¡Bienvenidos al **Desafío 1** de la Comunidad Online de Club de Apps!

Este tutorial está inspirado en el clásico juego **Space Invaders**, desarrollado en el año 1978, donde objetivo es eliminar oleadas de alienígenas con un cañón láser y obtener la mayor cantidad de puntos posibles. En esta ocasión deberán crear su propia versión del juego mediante el uso de **App Inventor.** El principal objetivo de este desafío es recordar las distintas funcionalidades que tiene App Inventor a través de las componentes de *Diseño y animación* para que puedan incentivar a sus estudiantes a crear sus propios juegos mediante el uso del lienzo.

Qué Construirás:

Para esta versión del juego vamos a implementar las siguientes funcionalidades:

1. Animar las distintas naves que tendrá el juego.
2. Utilizar la interacción con la pantalla para manejar el comportamiento de tu base o nave.
3. Lograr que tu base dispare un misil capaz de derribar naves enemigas.
4. Conseguir puntaje tras derribar naves enemigas.
5. Pausar el juego mediante un botón.
6. Crear una pantalla de bienvenida para empezar el juego.
7. Poder recordar el puntaje más alto.

Qué aprenderás:

Este tutorial cubre los siguientes componentes y conceptos:

1. Uso del componente **Lienzo** que funciona como una superficie libre en la pantalla del teléfono.
2. Uso del componente **SpriteImagen** para realizar animaciones sobre el *Lienzo.*
3. Uso del componente **Reloj** para generar la animación del juego y mover los distintos *Sprites*
4. Uso del componente **Sonido** para emitir vibraciones mediante el teléfono.
5. Uso del componente **Botón** para poder pausar el juego.
6. Uso de bloques ***matemáticos*** para crear incrementos y restas.
7. Uso del bloque matemático **Módulo** para identificar ciclos y períodos de tiempo.
8. Uso de una **Screen** extra para crear una pantalla de bienvenida
9. Uso de ***procedimientos*** para implementar comportamientos repetitivos, como el movimiento de las naves.

Para Empezar

Tal como lo han hecho en otras ocasiones, lo primero que harán será definir la interfaz gráfica del juego, de esa manera separaremos el desarrollo en dos partes. La primera, que estará enfocada principalmente en el diseño que tendrá la aplicación y la segunda etapa, donde programarán el comportamiento del juego.

Diseñando los componentes:

Para que el juego no sólo funcione bien sino que además se vea completo, deberán utilizar los siguientes componentes desde el Diseñador:

- Un **Screen** que servirá como pantalla de bienvenida y otra **Screen** donde encontrarás todos los componentes del juego.
- Un **Lienzo** que será el campo de juego.
- Distintos **SpritesImagen** que permitirán animar tanto las naves enemigas como tu base.
- Una **Pelota** que servirá para derribar naves enemigas.
- Un **Reloj** para hacer que las naves enemigas se muevan cada cierto periodo de tiempo.
- **Etiquetas** que permitirán mostrar tu puntaje.
- Un **Botón** que te permitirá pausar el juego.

Creando una pantalla de bienvenida:

Lo primero que harán será crear una pantalla de bienvenida sencilla que tendrá una imagen de fondo y un botón que permitirá comenzar el juego. App Inventor considera como la ventana principal al *Screen1* lo que significa que independiente de la cantidad de ventanas que tenga el proyecto, cuando crees el ***.apk*** de la aplicación, siempre se ejecutará el *Screen1* como primera ventana. Es por eso que debes usar esta ventana como pantalla de bienvenida.

El diseño que tendrá esta pantalla dependerá de qué tan creativos seas a la hora de escoger una imagen de fondo para la ventana y de cómo posicionar los distintos botones en ella. Para crear nuestra ventana de bienvenida haremos lo siguiente:

- En esta oportunidad vas a utilizar un solo botón. Arrastra desde la *Paleta* un botón y suéltalo en el celular que se ve en el visor.
- Pincha el componente *Screen1* y desde las propiedades, carga una imagen de fondo y

seleccionala como *ImagenDeFondo.*

- Cambia la propiedad *DispHorizontal* y *DispVertical* del *Screen1* dejando ambos valores como *centro*. Verás que tu botón ahora se encuentra al medio de la pantalla.
- Cambia la propiedad *texto* del *Botón* para que muestre el mensaje *"Comenzar a Jugar"*

Si has seguido todas las instrucciones, tu aplicación debería lucir tal como se muestra en la figura 1.1:

