

## 2. Gamificación (continuación)

- [Gamificación \(continuación\)](#)
- [La idea del juego ampliada](#)
- [Movimiento de la letra](#)
- [Movimiento del lápiz](#)
- [Movimiento del disparo](#)
- [Colisión del disparo con un objeto](#)
- [Últimos ajustes](#)
- [Resultado final](#)

# Gamificación (continuación)

Es este bloque de contenidos se plantea un reto mayor: continuar mejorando el juego que ya tenemos creado. Va a requerir añadir más complejidad en los programas de los objetos, ¡pero nada que no esté ya a vuestro alcance! Espero que os sintáis con ganas de continuar con este bloque de contenidos que se sale de la programación planteada en el curso de Aularagon.

El resultado de lo que vamos a conseguir lo puedes ver aquí:

[Ver resultado final](#)

<https://scratch.mit.edu/projects/embed/125282917/?autostart=false>

# La idea del juego ampliada

Vamos a programar un nuevo juego con Scratch, que va a ser una ampliación del anterior. La idea de este juego es la del típico matamarcianos. Pero en nuestro caso, lo que vamos a hacer es disparar pintura con un lápiz a unas letras. Las letras se moverán para evitar ser alcanzadas. El objetivo final será acertar a disparar a todas las letras.

Empezaremos montando los objetos que participarán en nuestro juego. Scratch tiene en su biblioteca de imágenes: letras, lapicero, rayo. Con estas tres imágenes ya tenemos lo necesario para montar nuestros objetos.

Sin embargo, con idea de hacer el juego algo más original, podemos coger dibujos de Internet para montar el juego, o incluso crear con alguna herramienta de dibujo nuestros propios objetos. Para el juego que vamos a trabajar, vamos a utilizar los siguientes dibujos. Descárgatelos a tu ordenador (botón derecho - guardar imagen como) porque vamos a montar el juego con ellos.



Creación propia, utilizando un editor de textos (OpenOffice Writer), simplemente añadiendo un rectángulo y una letra dentro, y luego capturando la pantalla.



Creación propia, utilizando un editor de textos (OpenOffice Writer), simplemente añadiendo un rectángulo y una letra dentro, y luego capturando la pantalla.



Creación propia, utilizando un editor de textos (OpenOffice Writer), simplemente añadiendo un rectángulo y una letra dentro, y luego capturando la pantalla.



Fuente: <https://openclipart.org/detail/29133/pencil> El lapicero está rotado con Gimp.



Creación propia, utilizando un programa de edición de imágenes (Gimp)



Fuente: <https://openclipart.org/detail/49363/blackboard>

## Caso práctico: incluir los objetos de nuestro juego

Empezamos a montar el entorno de objetos de nuestro juego.

Añade las letras A, B y C, el lápiz y el disparo como objetos de Scratch. Revisa que los nombres de objetos de las letras A, B y C se llamen a, b, y c, (botón dcho sobre el objeto - info) porque usaremos estos nombres posteriormente.

Añade la pizarra como fondo.

Ponle un nombre al proyecto: LAPICERO TRAVIESO

### Solución

Te tienes que quedar así:



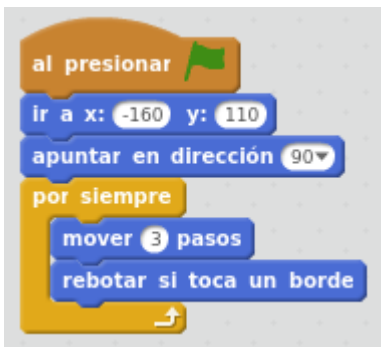
# Movimiento de la letra

## Caso práctico

Vamos a hacer que la letra A se mueva. La letra A debe moverse de izquierda a derecha cambiando de dirección cuando llegue al borde. Esto ya lo hemos hecho en el juego anterior.

Al presionar bandera, posicionamos la letra en un lugar fijo y apuntando a la derecha. A continuación haremos que, de forma indefinida, la letra se mueva 3 pasos, y cuando llegue al borde de la pantalla, haremos que cambie de dirección.

### Solución



# Movimiento del lápiz

## Caso práctico: el lápiz se mueve con las teclas izquierda y derecha

Queremos que el lápiz se mueva a la izquierda cuando presionemos la flecha izquierda. Y que se mueva a la derecha cuando presionemos la flecha derecha.

