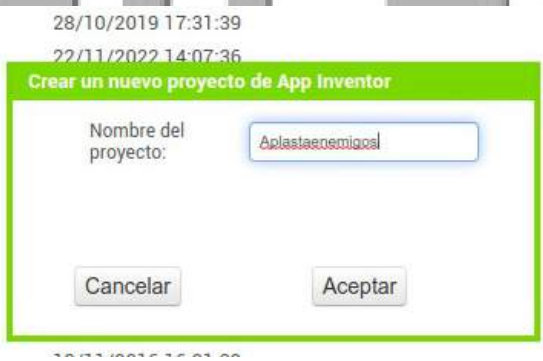
Profesor Marcelo Rebellato

Aplasta enemigos

Objetivo: Como decíamos antes, este juego consistirá en ir aplastando con nuestra pelota todos los objetos que aparezcan a lo largo y ancho del escenario. Para que la pelota ruede por el escenario tendremos que inclinar el dispositivo. La pelota rodará siempre hacia la parte del escenario que se encuentre más cerca del suelo. Recogeremos cada objeto cuando la pelota choque con él.

Comenzaremos un proyecto llamado “aplastaenemigos” para ello dentro del menú proyecto elegimos “Comenzar nuevo proyecto”.



Empezaremos creando el lienzo Escenario, y dentro él incluiremos un componente Pelota, que llamaremos Pelota1. De momento indicaremos en el Diseñador que la altura y la anchura del escenario se ajusten automáticamente al contenedor. En cuanto a las propiedades de la pelota, definiremos que su Radio es 15, para que se vea suficientemente grande en el escenario.

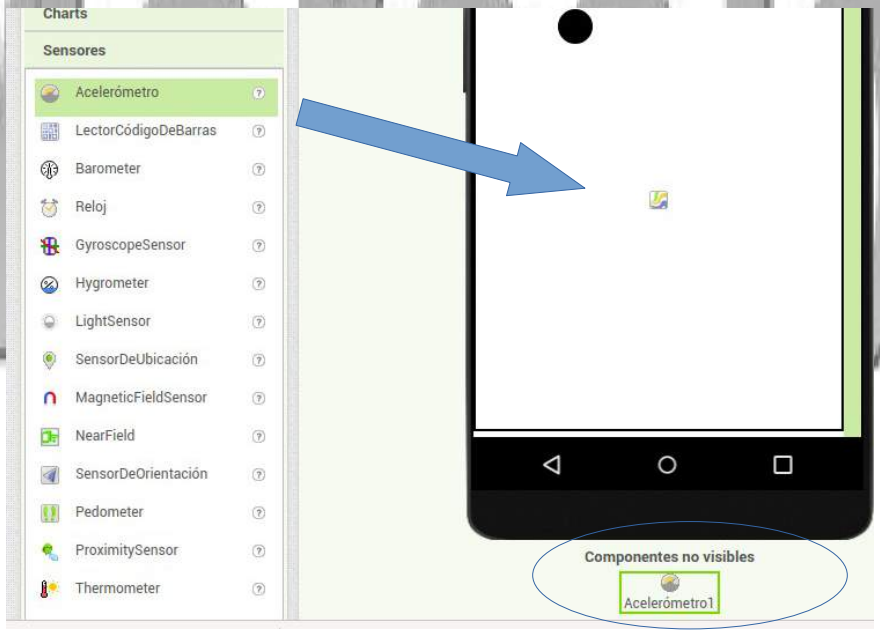
Tenemos que adaptar el escenario a los límites de la pantalla, tomando las propiedades de ancho y largo de la pantalla de cada dispositivo. Además, evitaremos que la pantalla rote automáticamente poniendo el valor de la propiedad OrientaciónDeLaPantalla a “Vertical”. Con esto la pantalla siempre mantendrá la orientación vertical, aunque inclinemos el dispositivo. No obstante, como esto no funciona con todos los dispositivos, en algunos casos será necesario desactivar manualmente la rotación de la pantalla en el propio dispositivo.