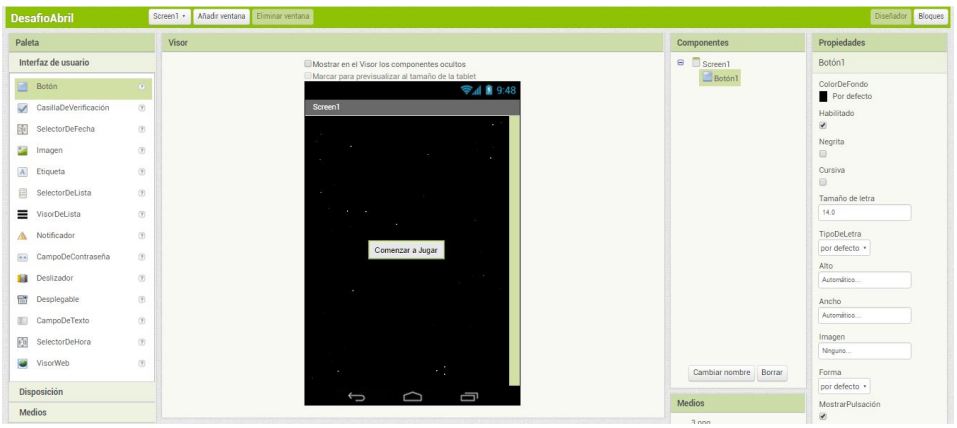

Figura 1.1: Interfaz de Bienvenida

Recuerda que puedes configurar tu pantalla de bienvenida como tú quieras. El siguiente paso será programar el comportamiento de esta ventana. Para eso iremos al editor de bloques donde tendremos un solo evento: ***cuando Botón1.Clic.*** Como lo que buscamos es llamar a un nuevo Screen cuando el *Botón1* sea clickeado, debemos crear una nueva ventana. Para ello deberás hacer click en el botón que dice "*Añadir Ventana"* que se encuentra justo arriba del *Visor,* dejando como nombre de la ventana, *Screen2*. Posterior a esto, desde los bloques de tu *Screen1* programaremos el evento para el botón que hemos agregado:

- Arrastra el bloque ***cuando Botón1.Clic*** y suéltalo en el *Visor*
- Desde la sección *Control*, arrastra el bloque abrir **otra pantalla Nombre de la pantalla** y suéltalo en el interior del bloque de evento del botón. Selecciona desde *Texto* el nombre del *Screen* como *Screen2*.

Los bloques deberían lucir como en la Figura 1.2. El siguiente paso será programar nuestro juego.

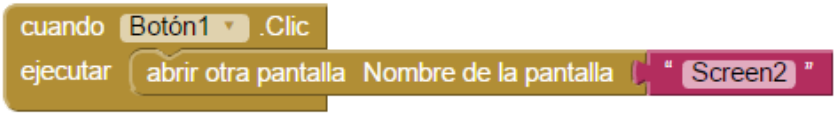

Figura 1.2: Bloques del Screen1

Agregar el Lienzo

El principal componente que usarás para esta aplicación será el **Lienzo**. Sobre él agregarás los distintos sprites que representarán las distintas naves y la barra que controlará el disparo de la propia nave o base.

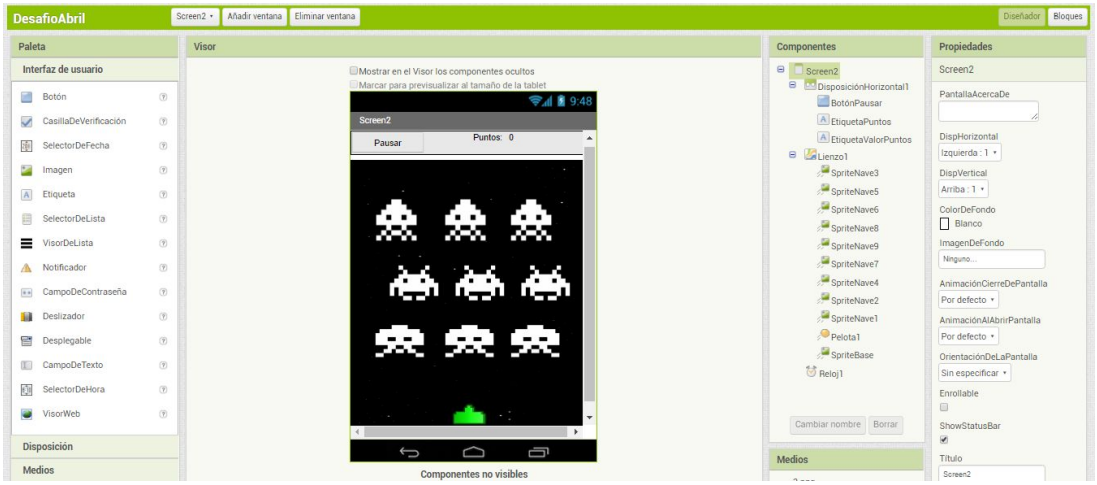

Figura 1.3 : Aspecto visual del Screen2

Para hacer que el Screen2 se vea como en la figura 1.3 debes hacer los siguientes pasos:

1. Primero, agrega un lienzo a la aplicación. Para ello, debes ir a la ***Paleta*** y desde el apartado ***Dibujo y animación*** arrastrar un lienzo hacia el visor.
2. Justo antes del lienzo, agregas una disposición horizontal que tendrá en su interior un **Botón** y **dos etiquetas.** El botón permitirá pausar el juego mientras que las etiquetas permitirán mostrar el puntaje del jugador.
3. Escoge un fondo para el lienzo cambiando su propiedad ***Imagen de Fondo***.
4. Por último, ajusta desde las propiedades el ancho y el alto del lienzo seleccionando como valor ***ajustar al contenedor***.

