



Girls Who Code en casa

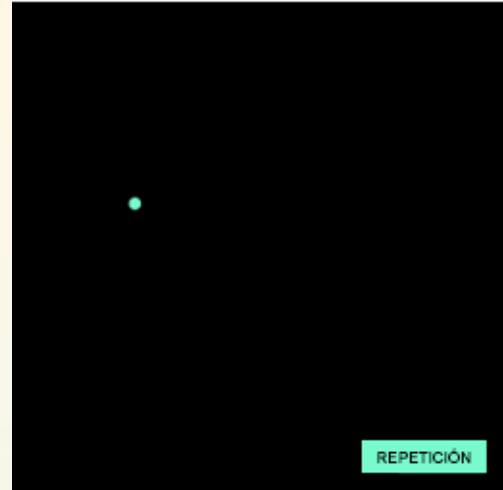
Juego Meteor Catcher: Parte 3

Hacer que el meteorito caiga

Descripción de la actividad

Al final de la Parte 2, usaste el sistema de coordenadas para dibujar el primer componente de tu juego: ¡el meteorito! Luego, estableciste el color de tu meteorito y el fondo del bordado. En esta parte, aprenderás a crear y usar variables en p5.js para mover el meteorito por la pantalla a una velocidad especificada. Combinaremos variables y operadores aritméticos como $+$ y $=$ para simular el movimiento. Sí, es mágico, ¡pero en realidad son cálculos simples! Haz clic [aquí](#) para obtener una vista previa de lo que aprenderás al final de la actividad.

Ya deberías haber completado la [Parte 1](#) y la [Parte 2](#) de la serie de juegos **Meteor Catcher** antes de embarcarte en esta actividad.



Nota: Incluimos un botón de repetición para que pueda restablecer el comportamiento del meteorito. Si no incluimos esto, solo verás un cuadro negro una vez que el meteorito se haya caído de la parte inferior de la pantalla. Solucionaremos esto en la siguiente actividad, Parte 4, con una sentencia condicional.

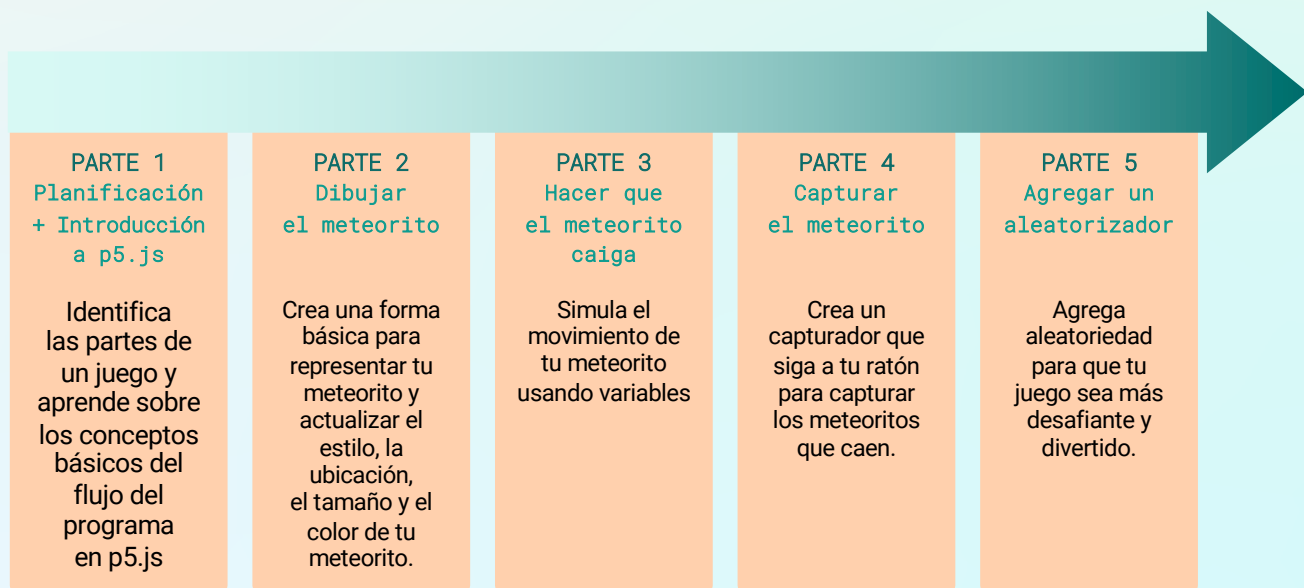
Objetivos del aprendizaje

Al finalizar esta actividad, serás capaz de:

- ☐ describir cómo simular el movimiento básico en un programa.
- ☐ programar diferentes comportamientos en elementos usando variables y operadores aritméticos.

Materiales

- [Editor en línea p5.js](#)
- [Proyecto de ejemplo del juego capturador de meteorito](#)
- [Guía de referencia de la parte 3 del juego capturador de meteorito](#)



Artículo destacado sobre “Mujeres en tecnología”: Robin Hunicke



Fuente de la imagen:
[UCSC DAMN](#)

Robin es diseñadora de videojuegos que enseña en la UC Santa Cruz y es cofundadora de Funomena. Robin comenzó su carrera en Electronic Arts como diseñadora líder, donde diseñó MySims. Luego trabajó con thatgamecompany, una compañía de juegos independiente, donde fue una de las dos mujeres en el equipo para producir el juego [Journey](#). Journey ganó varios premios al Juego del Año e incluso fue nominado para los Premios Grammy 2013 por Mejor banda sonora para medios visuales.

Con su equipo en [Funomena](#), Robin comenzó a crear videojuegos utilizando todas las plataformas diferentes, incluidas las gafas de realidad virtual. Su equipo ha hecho juegos experimentales, incluidos Luna y Woorld. A pesar de los pequeños márgenes en la producción de juegos de realidad virtual (VR), Robin cree que es importante desarrollar juegos que asuman riesgos y empujen su creatividad. En 2008, Robin fue nombrada entre las 20 Mejores Mujeres que Trabajan en la Industria de Videojuegos de Gamasutra y en 2009 recibió el Premio al Juego de Diseño de Microsoft.

Robin es una gran defensora de la diversidad en la industria del juego. Su trabajo gira principalmente en torno a la amplificación del trabajo y las voces de grupos subrepresentados. Gran parte de su trabajo como profesora en UC Santa Cruz ofrece a los estudiantes un programa que combina cursos de arte y programación para el diseño de juegos.

Ve este [video](#) para obtener más información sobre Robin y cómo trabaja para ser una fuerza positiva en la industria del juego. Obtén más información sobre Robin leyendo su [breve biografía del cuerpo docente](#) y leyendo sobre su juego AR [Woorld](#), explorando el juego [Journey](#), o sus otros proyectos.

Reflexión

Ser un experto informático es más que sencillamente ser bueno programando. Tómame unos minutos para reflexionar sobre cómo Robin y su trabajo reflejan las características que todos los verdaderos expertos informáticos deben desarrollar en sí mismos: valentía, resiliencia, creatividad y propósito.



CREATIVIDAD

¿Cómo aborda Robin los juegos de una manera diferente de la esperada?
¿Cuáles son las ventajas de abordar un proyecto de una manera inesperada?

Comparte tus respuestas con un miembro de la familia o amigo. Anima a los demás para que lean sobre Robin y se unan a la charla.

Paso 1: Uso de variables en p5.js (5-10 minutos)

Revisar las variables en JavaScript (3 a 5 minutos)

Antes de profundizar, hagamos una revisión rápida de las variables. Recuerda que las variables se usan para almacenar información (datos) en un programa de computación. Son particularmente poderosas porque podemos cambiar fácilmente el valor de una variable en el transcurso de nuestro programa.