En Screen programaremos lo siguiente:

Esto permitirá que el lienzo “Escenario” se adapte a la pantalla y no se gire.

Movimiento de la pantalla

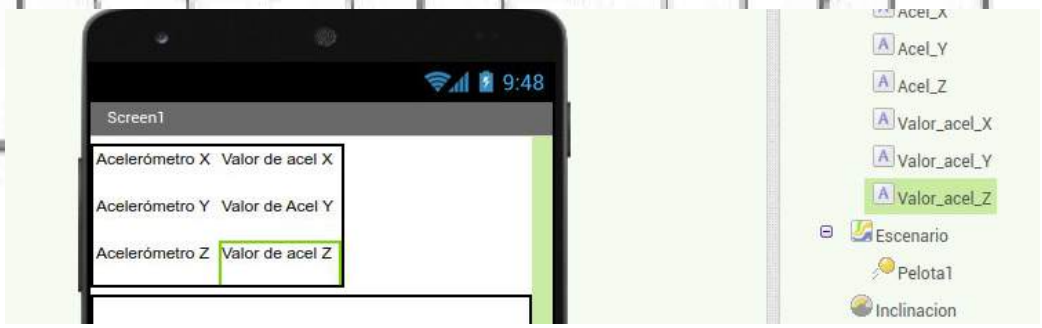
De la paleta **sensores** incluimos en nuestro visor un componente **Acelerómetro.** Lo llamaremos **Inclinación**. Este componente no es visible, así que aparecerá en la parte inferior del Visor, en la zona destinada a componentes no visibles.



Ahora utilizaremos el bloque **cuando,Inclinación.CambioEnAceleración** para mover la pelota. Este bloque incluye tres variables, cada una de las cuales almacena la inclinación del objeto en uno de los ejes de coordenadas. Vamos a hacer que la pelota tenga una dirección dependiendo de los valores de estas tres variables.

Como la definición de las dimensiones del escenario se hace dentro del **cuando.Screen1. Inicializar,** en nuestro caso probaremos reduciendo la altura del escenario en 150 unidades.

Usaremos **Disposición Tabular** de la paleta **Disposición**. Como queremos mostrar los datos de tres etiquetas con sus tres valores correspondientes, tendremos que indicar que la Disposición Tabular tenga dos Columnas con tres Registros.



En la programación utilizaremos el bloque **cuando Inclinación CambioEnAcceleración**, dentro pondremos el bloque **"Poner Valor ….. Texto como"** de las etiquetas correspondientes y en la zona x**Accel** o y**Accel** o z**Accel,** hacemos clic para encontrar los bloques **tomar xAccel, tomar yAccel y tomar zAccel.**

Veremos que para la Y los valores comprenderán de -10 (cuando el dispositivo está vertical y apuntando hacia el suelo) a 10 (cuando el dispositivo está vertical, y hacia arriba). Para la X, los valores comprenderán también entre 10 (cuando está completamente inclinado con la pantalla hacia la izquierda) y -10 cuando está completamente inclinado con la pantalla hacia la derecha).
En cuanto a la Z, inicialmente sin uso en este juego, irá de 10 (cuando el dispositivo tiene la pantalla hacia arriba) a -10 (cuando el dispositivo tenga la pantalla hacia abajo, paralela al suelo).

Comportamiento de la pelota

Dentro del bloque **cuando Inclinación CambioEnAcceleración**, pondremos de sprite "Pelota 1", el bloque "**Llamar Pelota 1 MoverA**". En el mismo encastramos el bloque suma de la paleta Matemática y dentro de este los bloques de "**Pelota1 X"** y "**Pelota1 Y**" del componente "Pelota1".
En la segunda parte del primer sumar incluimos uno de resta, donde dentro hacemos 1 – Tomar xAccel y en el segundo sumar el bloque Tomar yAccel

Cuando inclinemos el dispositivo hacia la derecha, en el eje X, la pelota caerá hacia la izquierda, y viceversa. Para solucionar este problema, y adaptar el funcionamiento al comportamiento natural de una pelota, fue que modificamos el bloque que indica el posicionamiento en la componente X de la coordenada. En lugar de sumarle el valor de xAccel le sumaremos el valor de restar xAccel a 1.