El siguiente paso será agregar los componentes necesarios para que funcione el juego. A continuación anota lo que necesites agregar para que el juego comience a tomar forma:

1. Necesitarás del componente ***SpriteImagen*** para crear las distintas naves del juego. Para este tutorial agrega un número limitado de naves, pero tú puedes agregar la cantidad que deseas. Debes tener en cuenta que mientras más naves agregues más precaución debes tener en los bloques a la hora de controlar su comportamiento. En esta ocasión agregaremos 9 *Sprites* así que debemos tener cuidado a la hora de programar. Arrastra un

componente *SpriteImagen* desde la *Paleta*, en el apartado *Dibujo y animación,* y ponlo en cualquier parte de nuestro *Lienzo*. **Repite el proceso con los 9 *SpriteImagen*.**

2. Ajusta el tamaño de los *Sprites* desde las propiedades si es que son muy grandes. Renombra cada uno de los componente para hacer más comprensible su programación. En nuestro caso hemos nombrado a nuestros *Sprites* como *SpriteNave1*, *SpriteNave2*, ..., y *SpriteBase* (para el caso de nuestra nave).
3. El siguiente paso será ordenar y posicionar nuestros *Sprites*. La idea es conseguir una estructura similar a la de la Figura 1.3. Carga la imagen que va a tener cada uno de los *Sprites* cambiando su propiedad *Foto.* Ordenaremos los *sprites* usando 3 de ellos por fila dejándolos equitativamente en la pantalla.
4. Agregaremos dos componentes extras al *Lienzo.* Un nuevo *SpriteImagen* que representará tu base o nave la que a su vez disparará una ***Pelota.*** Para agregar un *Pelota* debes dirigirte a la *Paleta* y arrastrarla desde la sección *Dibujo y animación.*

Hasta el momento tenemos todos los elementos visuales de nuestra aplicación, sin embargo, aún nos faltan otros componentes que permitirán que nuestra app funcione correctamente:

- Necesitaremos un componente que nos permita controlar cada cuántos segundos se van moviendo las naves. Para eso, arrastraremos desde la *Paleta* un **Reloj** desde la sección *Sensores.* Dado que el *Reloj* no es un componente visual, cuando lo soltemos en el *Visor* quedará en el apartado inferior como un ***Componente no Visible***

Si hemos seguido todos los puntos hasta aquí, ya habremos construido la interfaz gráfica de nuestro juego. Si probamos nuestra aplicación desde el *Mit Ai2 Companion*, veremos que los componentes están situados en la pantalla del dispositivo, por lo tanto, el siguiente paso que debemos hacer es programar el comportamiento de nuestra aplicación.

Agregar comportamiento a los componentes

Desde el ***Editor de Bloques*** implementaremos el comportamiento de nuestra aplicación. Lo que haremos en esta oportunidad será mover las distintas naves cada ciertos segundo con dos movimientos: las naves oscilarán de izquierda a derecha cada 1 segundo y descenderán en bloque por la pantalla cada 5 segundos. Si las naves enemigas llegan a tocar a nuestra propia base, el juego termina

Partiremos programando los bloque que permiten controlar nuestra base para que dispare la pelotita hacia las naves enemigas. Nuestra base podrá moverse solo hacia la derecha o

izquierda de la pantalla pero no podrá hacerlo de manera vertical. Además, cada vez que nuestra base se mueva y llegue a su posición de destino, deberá lanzar esta pelotita. También programaremos la opción de disparar cuando toquemos la base (en caso de que no queramos mover la base y necesitemos disparar desde una posición fija). El objetivo del juego es eliminar cada una de las naves enemiga antes que éstas impacten nuestra base.

Mover nuestra base

Dado que nuestra base es en realidad un *SpriteImagen* podemos hacer que ésta se desplace libremente por la pantalla. En estricto rigor lo que haremos será programar nuestra base para que se mueva sólo de manera horizontal. Para ello, desde el *Editor de Bloques* usaremos el controlador de eventos ***cuando SpriteBase.Arrastrado*** que indica si un sprite fue arrastrado o no por la pantalla entregando como resultado las coordenadas por donde fue desplazado el sprite, y le incluiremos el bloque ***llamar SpriteBase.MoverA,*** que nos servirá para establecer el movimiento horizontal de nuestra nave. La figura 1.4 muestra como deberían verse los bloques:

Figura 1.4: Movimiento horizontal de nuestra base.

Con el movimiento de nuestra base listo, el siguiente paso será hacer que ésta dispare la pelota cada vez que la toquemos. Dado que para arrastrar nuestro *Sprite* debemos necesariamente tocarlo, la pelotita también será lanzada en este caso. La siguiente figura muestra cómo quedará el bloque que vamos a construir:

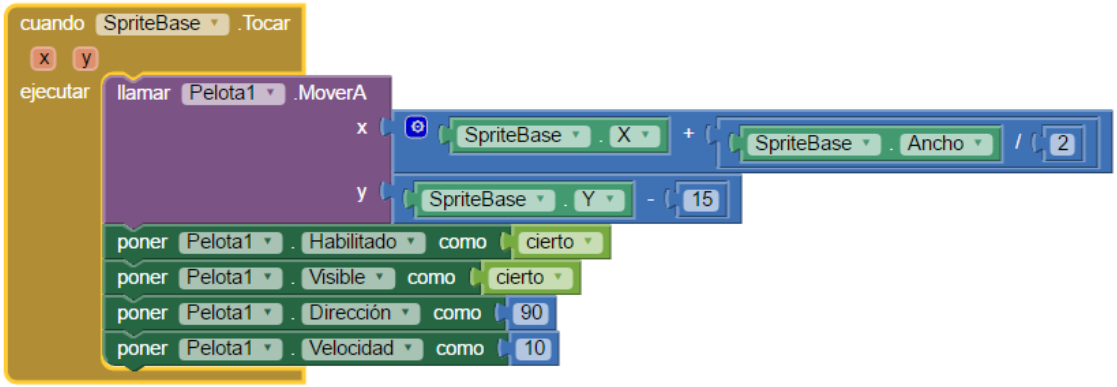

Figura 1.5: Bloques para el disparo de la pelotita

Si nuestra base es tocada, la *pelotita* será lanzada hacia arriba de la pantalla. La idea es usar la *pelotita* para derribar las naves enemiga. Para conseguir este comportamiento, crearemos los bloques de la siguiente manera:

1. Arrastra el bloque ***SpriteBase.Tocar***
2. Dirígete hacia los bloques de la *pelota* y arrastra el que dice ***Pelota1.MoverA*** . Encájalo en el interior de nuestro bloque ***SpriteBase.Tocar***.
3. Para hacer nuestro juego visualmente atractivo, haremos que la pelotita a la hora de ser lanzada se posicione fuera de nuestra base. Para conseguir ese objetivo posicionaremos la pelotita al centro de nuestra base mediante los siguientes bloques:
    a. Definiremos la posición ***X*** como la suma (desde *matemáticas*) entre la posición horizontal actual de nuestra base (***SpriteBase.X***) y la mitad del ancho de nuestra base (***SpriteBase.Ancho***). De esa forma nuestra pelotita quedará justo al medio de la base al momento de ser lanzada.
    b. Definiremos la posición ***Y*** como la posición vertical actual de nuestra base ***(SpriteBase.Y)*** menos un valor que permita que la pelotita queda justo arriba. En este caso usaremos un valor de ***15***
4. El último paso será definir las características que tendrá nuestra pelota a la hora de ser lanzada, para eso agregaremos los siguientes bloques los que encajaremos en ***SpriteBase.Tocar***:
    a. Desde los bloques de nuestra pelota, arrastraremos el bloque **poner** ***Pelota1.Habilitado*** y encajaremos un bloque ***cierto*** que obtendremos desde ***Lógica.***
    b. Arrastraremos el bloque poner ***Pelota1.Visible como*** y lo dejaremos con el valor *cierto,* tal cual lo hicimos con la propiedad habilitado
    c. Arrastraremos el bloque poner ***Pelota1.Dirección como*** y desde ***Matemáticas*** le asignaremos un 90. Esa es la forma de hacer que nuestra pelota salga lanzada hacia arriba de la pantalla.
    d. Por último, arrastraremos el bloque poner ***Pelota1.Velocidad*** y la dejaremos como *10*. No olvides que puedes elegir la velocidad a tu gusto.

Hasta este punto, habremos programado todo lo necesario para que nuestra base se mueva y dispare. El siguiente paso será programar las naves enemigas.

Moviendo las naves enemigas

Para mover las naves estableceremos el criterio que nosotros queramos. Éstas podrían

aparecer al azar por la pantalla o moverse sin un patrón definido. Sin embargo, en esta oportunidad las naves se moverán de manera uniforme. El movimiento será de dos manera distinta, teniendo un pequeño movimiento de forma horizontal (las naves se desplazarán a la izquierda o la derecha y luego volverán a su posición anterior) y de manera vertical (cada cierto tiempo las naves descenderán por la pantalla). Para poder lograr este movimiento el primer paso que haremos será crear un procedimiento **moverNavesEnemigas.** Los pasos para crear este procedimiento serán descritos a continuación:

Procedimiento MoverNavesEnemigas:

- Selecciona la sección *Procedimientos* desde el *Editor de Bloques.*
- Arrastra al espacio de trabajo el bloque **como Procedimiento ejecutar** (debes tener cuidado de no arrastrar el *bloque como procedimiento resultado*)
- El procedimiento que crearemos necesita conocer cuánto vamos a mover la naves enemigas, tanto horizontal como verticalmente. Es por eso que debemos componer nuestro bloque de tal manera que acepte dos *parámetros* de entrada. Para ello desde el engranaje azul que tiene nuestro bloque agregaremos dos entradas las que denominaremos *distanciaX* y *distanciaY.*
- Dado que tenemos 9 naves en nuestro juego, el siguiente paso lo repetiremos nueve veces, uno por cada nave. Vamos a mover cada una de las naves enemigas según el valor que definamos en *distanciaX* y *distanciaY.* Para ello, iremos al *SpriteNave1* y arrastraremos el bloque ***SpriteNave1.MoverA,*** que completaremos de la siguiente manera:
    - Definiremos **X** como la suma de de la distancia actual de la nave más (+) cuánto queremos moverla horizontalmente, determinado por el valor que tenga *distanciaX*. Finalmente nuestro bloque quedará de la siguiente manera:
    - Repetimos el mismo proceso para **Y**