Para crear una variable, debemos declararla primero. Esto le indica al programa que queremos crear un contenedor y nombrarlo. En JavaScript, declaramos una variable como esta: `let meteorDiameter;`. Podemos declararla e inicializarla (o asignarle un valor) al mismo tiempo como este: `let meteorDiameter = 50;`. Analicemos la sintaxis:

JAVASCRIPT	DESCRIPCIÓN
<code>let meteorDiameter = 50;</code>	<ul style="list-style-type: none">→ <code>let</code>: Palabra clave que menciona el borrado para crear una variable.→ <code>meteorDiameter</code>: El nombre de nuestra variable. Este nombre puede ser cualquier cosa que desee, pero asegúrate de que sea descriptivo. Puede ser solo una palabra, así que usamos <u>camelCase</u>.→ <code>=</code>: asigna el valor a la variable. Esto también se denomina inicialización.→ <code>50</code>: El valor almacenado actualmente en la variable. Puede almacenar cualquier tipo de datos en una variable: números, letras, cadenas, etc.→ <code>;</code>: Todas las líneas de código en p5.js deben terminar con punto y coma.

Los programadores generalmente crean y definen todas las variables que necesitarán en la parte superior. Estas se denominan variables globales. Esto significa que puedes usar esas variables en cualquier lugar de tu código. También hace que tu código sea más legible tanto para el programador como para cualquier otra persona que revise tu código. Si necesitas más información sobre las variables, ve [este video de Coding Train](#).

Agregar variables (3 a 5 minutos)

En este momento, nuestra elipse no contiene ninguna variable. Si queremos simular el movimiento, tenemos que intercambiar estos valores estáticos por variables para poder cambiar el valor de la posición x e y con el tiempo. Debemos declarar e inicializar variables para cada parámetro en nuestra elipse: x, y, al igual que el ancho y la altura.

Nombrar tus variables. Puedes usar los nombres de las variables que usamos o crear los tuyos propios. Si usas el tuyo, recuerda consultarlos correctamente más adelante

Paso 1: Uso de variables en p5.js (cont.)

Agregar las siguientes variables por encima de la función `setup()`:

- ☐ Crear una variable para almacenar la posición x y asignarle un valor de 200. Nombramos a esta variable `meteorX`, pero puedes crear tu propio nombre de variable.
- ☐ Crear una variable para almacenar la posición y, y asignarle un valor de 0. Nombramos a esta variable `meteorY`, pero puedes crear tu propio nombre de variable.
- ☐ Crear una variable para almacenar el ancho y la altura y asignarle un valor de 20. Estos valores serán los mismos ya que es un círculo. Nombramos a esta variable `meteorDiameter`, pero puedes crear tu propio nombre de variable.

Ahora que tenemos las variables, ¡usémoslas! En la función `ellipse()`, reemplazar los valores numéricos con la variable correspondiente que acabamos de crear para los siguientes parámetros:

- ☐ posición x
- ☐ posición y
- ☐ ancho y alto



No olvides verificar tu código con la Guía de referencia en la página 2.

Paso 2: Hacer observaciones sobre el movimiento (5 a 10 minutos)

Nuestro objetivo en esta parte es hacer que el meteorito caiga desde la parte superior de la pantalla a la parte inferior de la misma. Pero, ¿cómo traducimos eso en código? Analicemos un ejemplo que nos ayude a resolverlo.

Examina [este bordado](#) debajo de un círculo que se mueve de izquierda a derecha. Tómate entre 60 y 90 segundos para hacer observaciones sobre el comportamiento del círculo. Piensa en las siguientes preguntas:

- ¿En qué eje se mueve el círculo?
- ¿Cómo cambia la posición del círculo? ¿Qué valor en `ellipse()` tendrías que cambiar para que esto suceda?
- ¿Crees que el código para que esto suceda está en `setup()` o `draw()`?



Usa tus observaciones para escribir una línea de pseudocódigo para decirle al programa cómo mover la pelota. No pases a la siguiente parte hasta que hayas terminado.



No olvides revisar tus ideas con la Guía de referencia en la página 2.