Profesor Marcelo Rebellato

Crear los personajes a aplastar

A continuación, y desde la ventana del Diseñador, incluiremos en nuestro escenario cinco objetos SpriteImagen iguales. Le daremos un nombre diferente a cada uno, y a todos le asignaremos el aspecto de nuestros dos sprite enemigos en la propiedad fondo y pondremos un alto de 50 pixel y un ancho de 30 pixel. Podemos llamarles Enemigo_1 a Enemigo_5.

Manejo de los enemigos

Con el bloque **llamar. enemigo_1.MoverA.** Especificaremos para cada objeto una posición aleatoria. Con los bloques azules **entero aleatorio entre** de la paleta matemática definiremos en qué coordenada X aparecerá la esquina superior del sprite dentro del escenario.

Con un 0 indicaremos que el objeto puede aparecer desde el margen izquierdo del escenario. A continuación le diremos con el bloque verde **Escenario.Ancho** que el límite máximo es el límite derecho del escenario.
Con lo anterior si el bloque azul **entero aleatorio entre** nos devolviera precisamente un valor de X muy cercano al límite derecho la mayor parte del enemigo sobrepasaría el

límite derecho del escenario, y no sería visible. Para solucionar esto se resta a **Escenario.Ancho** el ancho del sprite del enemigo, **enemigo_1.Ancho.**
Haremos lo mismo con la coordenada vertical Y, y repetiremos los mismos bloques cinco veces, porque hay cinco objetos iguales.

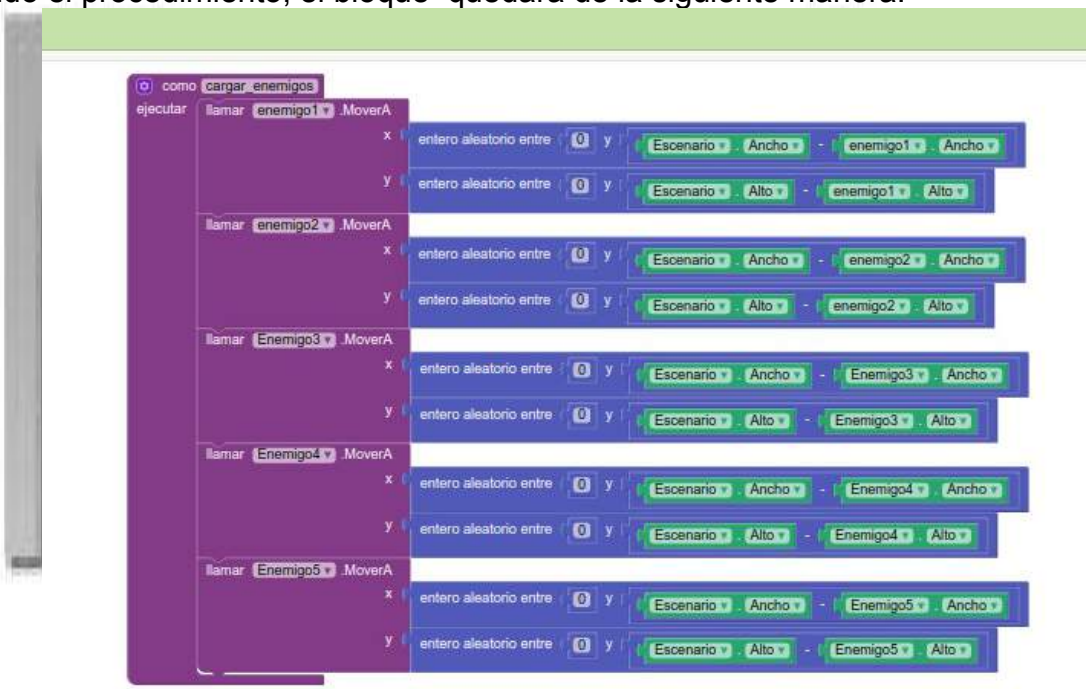
Procedimiento para tareas definidas y repetitivas

Definiremos un procedimiento, donde incluiremos todos los bloques que sirven para colocar los objetos.
Un procedimiento es un conjunto de pasos bien definidos para ejecutar una tarea concreta que debe ser realizada muchas veces.