Esto ya lo hemos hecho en el juego anterior. Sin embargo, si somos "exquisitos", habrás observado que si mantienes pulsada una tecla de flecha izquierda o derecha, debido a que hay un retardo en el teclado antes de empezar a mandar la señal de repetición a nuestro programa, se produce también un retardo en el primer movimiento del lápiz.

Vamos a construir una solución que evita este inconveniente con la siguiente lógica: Al presionar Bandera, estaremos comprobando de forma indefinida si hay alguna tecla presionada. Moveremos el lápiz 10 pasos a la derecha si se presiona la tecla "flecha derecha", y moveremos el lápiz 10 pasos a la izquierda si se presiona la tecla "flecha izquierda".

Empezamos a construir la solución paso a paso. En primer lugar, vamos a hacer que al presionar Bandera, haremos que si la tecla "flecha derecha" está presionada, mover el lápiz 10 pasos a la derecha

### Solución

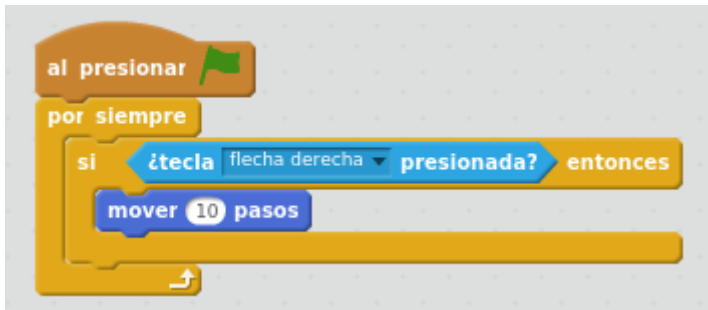


Pero esto no funciona. porque estos bloques sólo se arrancan una vez. Y necesitamos que el programa esté constantemente comprobando si la tecla está presionada. Por lo tanto, tenemos que poner todo dentro el bloque de Control "por siempre", para que el programa esté constantemente comprobando si la tecla está presionada. Ahora ya habremos conseguido que el lápiz se mueva a



la derecha cuando presionemos la tecla de flecha derecha,

## Solución



Lo único que queda es añadir la comprobación de si la tecla de flecha izquierda esté presionada, se mueva el lápiz 10 pasos a la izquierda.

## Solución





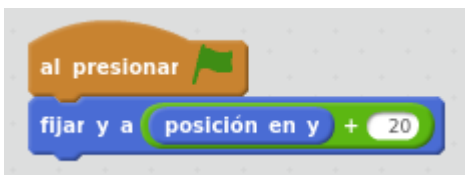
# Movimiento del disparo

## Caso práctico: disparar al presionar espacio

Vamos a hacer que cuando se presione la tecla espacio, el disparo de pintura salga del lápiz y empiece a moverse hacia arriba hasta que llegue al borde superior.

Empezamos haciendo que el disparo se mueva en 20 pasos hacia arriba: Fijaremos la posición Y del objeto disparo sumándole 20 a su posición actual. Utilizaremos el bloque de Operadores " + "

### Solución



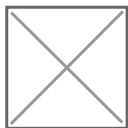
Ahora, el disparo tiene que moverse 20 pasos hacia arriba pero constantemente, hasta que llegue al borde.

### Solución



Ponemos como condición previa a todo esto que se ponga en marcha si se ha presionado la tecla espacio.

## Solución



Esto sólo se arranca una vez, necesitamos meter un bucle para que el programa esté constantemente comprobando si se ha presionado la tecla espacio:



El disparo debe salir del lápiz, por tanto, debemos posicionar el disparo encima del lapicero. posicionaremos el disparo 50 pixeles por encima de la posición del lápiz. ¿En qué momento hay que posicionarlo?: antes del bucle que mueve el disparo hacia arriba.

## Solución



¿Cuándo mostrar y ocultar el disparo?



- Al principio del todo, debemos ocultar el disparo.
- Una vez que tenemos el disparo posicionado encima del lapicero, lo mostramos
- En última instancia, hacemos que desaparezca después de detectar que ha tocado el borde.

