



# Girls Who Code en casa

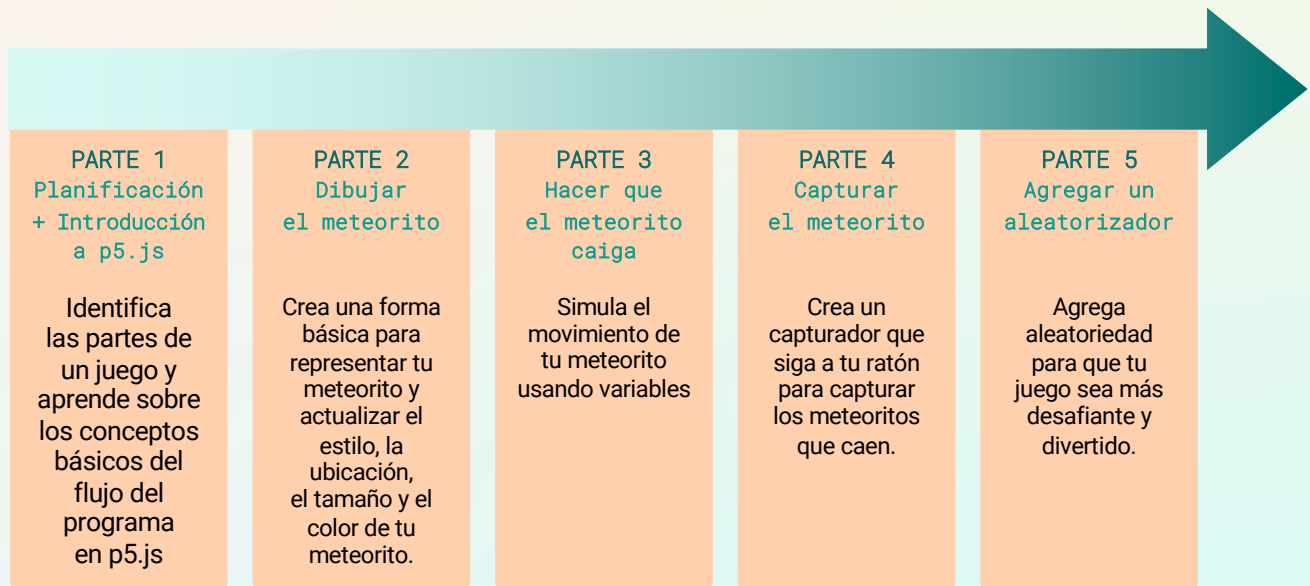
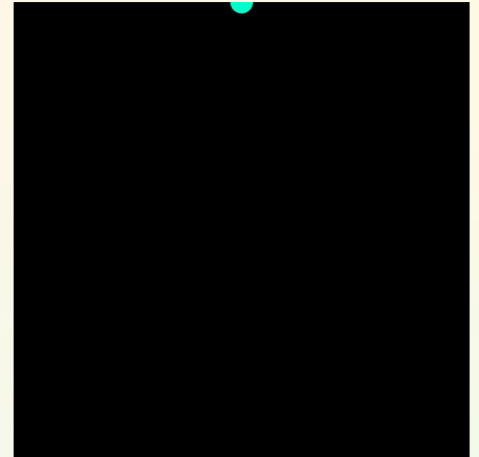
**Juego Meteor Catcher: Parte 2**

Dibujar el meteorito

## Descripción de la actividad

¡Bienvenido nuevamente! La semana pasada, exploraste el entorno de p5.js y comenzaste a planificar las partes de tu juego Meteor Catcher. En esta parte, usarás el sistema de coordenadas para dibujar el primer componente de tu juego: ¡el meteorito! También aprenderás más sobre cómo usar el color en p5.js y configurar el color para tu meteorito y fondo de bordado.

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



## Objetivos del aprendizaje

Al finalizar esta actividad, serás capaz de:

- ☐ describir el sistema de coordenadas p5.js y su relación con los píxeles en la pantalla.
- ☐ usar funciones y comandos incorporados para dibujar formas básicas en el plano de coordenadas.