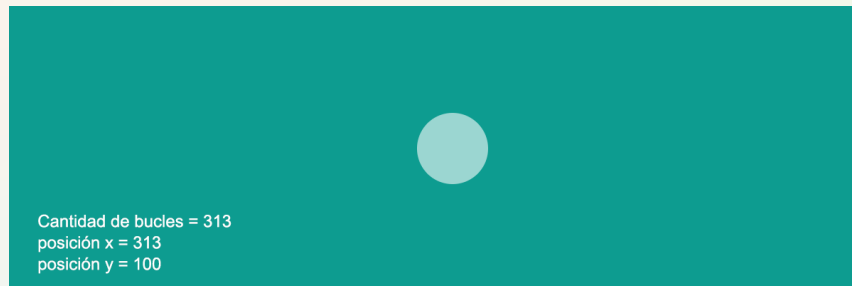
Paso 2: Hacer observaciones sobre el movimiento (cont.)

Movimiento de escritura como código

Para imitar el movimiento horizontal, queremos que el valor de x cambie cada vez que el programa entre en bucle. Recuerda: el programa funciona en un bucle. El programa ejecuta cada línea de código en `draw()` en secuencia. Una vez que llegues al final del programa, vuelve a la parte superior y comienza de nuevo. Hace esto para siempre hasta que le digas que pare. **Podemos escribir el movimiento como una línea de código estableciendo el valor x igual a sí mismo más un número:**

```
ellipse(xPosition, yPosition, 50, 50);  
xPosition = xPosition + 1; // También se puede escribir como xPosition++
```

Esto significa que el valor de x aumentará en ese número cada vez que el programa entra en bucle. Este número determina qué tan lento o rápido se mueve el meteorito por la pantalla. En otras palabras, establece la velocidad. En el [bordado de ejemplo](#) a continuación, establecemos la velocidad en 1 para que la posición x aumente en 1 con cada bucle:

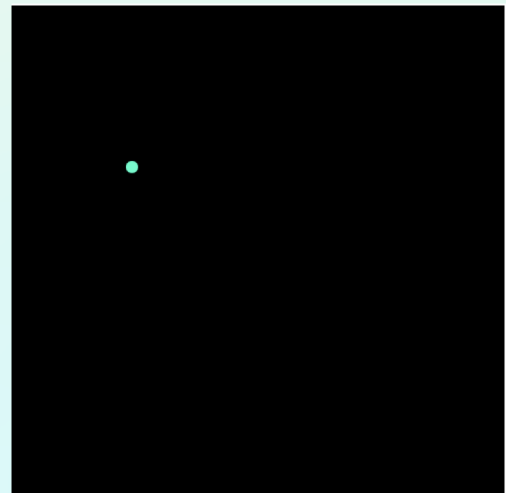


Paso 3: Agregar movimiento a tu meteorito (5-10 minutos)

Nuestro círculo de ejemplo se movió horizontalmente, pero queremos que el meteorito se mueva verticalmente por el eje y como se muestra en el bordado a continuación. Esto significa que necesitamos aumentar la posición y en lugar de la posición x. Haz clic en este enlace para obtener una vista previa de un [bordado de ejemplo](#).

Sigue estos pasos para que tu meteorito caiga:

- ☐ **Crear una nueva variable por encima de `setup()` para almacenar la velocidad.** Nombramos a nuestra variable `speed` pero puedes nombrarla como quieras. Solo asegúrate de consultarla correctamente más adelante en tu código.
- ☐ **Asignarle un valor que haga que el meteorito caiga lentamente.** Pista: Puedes usar decimales.
- ☐ En la función `draw()`, agregar una línea de código que cambie la posición y de tu meteorito para hacer que caiga desde la parte superior de la pantalla hasta la parte inferior de la pantalla.



Este meteorito está programado para restablecerse, pero tu bordado no lo hará hasta la Parte 5.



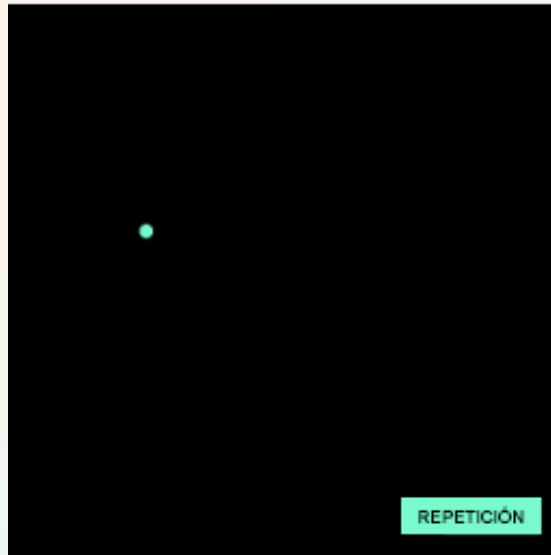
No olvides verificar tu código con la Guía de referencia en la página 3.