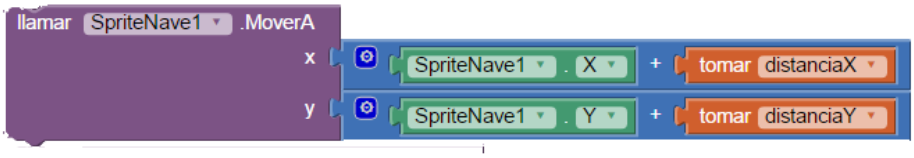

Figura 1.6: determinando el movimiento de las naves enemigas

- Una vez que tengamos este bloque, lo encajaremos dentro de nuestro procedimiento. El siguiente paso es repetir el procedimiento por cada una de las naves (sí, puede resultar un paso algo tedioso pero no hay que olvidar que debemos mover todas las naves). El bloque a continuación muestra cómo debería quedar nuestro procedimiento. **¡Importante!**, por temas de visualización, sólo se mostrarán los dos primeros bloques además del último. El resto estarán ocultos pero tienen la misma estructura de los

bloques que se ven en la figura 1.7.

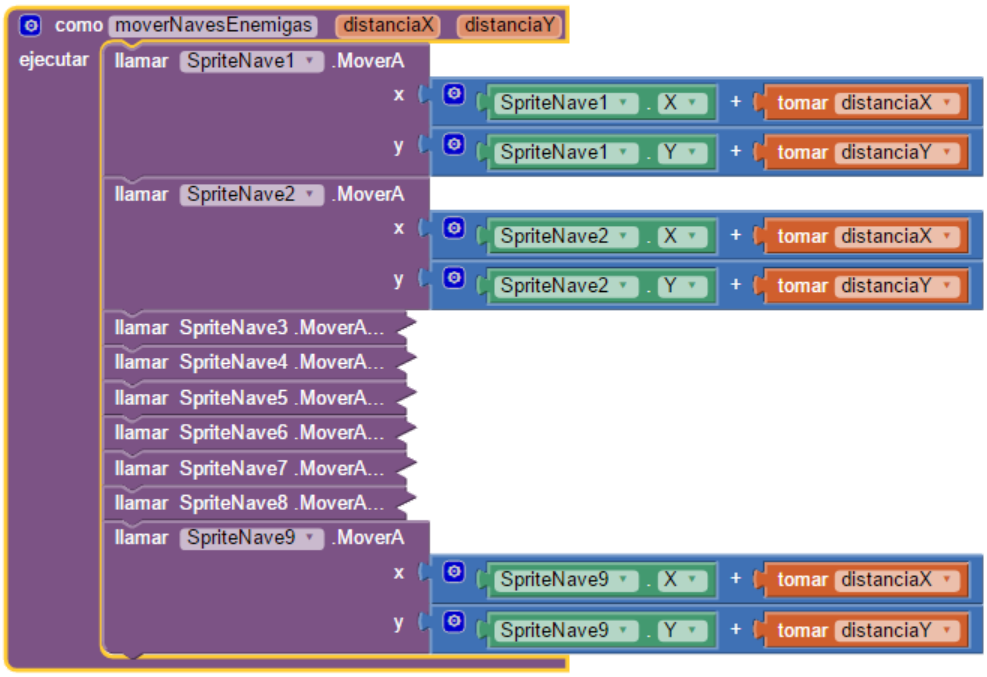

Figura 1.7: procedimiento para definir el movimiento de las naves

Si has logrado seguir todos los pasos, te habrás dado cuenta que hemos creado el procedimiento que mueve las naves enemigas, sin embargo no hemo definido en qué punto vamos a mover estas naves. Para ello, utilizaremos el componente **Reloj** que agregamos anteriormente desde el diseñador. Estableceremos que las bases enemigas se moverán cuando haya ocurrido 1 segundo del juego y cuando se inicializa la pantalla. Además, debemos establecer un tiempo para que las naves desciendan. Dejaremos ese tiempo en 5 segundos. **¡Atención!** Para que los tiempos sean efectivamente de 1 segundo y 5 segundos, el intervalo del temporizador debe estar asignado a 1000 milisegundos. Puedes variar el intervalo como desees, sobre todo si deseas hacer el juego más difícil.

1. Como queremos hacer que las naves se muevan cada cierto intervalo de tiempo, necesitaremos del *Reloj*. El primer paso será arrastrar el bloque ***Cuando Reloj1.Temporizador***. Esto permite que cada cierto tiempo (el que está definido en el intervalo) podamos realizar ciertas acciones.
2. Para simular que están pasando segundos, debemos tener una variable que se incremente con cada *tick* del reloj. Para ello, desde la sección variables crearemos una que llamaremos ***Segundos*** y la vamos a iniciar en cero, tal cual lo muestra la imagen a continuación:

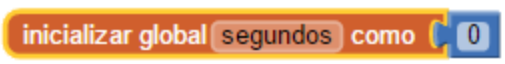

Figura 1.8: Variable global para ajustar el tiempo del reloj

3. Incrementaremos nuestra variable *segundos* de uno en uno según el intervalo que tenga nuestro reloj. Para ello desde la misma variable obtendremos el bloque ***poner GlobalSegundosA*** el que encajaremos con una suma entre el valor actual de los segundos más (+) uno.
4. Tal como mencionamos anteriormente, definiremos que cada 5 segundos la naves se desplacen hacia abajo por la pantalla. Para determinar si han pasado 5 segundos debemos revisar que el valor de nuestra variable *segundos* sea divisible por 5. Un número es múltiplo de otro si el resto de su división entera es 0. Para lograr dicho comportamiento haremos lo siguiente:
    a. Desde la sección ***Control*** arrastraremos un bloque de decisión ***si-entonces*** el que encajaremos dentro de nuestro *Reloj1.Temporizador.* Como debemos preguntarnos si han pasado 5 segundos, nuestra condición será determinar si el resto de dividir nuestra variable por 5 es 0. Esa condición la vamos a construir desde matemáticas mediante el bloque módulo.
    b. Una vez que hayamos construido esta condición, el siguiente paso será mover nuestras naves. Este movimiento solo ocurrirá si se cumple la condición, y de ser así llamaremos al procedimiento que creamos anteriormente y lo encajaremos dentro de nuestro bloque ***si-entonces***. Dado que al momento de crear nuestro procedimiento *MoverNavesEnemigas* definimos la necesidad de contar con los valores de como desplazarse, tanto vertical como horizontalmente, estamos obligados a entregar estos valores al momento de llamar a nuestro procedimiento. Como estamos moviendo nuestra nave solamente de manera vertical, diremos que el valor de *distanciaX* es 0 y que el valor de *distanciaY* es 10 (o cualquier número positivo de pixeles que desees. Debes tener cuidado con usar números muy grande ya que podrían hacer que en poco tiempo las naves enemigas lleguen al final de la pantalla).
5. Debemos repetir exactamente el mismo ejercicio pero esta vez logrando que las naves se muevan de manera horizontal:
    a. Desde la sección Control arrastraremos un bloque *Si-Entonces* el que encajaremos dentro de nuestro evento ***Reloj1.Temporizador***.
    b. Haremos que las naves tengan un movimiento muy sencillo. Si ha pasado un segundo, todas las naves se moverán 20 píxeles a la derecha. Luego, cuando haya pasado otro segundo, las naves volverán a la posición anterior (restando los 20 píxeles que nos habíamos movido). Para lograr este movimiento, debemos hacer el mismo chequeo de los segundos que hicimos en el bloque anterior cuando queríamos mover la nave de manera vertical. La diferencia es que ahora nos importa si ha pasado 1 o 2 segundos. Para conseguir esto vamos a trabajar con el

valor del resto que se obtiene de dividir nuestra variable *segundos* en dos. Si el resto es 0 quiere decir que ha pasado una cantidad par de segundos mientras que si el resto es 1, habrá pasado una cantidad impar. Nuestra condición corresponde a ese chequeo. Si ha pasado una cantidad par de segundos, entonces moveremos las naves 20 píxeles a la derecha, sino, moveremos la nave a su posición original.

La Figura 1.9 muestra como se verían los bloques una vez que completes este paso.

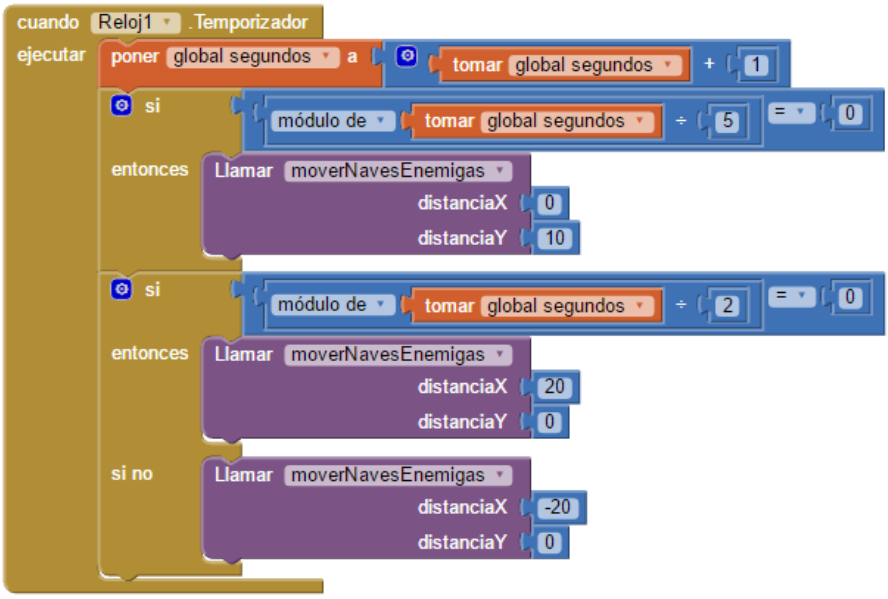

Figura 1.9: Movimiento de las naves según segundos del reloj.

Con estos bloques, hemos establecido el movimiento principal de las naves enemigas. Consideraremos un último detalle respecto al movimiento. Definiremos el movimiento inicial arrastrando desde la sección *Screen2* el evento ***CuandoScreen2.Inicializar*** y encajaremos un ***llamar MoverNavesEnemigas*** entregando como entrada definiremos el X e Y de la posición a la que deseamos mover las naves. Además, dejaremos nuestra pelotita deshabilitada al inicializar la pantalla evitando de esa forma evitamos que salga disparada a penas se inicie nuestra aplicación.

