Fuente: www.appinventor.org
Traducción hecha con Google Traductor y mejorada por mi: piatticarlos@gmail.com

Programando la memoria de tu aplicación

Así como la gente necesita recordar cosas, las aplicaciones también. En este capítulo se explica cómo se programa una aplicación para memorizar información.

Cuando alguien le dice el número de teléfono de un pizzería para realizar una llamada inmediata, su cerebro lo almacena en una posición de memoria. Si alguien le dice en voz alta algunos números para que los anote, también almacena los resultados inmediatos en una posición de memoria. En tales casos, no somos plenamente conscientes de cómo el cerebro almacena o recuerda la información. Una aplicación también tiene una memoria, pero su funcionamiento interno no es misterioso como el cerebro humano. En este capítulo, usted aprenderá cómo configurar la memoria de una aplicación, la forma de almacenar la información en ella, y cómo recuperar esa información en un momento posterior.

Los llamados Slots de memoria

La memoria de una aplicación se compone de un conjunto de celdas de memoria. Algunos de estos *slots* de memoria se crean en su aplicación cuando se arrastra un componente, estas celdas se denominan propiedades. También se puede definir con el nombre slots de memoria a los que no están asociados con un componente en particular, los cuales se llaman variables. Mientras que las propiedades son típicamente asociadas con lo que es visible en una aplicación, las variables pueden ser consideradas como ocultas en la aplicación y son el "principio" de la memoria.

Propiedades

Los componentes (al menos los visibles como Button, TextBox y Canvas) forman parte de la interfaz de usuario. Sin embargo, para la aplicación, cada componente está completamente definido por un conjunto de propiedades. Los valores almacenados en los slots de memoria de cada propiedad determinan cómo aparece el componente.
Se pueden modificar las propiedades de los slots de memoria directamente en el Diseñador de componentes, como se muestra en la Figura 16-1.

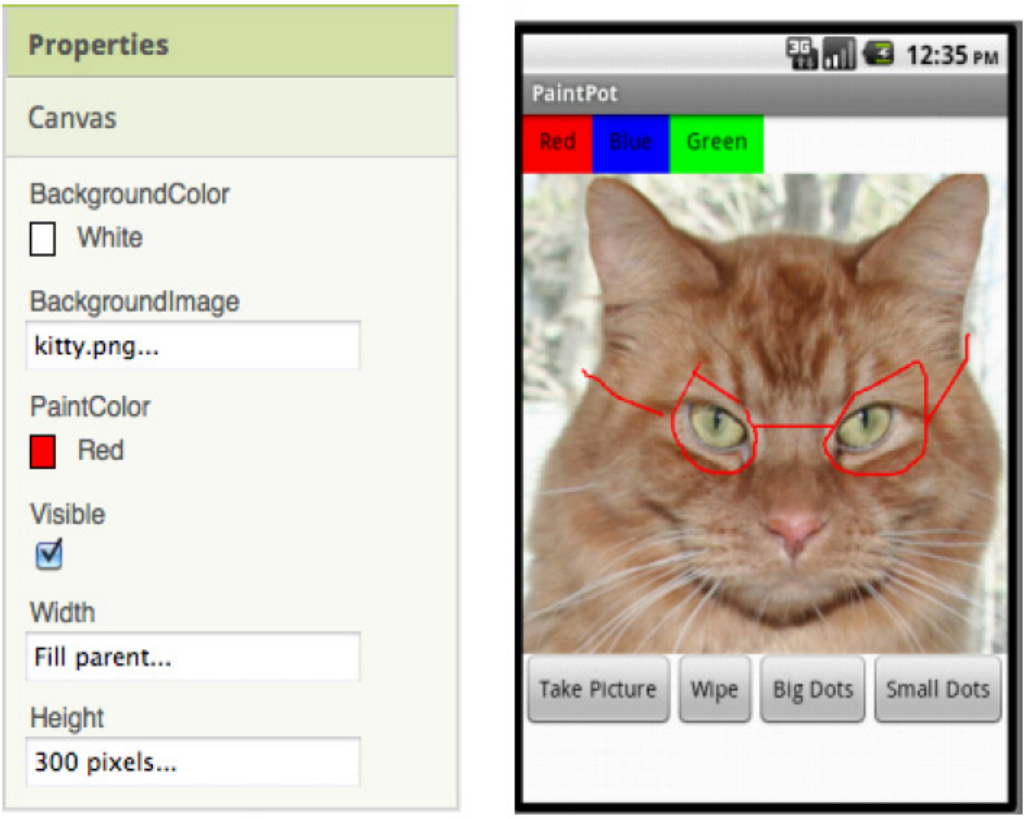

Figure 16-1. Modifying the memory slots in the property form to change the app's appearance