## Solución



# Colisión del disparo con un objeto

## Caso práctico: control de la colisión del disparo con una letra

### Prueba

La colisión la vamos a controlar dentro del movimiento del disparo, por lo que continuaremos añadiendo más programación al objeto de disparo. También necesitaremos añadir más programación en las letras. El objetivo es que mientras se está moviendo el disparo, comprobar si colisiona con alguno de los objetos (a, b, c). Si colisiona, quitaremos la letra de la escena, para que no moleste, y sumaremos ¡1 punto al marcador!

Dentro del objeto disparo, si el objeto disparo toca al objeto A, enviaremos el mensaje "impacto-A". Este mensaje habrá que crearlo en el bloque Eventos - "enviar..." - nuevo mensaje.

### Solución

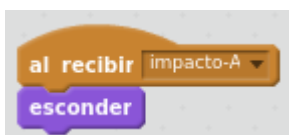


¿Dónde colocaremos estos bloques de programación?: en la programación del objeto disparo, justo después de haberlo movido 20 pasos hacia arriba:



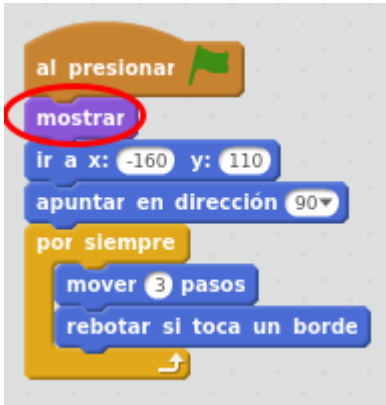
En la programación del objeto A, necesitaremos recoger el mensaje "impacto-A" en el objeto A. Una vez recogido el mensaje, ocultaremos el objeto A.

## Solución



Ahora resulta que cuando el objeto A es impactado, se oculta y ya no se muestra ni aunque iniciemos una nueva partida. La solución es hacer que al inicio del juego, el objeto A se muestre.

## Solución



## Caso práctico: ¡Ahora hay que sumar los puntos del juego!

Tendremos los puntos de juego visibles en la pantalla. Haremos que al colisionar el disparo con un objeto, se sume 1 punto.

Hay que añadir una variable de puntos. Al añadirla, la variable se verá en el escenario de juego, la posicionaremos en la zona superior izquierda.

### Solución



En la programación del objeto A, cada vez que se detecte que el objeto A ha sido colisionado por el disparo, se sumará 1 punto en la puntuación del juego.

## Solución



# Últimos ajustes

## Caso práctico: Programar las letras B y C

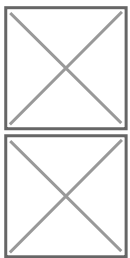
Ahora hay que programar las letras B y C igual que la A para que tenga el mismo comportamiento. Su programación será prácticamente igual a la del objeto A, cambiando el nombre del mensaje y sus coordenadas.

- Para no empezar de cero, es posible arrastrar todos los bloques de programación de la A sobre el objeto B.
- Posteriormente, hay que cambiar datos específicos para objeto B:
- El mensaje será "impacto-B"
- Las coordenadas iniciales para B serán  $x=0$  y  $y=110$

De igual forma a lo realizado en la letra B, replicamos la programación para el objeto C.

- Su mensaje será "impacto-C"
- La posición fija del objeto C puede ser  $x=160$  y  $y=110$

### Solución







**OTRA OPCIÓN:** Podríamos duplicar el objeto A. Posteriormente habría que:

- Añadirle el disfraz de letra B
- Borrarle al objeto el disfraz de letra A.
- Cambiarle el nombre al objeto por "A" (Botón dcho del ratón sobre el objeto B - info)
- Cambiar los datos específicos del objeto B (mensaje "impacto-B", coordenadas -30, 110), y no olvidar añadir la comprobación de colisión en la programación del disparo.

## Caso práctico: Añadir un temporizador

Añadamos un temporizador a nuestro juego.

Crear una variable llamada TIEMPO.

En la programación del Fondo (los fondos también pueden tener sus propios programas), incluiremos un nuevo programa. Al inicio reiniciaremos el cronómetro. En un bucle que se ejecutará siempre, pondremos el valor del cronómetro en la variable TIEMPO y así se visualizará en

pantalla.