Los procedimientos son muy importantes, porque permiten organizar mejor el código y ahorrar esfuerzo a la hora de programar.

El bloque para definir procedimientos se encuentra dentro de la paleta "**Procedimientos**".

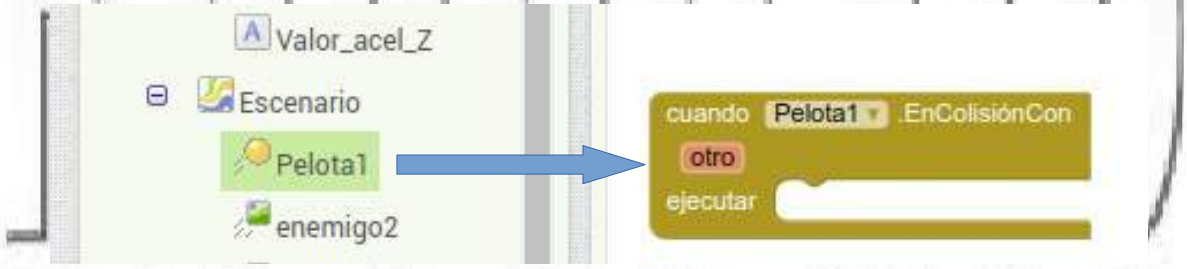
Definido el procedimiento, el bloque quedará de la siguiente manera.



En nuestro caso el procedimiento anterior lo llamaremos del bloque **cuando.Screen1.Inicializar ejecutar**.

Implementar la mecánica del juego

Para saber cuándo la pelota choca con un objeto utilizaremos el bloque c**uando spritePelota.EnColisiónCon otro ejecutar**. Este bloque está dentro de la paleta de recursos relacionados con el objeto Pelota1.



Lo que queremos es que desaparezca el objeto contra el que ha chocado la pelota. Podremos hacer referencia a este objeto a través del parámetro **otro**.

Vamos a utilizar bloques ubicados en **Cualquier componente / CualquierSpriteImagen.**

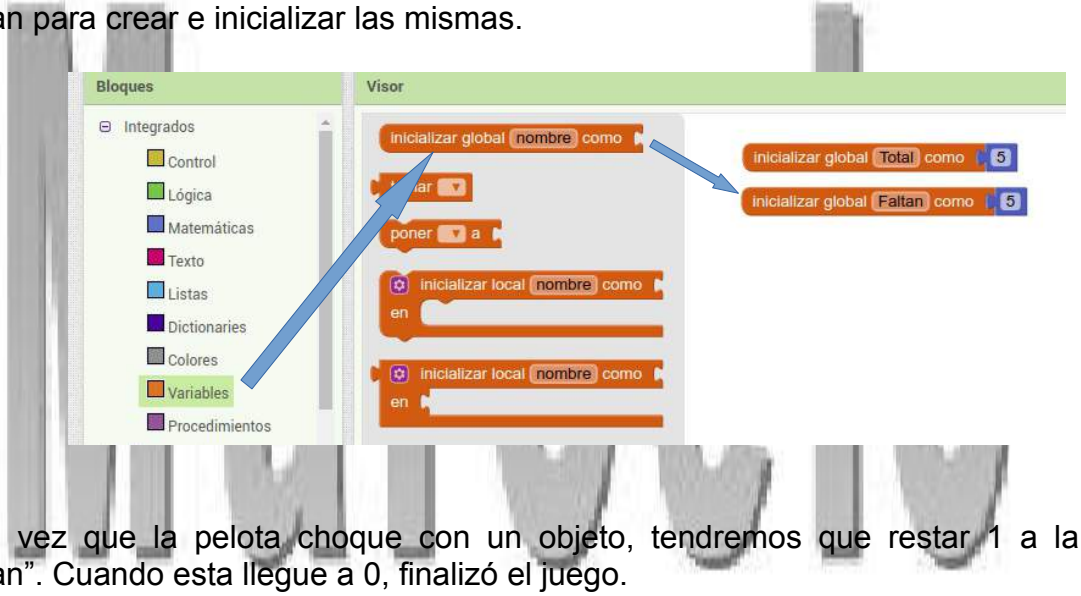
En este caso hará referencia a objetos del tipo SpriteImagen, como son nuestros cinco “enemigos”.