Figura 1.10: Movimiento inicial de las naves

Hasta este punto ya hemos programado el movimiento y disparo de nuestra base además

del movimiento de las naves enemigas. El siguiente paso será hacer que las naves enemigas caigan si reciben un impacto desde nuestra base.

Interactuando con las naves enemigas

Como el objetivo del juego es lograr eliminar las naves enemigas antes que éstas lleguen a nuestra base, debemos programar ese comportamiento en este momento. Hasta ahora, si la *pelotita* impacta a las naves no ocurre nada ya que atravesará las naves porque no estamos detectando la colisión. Debemos no sólo ser capaces de identificar si efectivamente la *pelotita* colisiona con una de las naves enemigas sino que también debemos saber con cuál de estas naves colisionó. Es importante conocer cuál fue la nave colisionada porque debemos sacarla del juego. Si tuviésemos pocos *sprite* en nuestro *Lienzo*, probablemente este paso resultaría muy rápido de programar, sin embargo, como tenemos 9 *Sprites*, hay que hacer una revisión una a una de las colisiones. Este paso se puede hacer de más de una forma, por lo que si se te viene otra solución a la cabeza, puedes intentar programarla y ver si te funciona. Para este caso, vamos a aprender a utilizar una nueva variante de nuestro bloque de control *Si-Entonces:* ***el Si-Entonces Sino, si***.

Cuando nos hacemos preguntas, no siempre ocurrirá que las respuestas tienen solo dos caminos. Hasta ahora siempre hemos trabajado con condiciones que implican sólo dos caminos; si la condición se cumple entonces seguimos el *camino1* pero sino se cumple, seguimos el *camino2*. Si pensamos en situaciones de nuestra vida cotidiana, nos daremos cuenta que no siempre pasa que tenemos sólo dos caminos ante una pregunta. Lo mismo ocurre cuando programamos. Hay veces que necesitamos preguntar muchas veces junto con programar qué es lo que quiero que ocurra en cada una de esas preguntas que hice. En el caso de nuestro juego, como queremos identificar cuál fue la nave que nuestra pelotita impactó, nuestra condición estará compuesta por más de una pregunta. En estricto rigor, lo que debemos hacer es preguntarnos, una a una, cuál fue la nave impactada por nuestra *pelotita*. La figura a continuación muestran los bloques necesarios para lograr este comportamiento (**¡Importante!**, la imagen muestra solo 4 Sprites a modo de ejemplo. En nuestro ejercicio debe ser con 9).

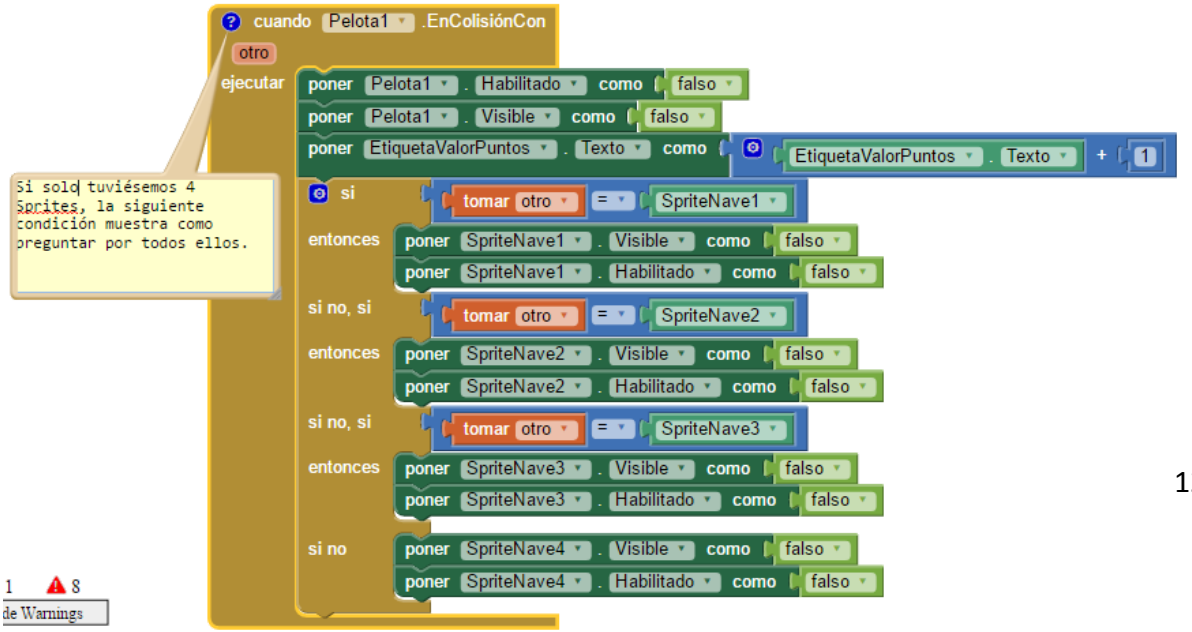

imagen Figura 1.11: Detectando la colisión de la pelotita

Como podemos ver en la imagen anterior, utilizaremos el evento **cuando *Pelota1.EnColosiónCon*** que nos dice por medio de la variable *otro,* con qué otro *Sprite* ha colisionado la pelota. En caso de que la pelota haya colisionado con un *Sprite* lo que debemos hacer es muy sencillo.