Figura 16-1. Modificando los slots de memoria en la ventana de propiedades para cambiar la apariencia de la aplicación

El componente *Canvas (lienzo)* de la figura 16-1 tiene seis propiedades. El BackgroundColor y PaintColor son slots de memoria que tienen un color. El BackgroundImage tiene un nombre de archivo (kitty.png). La propiedad Visible tiene un valor booleano (verdadero o falso, dependiendo de si la casilla está marcada). Los slots Width y Height contienen un número o una designación especial (por ejemplo, "patron de relleno").
Cuando se cambia una propiedad en el Diseñador de componentes, se especifica cómo debería aparecer la aplicación cuando se pone en marcha. El que usa la aplicación (el usuario final) nunca ve que hay un slot de memoria denominado Height que contiene un valor de 300. El usuario final sólo ve la interfaz de usuario con un componente que tiene la altura de 300 píxeles.

Al igual que las propiedades, a las variables se las llaman slots de memoria, pero no están asociadas con un componente en particular. Una variable se define cuando su aplicación tiene que recordar algo que no se va a almacenar en una propiedad de componente. Por ejemplo, un juego que tenga que recordar cuál es el nivel que el usuario ha alcanzado. Si el número de nivel se va a mostrar en un componente Label, puede que no necesite una variable, ya que sólo

podría almacenar el nivel en la propiedad Text del componente Label. Pero si el valor de *level* (nivel) es algo que el usuario no verá, usted define una variable para almacenarla.

El Cuestionario Presidentes (Capítulo 8) es otro ejemplo de una aplicación que necesita una variable. En esa aplicación, debe aparecer solamente una pregunta del cuestionario a la vez en la interfaz de usuario, mientras que el resto de las preguntas se mantienen ocultas para el usuario. Por lo cual, es necesario definir una variable para almacenar la lista de preguntas. Mientras que las propiedades se crean automáticamente cuando se arrastra un componente en el Diseñador de componentes, una nueva variable se puede crear de forma explícita en el Editor de bloques arrastrando un bloque **def variable**. Usted puede nombrar la variable haciendo clic en el texto "variable" dentro del bloque, y especificar un valor inicial arrastrando un **número**, **texto**, **color**, o un bloque **make a list** y conectarlo. Aquí están los pasos que habría de seguir para crear una variable llamada score con un valor inicial de 0:

1- Arrastre el bloque **def variable** (Figura 16-2) desde la carpeta “Definitions” de los bloques integrados.

Figure 16-2. A def variable block

2- Cambie el nombre de la variable haciendo clic en el texto "variable" y escribiendo "Score", como se ilustra en la Figura 16-3.

Figure 16-3. Changing the variable name

3- Establezca el valor inicial del número arrastrando el bloque number (número) y conectándolo en la definición de la variable (Figura 16-4).

Figure 16-4. Setting the value to a number

4- Cambiar el valor inicial del número predeterminado (123) a 0 (Figura 16-5).

Figure 16-5. Setting the initial value to 0

Cuando se define una variable, se le dice a la aplicación que establezca un nombre a la posición de memoria para almacenar un valor. Estas posiciones, como así también las propiedades, no son visibles para el usuario.
El bloque de inicialización que usted conecte especifica el valor que se debe colocar en la posición cuando la aplicación se inicia. Además de iniciar con números o texto, también puede colocar un bloque **make a list** dentro del bloque **def var**. Esto le indica a la aplicación que los nombres de las variables sean de una lista de posiciones de memoria en lugar de un único valor. Para obtener más información acerca de las listas, consulte el Capítulo 19.

Establecer y obtener una variable
Cuando se define una variable, App Inventor crea dos bloques para ello, los cuales aparecen en el cajón “My Definitions” (mis Definiciones), que se muestran en la Figura 16-6.