Lo que vamos a hacer es indicarle al programa que debe hacer desaparecer de la pantalla cada objeto cuando la pelota choca con él.
De la opción “**otros**”, en **del componente** ponemos la opción **Tomar otros** y de lógica seleccionamos “**Falso**” en como.

Puntos

En primer lugar definiremos dos variables, una llamada “**Total**” que mostrara el total de enemigos que debemos matar y otra “**Faltan**” que muestra cuantos quedan por matar.

De la paleta variables y matemática seleccionamos lo dos bloques que a continuación se utilizan para crear e inicializar las mismas.

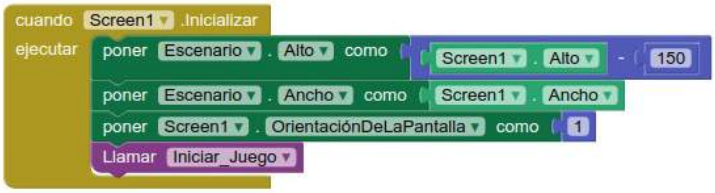


Cada vez que la pelota choque con un objeto, tendremos que restar 1 a la variable "Faltan". Cuando esta llegue a 0, finalizó el juego.
Podemos poner una Etiqueta luego del escenario para verificar si la variable funciona.

Para facilitar el mantenimiento de nuestro programa, y su lectura, vamos a definir dos procedimientos: l**niciar_juego** y **Fin_del_juego**.

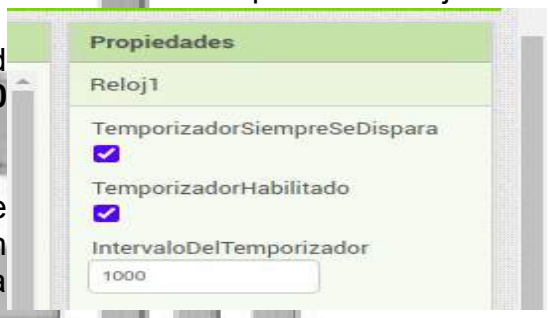
En el primero de ellos tendremos que ubicar nuevamente los objetos, y reiniciar todas las variables que tiene que manejar el juego. Es decir, dejar las cosas listas para empezar a jugar.

Ahora llamaremos al procedimiento "Iniciar_juego" desde el bloque **cuando Screen1.Inicializar ejecutar.**



Tiempo de juego

Tenemos que empezar por crear un objeto Reloj en el Diseñador. El componente Reloj se encuentra dentro de la paleta Sensores.
Le daremos el nombre “Tiempo”, así que su propiedad **IntervaloDelTemporizador** contendrá el **valor 1000** (1000 milisegundos es igual a 1 segundo).