- Deshabilitaremos la pelota para que ésta no pueda impactar a ningún otro *Sprite.* Por la misma razón la dejaremos invisible.
- Sumaremos un punto y lo escribiremos en la etiqueta que hemos diseñado para eso (recuerda que debemos ir sumando los puntos uno a uno).
- Debemos identificar cuál fue el *Sprite* impactado para poder ser borrado del juego. Para ello debemos utilizar la variable *otro* y compararla con cada uno de los *Sprites* en juego. Cuando el valor de la condición sea verdadero, quiere decir que hemos detectado cuál fue el *Sprite* con el que colisionó la pelota por lo que podremos deshabilitarlo y dejarlo invisible. Para obtener el bloque *SpriteNave1, SpriteNave2,...,* debes arrastrarlo desde los bloques asociados a cada *Sprite.*

Si pruebas tu aplicación en este momento, podrás darte cuenta que funciona sin problemas. Si presionas tu base ésta disparará la pelotita que si impacta a una de las naves enemigas la va a eliminar. Lo mismo ocurre si arrastras tu nave de manera horizontal que disparará la pelotita cuando dejes de moverla. El último comportamiento que debemos programar es el de detectar el final del juego, el que será gatillado cuando algunas de la naves enemigas impacte tu base. A diferencia de la colisión de la pelotita sobre las naves enemigas, no es necesario identificar cuál fue la nave que impactó tu base ya que bastaría con que cualquiera de ellas lo hiciera para perder el juego. Los pasos para agregar este comportamiento son los siguientes:

1. Arrastra el bloque ***SpriteBase.EnColisiónCon*** el que nos permitirá saber si nuestra base fue o no impactada por algún otro *Sprite.* Como no nos interesa saber cuál fue el Sprite que la impactó, no es necesario usar las condiciones que usamos cuando identificamos a cuál nave enemiga impactó nuestra pelota.
2. El siguiente paso es deshabilitar el *Reloj*, de esa forma las naves enemigas no seguirán moviéndose y podrás cambiar de pantalla rápidamente y sin dificultad.
3. Agrega desde Control el bloque ***abrir otra pantalla Nombre de la pantalla*** e indicale que volveremos a nuestra *Screen1*

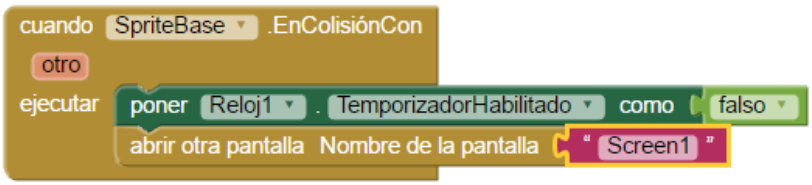

Figura 1.12: Detectando la colisión para el fin del juego

Con este paso habremos construido el esqueleto de nuestra aplicación el que funcionará sin problemas, sin embargo aún faltan ciertos detalles que te tocarán programar a ti junto a tu **Club de Apps** para que esta aplicación se vea 100% terminada.

Desafíos para tu club de Apps:

- Si deseas realizar alguna actividad distinta mientras estás jugando probablemente necesitarás pausar el juego. Tener un botón de pausa es algo que tienen la gran mayoría de los juegos (incluso los juegos en línea). Implementa los bloques que necesita el botón "pausar" para que al clickearlo el juego quede en pausa. Si vuelves a presionar este botón el juego deberá reanudarse.
- Otro aspecto muy importante que tienen los juegos es establecer un sistema de puntuación. Hasta el momento nuestro juego presenta una puntuación muy simple, entregando 1 punto por cada nave eliminada. Eso no lo hace muy dinámico. Tampoco podríamos saber qué jugador ha tenido mejor desempeño ya que cualquier usuario que haya eliminado las naves tendrá el mismo puntaje. Junto a tu club de apps inventen un sistema de puntuación que permita considerar el desempeño del jugador en base al tiempo que lleva jugando. Consideren el puntaje final del jugador en base a qué tan rápido eliminó las naves. Usen su creatividad en este punto.
- ¿Qué pasaría si queremos saber quién ha sido el mejor jugador? Implementen una forma que permita recordar quien ha tenido el mejor desempeño en el juego. La idea es que este puntaje se mantenga incluso aunque se haya cerrado la aplicación. Usa la pantalla de bienvenida para entregar la opción de consultar el mejor puntaje.
- Junto a su Club de Apps, personalicen el juego a gusto. Consideren la opción de agregar sonidos cuando impactas una nave enemiga, o cuando hayas perdido el juego. Si lo desean, agreguen un sonido de fondo junto a la opción de silenciar el juego si el usuario lo quisiese hacer. Consideren utilizar la vibración de teléfono para darle más ambiente al juego.
- En el esqueleto del juego que hemos construido hasta ahora, si la pelota no impacta ninguna nave, ésta quedará en la pantalla superior esperando a que vuelvas a disparar. Corrijan este comportamiento ocultando la pelota si es que esta ha tocado el borde superior de la pantalla.