Paso 4: Probar tu código (5 minutos)

Pongamos a prueba lo que hemos escrito hasta ahora para asegurarnos de que nuestro programa funcione de la manera que queremos. Haz clic en el botón reproducir para ejecutar tu borrador. Deberías tener:

- Un meteorito que cae a una velocidad lenta desde la parte superior de la pantalla hasta la parte inferior de la misma.
- Debería desaparecer en la parte inferior.
- No deberías tener un botón Volver a reproducir.

Ejemplo



Incluimos un botón Volver a reproducir para que pueda restablecer el comportamiento del meteorito. Si no incluimos esto, solo vería un cuadro negro una vez que el meteorito se haya caído de la parte inferior de la pantalla. Solucionaremos esto en la siguiente actividad, Parte 4, con una sentencia condicional.

¿No funciona como deseas? Prueba estos **consejos de depuración**:

- ¿Tu código está dentro de las llaves correctas?
- ¿Tiene punto y coma al final de cada línea de código?
- ¿Escribiste correctamente los nombres de las variables y de las funciones?
- ¿Tus funciones están en la ubicación y secuencia correctas? Recuerda que el orden es importante en el flujo del programa.
- ¿Está tu operador aritmético en el lugar correcto?
- ¿El valor de tu variable de velocidad es demasiado rápido (valor alto) o demasiado lento (valor bajo)?
- ¿Tu meteorito está cayendo de arriba hacia abajo? ¿Actualizaste la variable de posición y de tu meteorito?

Si necesitas un repaso sobre las mejores prácticas para la depuración, echa un vistazo [a esta fantástica publicación](#) de la comunidad p5.js.

Paso 5: Verificar la comprensión

¿Cómo cambiarías la ecuación de velocidad para hacer que el meteorito se mueva desde la parte inferior de la pantalla hacia la parte superior?



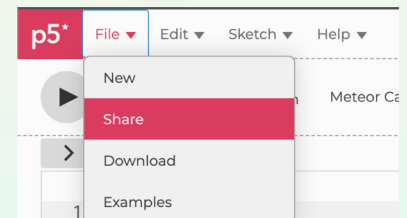
No olvides revisar tus ideas con la Guía de referencia en la página 3.

Paso 6: Compartir tu proyecto de Girls Who Code en casa (5 minutos)

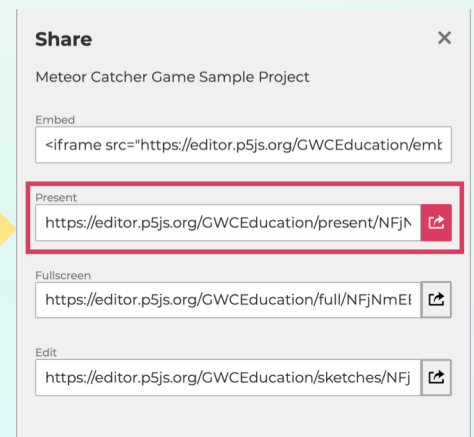
Nos encantaría ver tu trabajo y sabemos que a otros también les gustaría. Comparte tu juego con nosotros. No olvides etiquetar @girlswhocode #codefromhome, ¡y quizá la destaquemos en nuestra cuenta!

Sigue estos pasos para compartir tu proyecto:

- Guarda tu proyecto primero.
- En el **Menú** Archivo, elige la **opción** Compartir en el menú desplegable.
- Elige la **opción** Vínculo en el menú desplegable.
- Copia el **enlace** Presente y pégalo donde quieras compartirlo.



Enlace al proyecto



¡Espera más proyectos de Girls Who Code en casa!