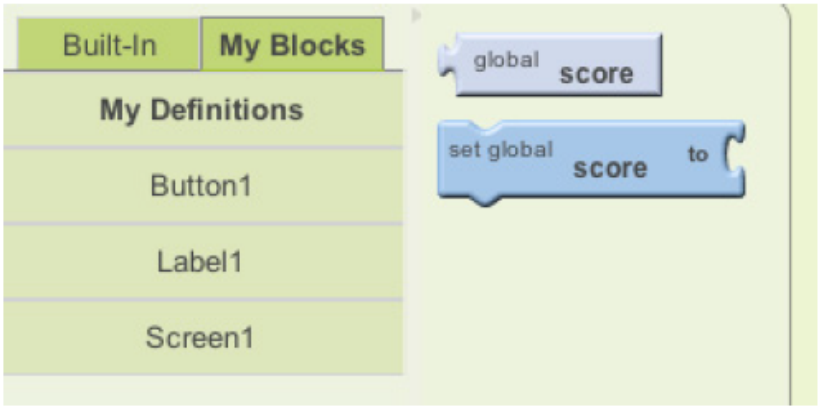

Figure 16-6. The My Definitions drawer contains set and get blocks for your variable

El bloque **set global to** permite modificar (ajustar) el valor almacenado en la variable. Por ejemplo, los bloques de la Figura 16-7 alojan un 5 en la variable **score**. El término "global" en el bloque **set global score to** se refiere a que la variable puede ser usada en todos los controladores de eventos del programa (a nivel general). Otros lenguajes de programación permiten definir variables que son "local" a una parte específica del programa; App Inventor no.

Figure 16-7. Placing a number 5 into the variable score

El bloque etiquetado **global score** ayuda a recuperar (obtener) el valor de una variable. Por ejemplo, si desea comprobar si la puntuación fue de 100 o mayor, conecte el bloque **global score** en la comprobación **if (si)**, como se muestra en la Figura 16-8.

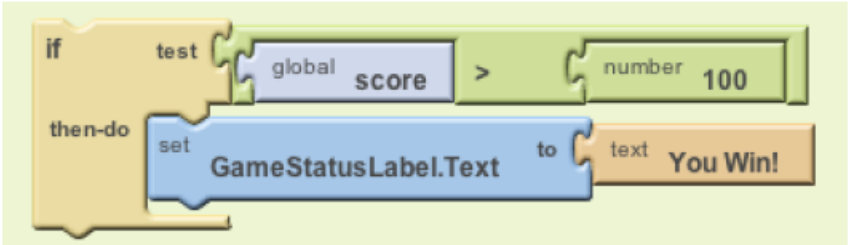

Figure 16-8. Using the global score block to get the value stored in the variable

Configuración de una variable a una expresión
Usted puede poner valores simples como un 5 en una variable, pero a menudo podrá establecer la variable a una expresión más compleja (*expresión* es el término “ciencias de la computación” para una fórmula). Por ejemplo, cuando el usuario hace clic en Siguiente para ir a la siguiente pregunta en la aplicación Quiz, usted tendrá que establecer la variable currentQuestion a uno más que su valor actual. Cuando alguien hace algo mal en una aplicación como la de un juego, es posible modificar la variable score a menos de 10 de su valor actual. En el juego MoleMash (capítulo 3), se cambia la ubicación horizontal (x) del topo a una posición aleatoria dentro del *canvas* (lienzo).

Incrementar una variable
Tal vez esta sea la expresión más común, o establecer una variable en función de su valor actual. Por ejemplo, en un juego, cuando el jugador recibe un punto, la variable score puede ser incrementada por 1. La figura 16-9 muestra los bloques para implementar este comportamiento.

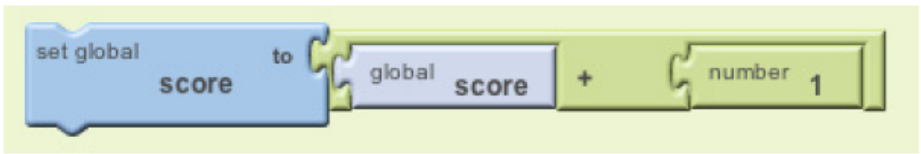

Figure 16-9. Incrementing the variable score by 1

Si usted puede entender este tipo de bloques, está encaminado para convertirse en un programador. Usted lee estos bloques como "poner en score uno más de lo que ya tiene",

que es otra manera de decir incrementar la variable. La forma en que funciona es que los bloques se interpretan al revés, no de izquierda a derecha. De modo que los bloques interiores (**global score** y el bloque number 1) se evalúan primero. Luego, el bloque **+** se lleva a cabo y el resultado es "ajustado" en la variable score.