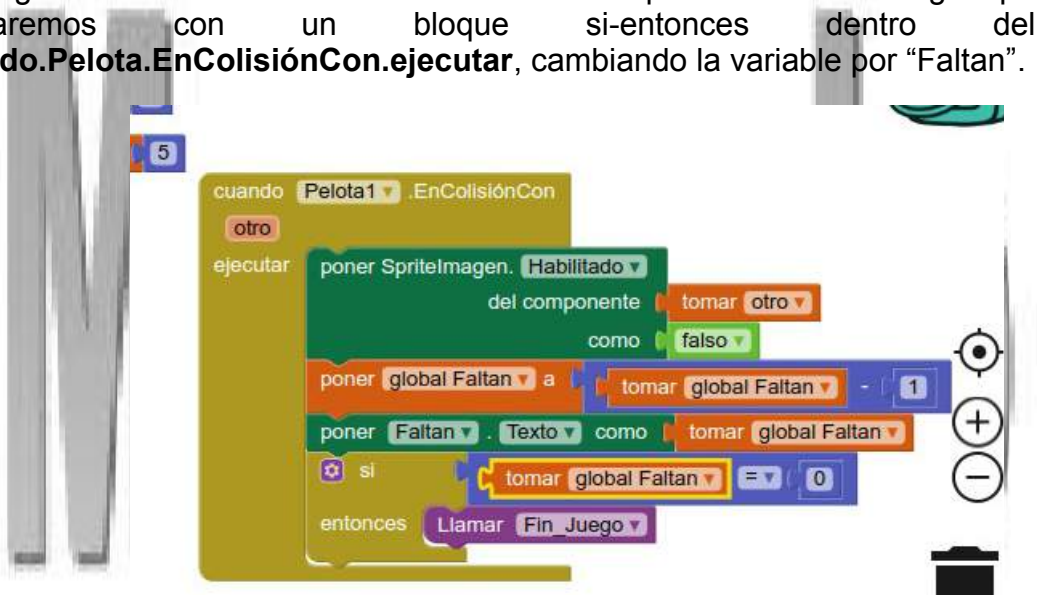
La idea es mostrar siempre en la pantalla el número de segundos restantes, y para eso necesitamos restar un segundo cada vez respecto de la cantidad que queda disponible.
Crearemos una variable llamada “**Tiempo**”, una etiqueta llamada “**Tiempo_restante**” y la programación quedará de la siguiente manera:

Inicializaremos la variable Tiempo en 30 segundos.

Fin del juego

Vamos a añadir un bloque si-entonces dentro del bloque **cuando.Tiempo.Temporizador ejecutar.** Cuando se cumpla la condición del si-entonces habrá llegado el momento de hacer una llamada al procedimiento **Fin_del_juego**. Esto será cuando los segundos lleguen a cero.

El juego también deberá terminar cuando no queden más enemigos por matar. Lo indicaremos con un bloque si-entonces dentro del bloque **cuando.Pelota.EnColisiónCon.ejecutar**, cambiando la variable por “Faltan”.



Tenemos entonces que definir qué hacer cuando el juego termine. Una opción sencilla y clara puede ser informar al usuario a través de un texto, y tal vez un sonido.

Podemos escribir el texto “Fin” en la etiqueta **Tiempo_restante**, usando un bloque “” de la paleta texto entro del procedimiento “**Fin_juego**”.

Veremos que el texto aparece, pero sólo durante un segundo. Esto se debe a que el programa comprueba en el bloque **cuando.Segundo.Temporizador ejecutar** si la variable Segundos_totales vale 0, pero esto sólo se cumple durante un segundo, porque el tiempo sigue contando, así que rápidamente el valor de la variable **Tiempo** pasa a ser -1, -2.... El programa sigue ejecutándose, y en la siguiente ejecución del bloque **cuando.Tiemo.Temporizador ejecutar** volverá a escribir el valor de la variable “Tiempos” encima del texto.

Para evitar esto tenemos que hacer que el bloque **cuando.Tiempo.Temporizador ejecutar** deje de ejecutarse. Esto se consigue dándole el valor falso a la propiedad **TemporizadorHabilitado** del componente Tiempo (Reloj).
Para ello vamos al componente Tiempo (Reloj) y arrastramos el bloque Poner.Tiempo. **TemporizadorHabilitado. Como,** y de la paleta “**Lógica**” el bloque **Falso**.

Pero atención, si queremos que el programa funcione correctamente la próxima vez, y ejecute el contenido de **cuando.Tiempo.Temporizador ejecutar**, tendremos que poner el valor de esa propiedad como cierto en el procedimiento **Iniciar_juego**.

Además antes deberemos restablecer el número de segundos restantes a 30 en la variable **Tiempo** y 5 en la variable **Faltan**.

Ahora te propongo un pequeño desafío (Ver archivo del Desafío).