## Materiales

- [Editor en línea de p5.js](#)
- [Proyecto de muestra del juego Meteor Catcher](#)
- [Guía de referencia de la parte 2 del juego Meteor Catcher](#)

## Artículo destacado sobre “Mujeres en tecnología”: Phoenix Perry



Fuente de la imagen:  
[Hackaday.io](https://hackaday.io)

Cuando Phoenix era más joven, sus padres le compraron una consola de juegos [Atari](#) para la casa. La introducción a este nuevo mundo de la tecnología despertó el interés de Phoenix en la tecnología y comenzó a aprender a programar en [BASIC](#). Después de graduarse con un título universitario, Phoenix comenzó su carrera como diseñadora web para [Evite](#). Aquí, duró muchas noches interminables, un comportamiento normal en la compañía, y tomó analgésicos de las tensiones en sus muñecas debido a la falta de descanso. Luego desarrolló el [túnel carpiano](#), un síndrome nervioso común en los programadores debido a un aumento de la tensión en los huesos y ligamentos de la mano.

Phoenix pasó muchos años lejos de la industria tecnológica debido a su condición, trabajando como directora de arte. Luego se encontró con el programa [Integrated Digital Media en NYU Tandon](#), donde se convirtió en profesora adjunta e investigadora para enseñar sobre el diseño, el juego y la realización del desarrollo del juego. Fue a través de esta experiencia que continuó hasta convertirse en cofundadora de [Code Liberation](#). Code Liberation espera enseñar, preparar y apoyar a las mujeres, las mujeres no binarias, las mujeres femininas y las niñas que identifican a las personas para que busquen empleos en STEAM. Ofrecen clases gratuitas, talleres, juegos de jame, hackatón y noches de juegos sociales a mujeres de todas las edades y en diferentes etapas de sus carreras. Phoenix utiliza su experiencia para ayudar a la próxima generación de mujeres a desarrollar las habilidades necesarias para unirse a la industria de la tecnología. Ahora se ha asociado con [University of Arts London](#) para abrir un capítulo de Code Liberation en el Reino Unido.

Mira este [video](#) para obtener más información sobre Phoenix y por qué la diversidad es importante en la tecnología. ¿Deseas obtener más información sobre Phoenix? Ve su charla [TED Talk](#), explora su sitio web [personal](#), y lee este [artículo](#) para obtener más información sobre el camino académico de Phoenix y las dificultades para llegar a donde está hoy.

## Reflexión

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



VALENTÍA

Después de desarrollar el túnel carpiano, Phoenix se vio incapaz de trabajar y moverse. Esto dificultaba su regreso a la industria de la tecnología. Analiza cómo Phoenix mostró valentía al regresar a la industria y al mismo tiempo apoyar y preparar a otras mujeres en Code Liberation.

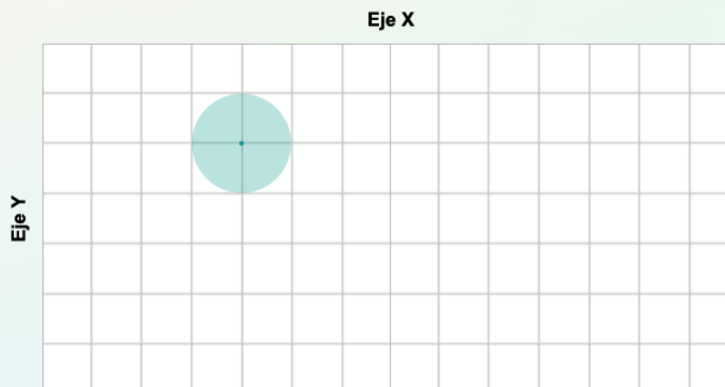
Comparte tus respuestas con un familiar o amigo. Animar a los demás para que lean sobre Phoenix y se unan a la charla.