Suponiendo que hubiera un 5 en el slot de memoria para score antes que sucedan estos bloques, la aplicación realizaría los siguientes pasos:

1. Recuperar el valor 5 del slot de memoria Score.

2. Añadir 1 al mismo para obtener 6.

3. Colocar el 6 en el slot de memoria Score (que sustituye al 5).

Para más información sobre incrementar, consulte el Capítulo 19.

Construyendo expresiones complejas
En la sección de Matemáticas de los bloques integrados (Figura 16-10), App Inventor ofrece una amplia gama de funciones matemáticas similares a las que te gustaría encontrar en una hoja de cálculo o una calculadora.

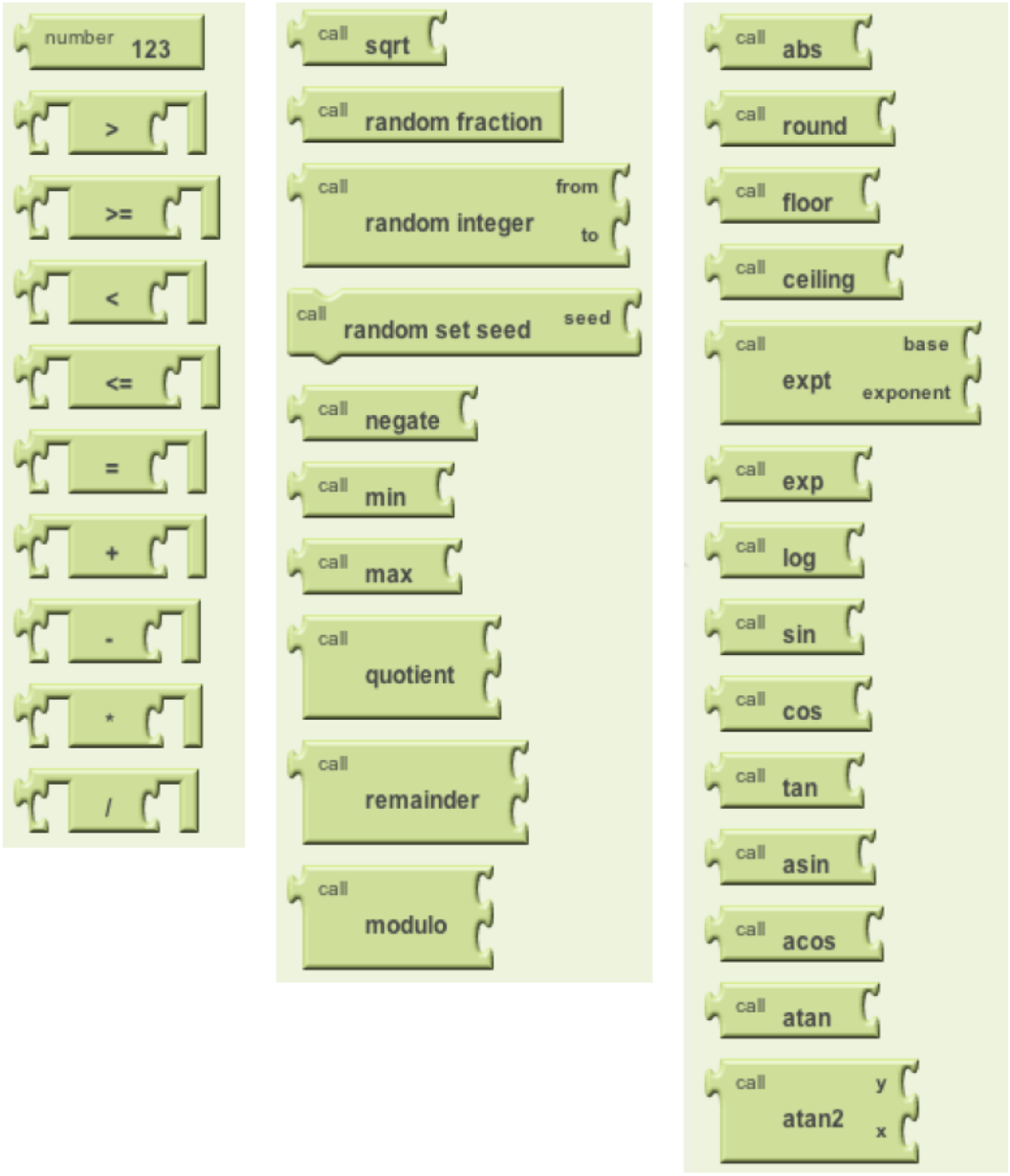

Figure 16-10. The blocks contained in the Math drawer

Puede utilizar estos bloques para construir una expresión compleja y luego conectarlos en la *expresión del lado derecho* del bloque **set variable to.** Por ejemplo, para mover un *sprite* con imagen en una columna al azar dentro de los límites de un *canvas*, se podría configurar una expresión que consiste en un bloque * (multiplicar), un bloque - (restar), y un canvas. La propiedad **width** (ancho), y la función **random fraction** (fracción aleatoria), como se ilustra en la figura 16-11.

Figure 16-11. You can use math blocks to build complex expressions like this one

Como en el ejemplo de incremento en la sección anterior, los bloques son interpretados por la aplicación de un modo de adentro hacia afuera. Suponiendo que el Canvas tiene un ancho de 300 y el ImageSprite tiene un ancho de 50, la aplicación podría realizar las siguientes pasos:

1- Recuperar los 300 y los 50 de los slots de memoria para el **Canvas1.Width** y **ImageSprite.Width**, respectivamente.

2- Restar: 300 – 50 = 250.

3- Llamar a la función **random fraction** (fracción aleatoria) para obtener un número entre 0 y 1 (por ejemplo, 0.5).

4- Multiplicar: 250 * .5 = 125.

5- Colocar los 125 en el slot de memoria para la propiedad ImageSprite1.X.

Mostrar las variables

Cuando usted modifica una propiedad de componente, como en el ejemplo anterior, la interfaz de usuario se ve directamente afectada. Esto no es así para las variables, cambiar una variable no tiene efecto directo en la apariencia de la aplicación. Si solo incrementa una variable de resultado, pero no modifica la interfaz de usuario de alguna otra manera, el usuario nunca sabrá que hubo un cambio. Es como el proverbio del árbol que cae en el bosque: si no hay nadie para oírlo, ¿realmente ocurrió?