## Solución



# Caso práctico: último reto, controlar el fin de juego

Cuando hayamos impactado con todas las letras, mostrar un mensaje en pantalla informando de "Fin de juego".

Una opción sería controlar constantemente el valor de los puntos, y hacer que cuando lleguen a 3, se acabe el juego. Pero otra posible solución algo mejor pensada puede ser controlar en una lista las letras que han sido impactadas. Para ello:

En la programación del lapicero, añadir un nuevo programa, que al iniciar el juego esté comprobando constantemente si hemos impactado las tres letras, y en tal caso mostraremos el mensaje "Fin de juego". Cómo podemos hacerlo:

- Añadiremos un disfraz nuevo al objeto lapiz. Construimos nosotros mismos el disfraz que va a ser el texto "Fin de juego".
- Crear una lista (pestaña Programas - Datos - Crear una lista, que podemos llamar LETRAS).
- Añadimos un bloque de programa en el objeto lapiz, que se inicie al presionar Bandera, donde de forma indefinida comprobaremos si la lista de letras contiene "a", "b" y "c". En tal caso, cambiaremos el disfraz del lapiz por el disfraz de "findejuego" y detendremos todos los programas.
- Tendremos que hacer que al inicio del juego, el disfraz del lapiz sea el del lapicero, para que no se quede fijo el mensaje de "Fin de juego". Y también habrá que ocultar y vaciar la lista.

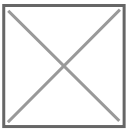
En cada objeto de letra: añadir un valor a la lista cada vez que se impacte una letra.

En el objeto de lapiz: Crear una lista. Ocultarla y vaciarla al inicio del juego.

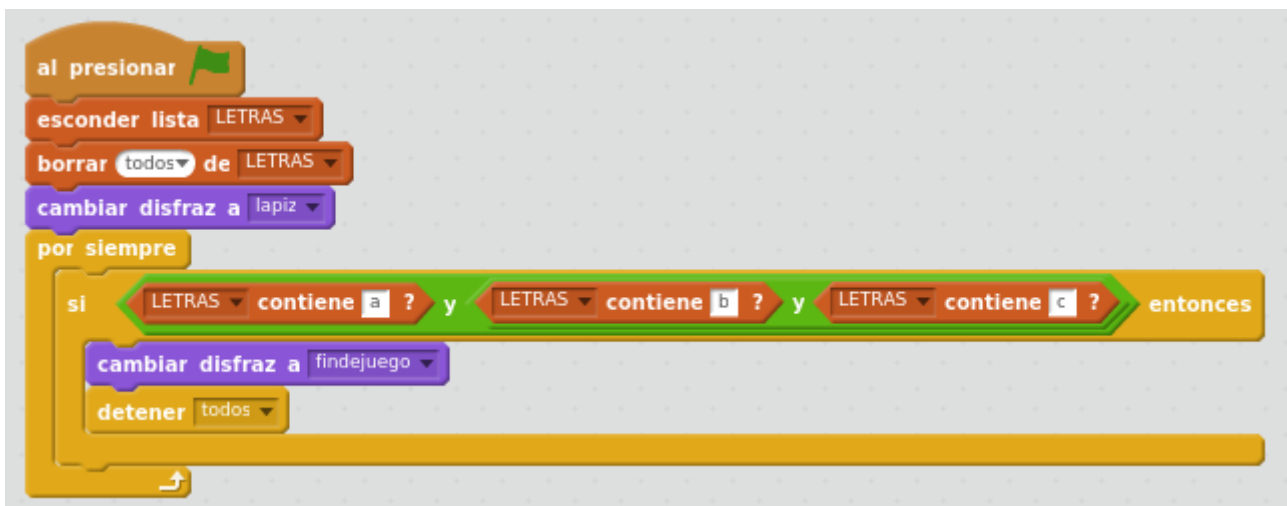
## Solución

Crear una lista (pestaña Programas - Datos) y la llamamos LETRAS.

