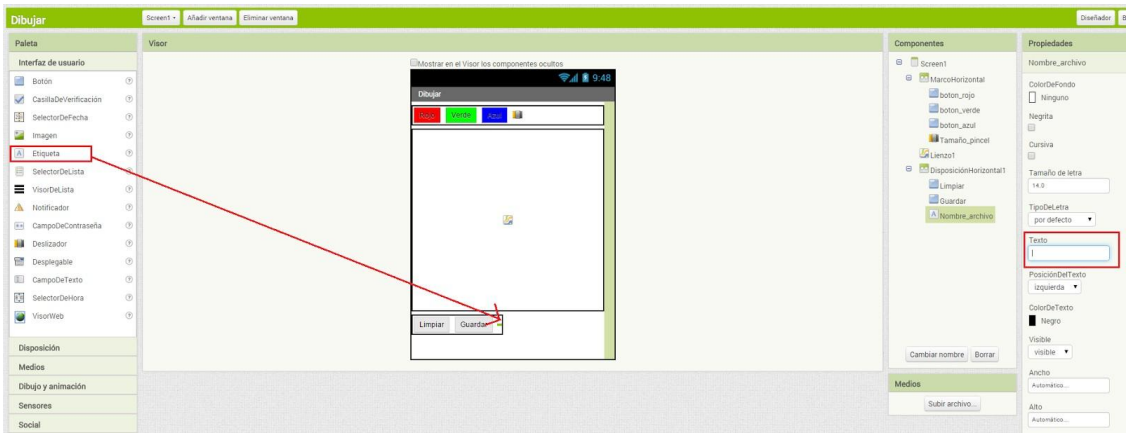


El bloque de código que usaremos, incluido dentro del cajón Lienzo, será **llamar.Lienzo1.Guardar**.



Este bloque es diferente a los vistos hasta ahora, porque no puede ser encajado directamente debajo de otro bloque, o dentro de un bloque color mostaza. El bloque tiene que ser encajado a otro por su izquierda. Esto se debe a que es un bloque que devuelve un valor, como acabamos de ver al final del apartado anterior, con el bloque de color verde claro **Lienzo1.AnchoDeLínea**. En este caso el bloque violeta devuelve una texto, que contiene el nombre del archivo donde se ha almacenado el dibujo dentro de la memoria del dispositivo.

Es conveniente colocar un componente Etiqueta, que llamaremos **Nombre_Archivo,** a la derecha del botón **Guardar**, y que tomará el valor de texto devuelto por el procedimiento **Lienzo1.Guardar**. Definiremos que su propiedad **Texto** esté inicialmente en blanco. El componente será difícil de ver en el dispositivo mientras no le asignemos un nuevo valor.



Veamos cuidadosamente el nuevo componente y los bloques utilizados, hasta interiorizar mentalmente su funcionamiento.

Cada vez que se pulse el botón **Guardar**, el dibujo quedará almacenado con un nombre que se asignará automáticamente, en función del día y la hora. Este nombre aparecerá en el texto de la etiqueta **Nombre_archivo** cada vez que hagamos clic en el botón **Guardar**.

Ideas para mejorar la aplicación

Podemos mejorar esta aplicación hasta donde queramos, usando la imaginación, e investigando cómo podemos utilizar los recursos en App Inventor para incluir en nuestra aplicación todo lo que vayamos inventando.

Por ejemplo…

- Poner un sello con tu nombre cuando pulses un botón
- Tomar una foto existente en la memoria del dispositivo y usarla como fondo del lienzo, etc.