Algunas veces usted no desea manifestar un cambio en la interfaz de usuario cuando cambia una variable. Por ejemplo, en un juego que puede rastrear las estadísticas (por ejemplo, tiros fallados), que sólo aparecen cuando el juego termina.

Esta es una de las ventajas de almacenar los datos en una variable en lugar de una propiedad de componente, no solo porque le permite mostrar los datos que desea cuando usted quiere, también le permite separar la parte computacional de la interfaz de usuario en su aplicación, haciéndola más fácil de cambiar que en un futuro hacerlo desde la interfaz de usuario.

Por ejemplo, en un juego que puede almacenar el resultado directamente en una etiqueta o en una variable, si lo almacena en una etiqueta, incrementa la propiedad Label´s Text cuando los puntos son anotados, y el usuario puede ver el cambio directamente. Si almacena el resultado en una variable y se incrementa la variable cuando se logran puntos, es necesario también

incluir bloques para mover el valor de la variable a una etiqueta.

Sin embargo, si usted decide cambiar la aplicación para mostrar el resultado de otra manera, la solución con variables sería la más fácil de usar. No tendría que encontrar todos los lugares que cambian la puntuación; esos bloques serían inmodificables. Sólo tendríamos que modificar los bloques correspondientes a la visualización.

Solucionarlo con Label en vez de utilizar variables sería más difícil de cambiar, ya que necesitaría reemplazar todos los cambios de incremento, por ejemplo con la modificación del width (ancho) de Label.

Resumen
Cuando una aplicación se inicia, comienza a ejecutar sus operaciones y a responder a eventos que ocurren. A veces al responder a eventos, es necesario recordar cosas. En el caso de un juego, podría ser la puntuación de cada jugador o la dirección en la que un objeto se está moviendo.

Su aplicación recuerda las cosas dentro de las propiedades del componente, pero cuando se necesitan más ubicaciones de memoria no asociadas a un componente, puede definir variables. Usted puede almacenar valores en una variable y recuperar el valor actual, al igual que lo hace con las propiedades.

Al igual que con los valores de propiedad, los valores de variables no son visibles para el usuario final. Si desea que el usuario final pueda ver la información almacenada en una variable, debe agregar bloques que muestran la información en una Label (etiqueta) o en otro componente de interfaz de usuario.

Fuente: www.appinventor.org
Traducción hecha con Google Traductor y mejorada por mi: piatticarlos@gmail.com