En la programación de las letras, añadir este bloque tras recibir el mensaje de impacto:



El programa del lápiz para controlar el fin de juego quedaría así:



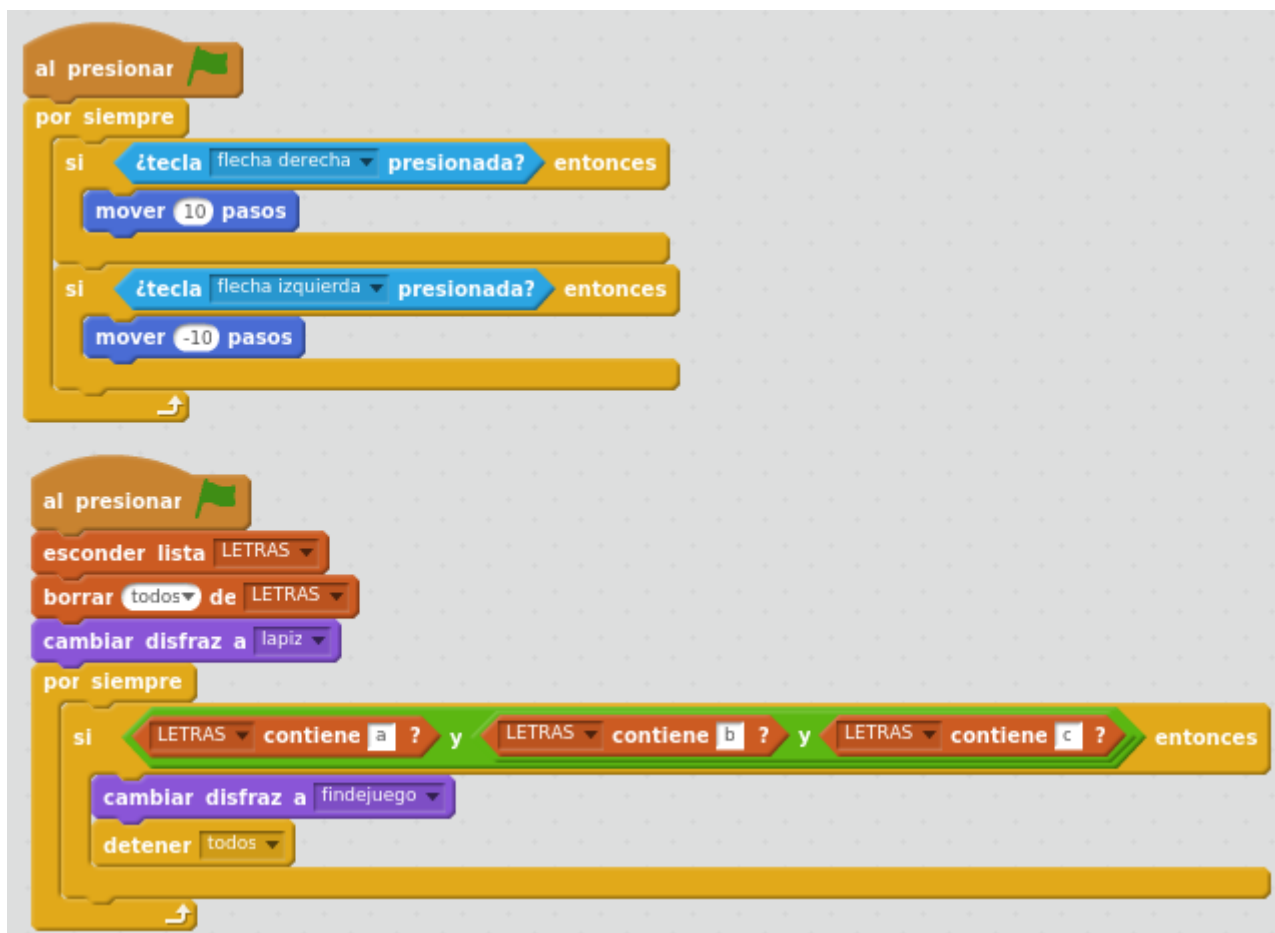
# Resultado final

Hemos creado una programación para el juego de lanzar pintura desde un lápiz contra varias letras.

Los programas de cada letra, por ejemplo de la letra A, quedan de la siguiente forma:



Los programas del lápiz quedan:



El programa del disparo queda:



El programa de la imagen de fondo queda:



**¡YA LO TENEMOS!. Utilizando Pensamiento Computacional, hemos conseguido montar un juego completo con Scratch.**

**[Ver resultado final](#)**

<https://scratch.mit.edu/projects/embed/125282917/?autostart=false>