## Paso 1: Ordenar un círculo (2-4 minutos)

Digamos que tu amigo te da una instrucción para dibujar un círculo en una hoja de papel. Puedes dibujar el círculo o pedir más información. Si solicitas más información, puedes preguntar, ¿en qué lugar del papel? ¿Qué tan grande? ¿De qué color? ¿Un círculo perfecto o más de un óvalo? Para responder la pregunta de dónde, tu amigo podría decir: “Un tercio del camino desde el lado izquierdo y tres cuartos del camino dirigiéndose hacia abajo”. Ese tipo de instrucción podría funcionar si estás hablando con un humano y no necesitas ser específico. Pero no funcionará para una computadora. En cambio, podemos usar el sistema de coordenadas para especificar una ubicación para los elementos que queremos mostrar en nuestro programa.

El sistema de coordenadas es un sistema que utiliza uno o más números para identificar la ubicación de un punto en el espacio. Los sistemas de coordenadas pueden estar en un plano 2D o en un espacio 3D. Los planos de coordenadas (es decir, 2D) tienen un eje x que se ejecuta horizontalmente y un eje y que se ejecuta verticalmente para formar una cuadrícula. Usan un par ordenado para indicar un punto: (posición x, posición y).

p5.js también puede funcionar en un espacio tridimensional con un eje z. Puedes ver la opción para un parámetro del eje z en la documentación p5 en algunas funciones, pero no debes incluirla.



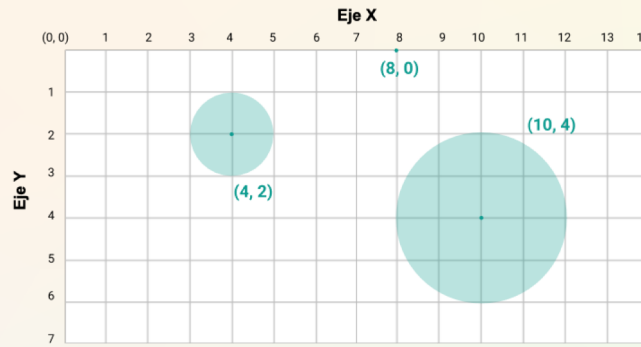
Considera el plano de coordenadas anterior. Piensa en las instrucciones que darías a una computadora para dibujar el círculo en ese lugar.



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

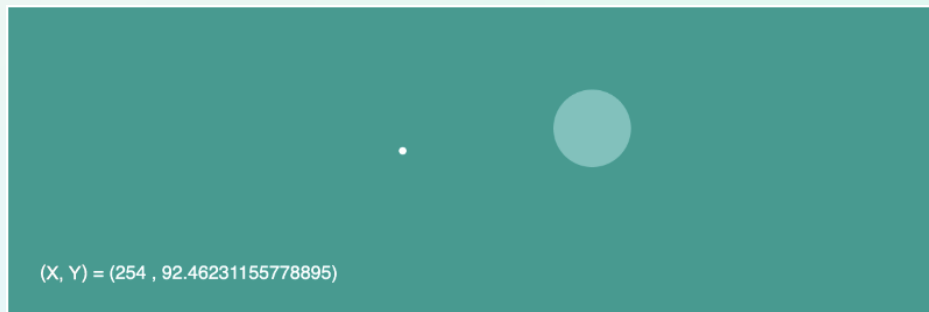
## Paso 2: Practicar el uso del sistema de coordenadas (5-8 minutos)

Cada píxel de la pantalla tiene una dirección única en el sistema de coordenadas. Para dibujar píxeles a la pantalla, debemos dar a nuestro programa la coordenada x (es decir, la ubicación en el eje x) y la coordenada y (es decir, la ubicación en el eje y) del píxel.



El origen o (0, 0), en el sistema de coordenadas de la pantalla se encuentra en la esquina superior izquierda. A medida que te mueves hacia la derecha en la pantalla, el valor de la coordenada x aumenta. A medida que te desplazas hacia abajo en la pantalla, el valor de la coordenada y aumenta. Esto puede parecer un poco diferente de los diseños del sistema de coordenadas que has visto en la clase de matemáticas, donde el origen está en el centro o en la esquina inferior izquierda.

Explora este [bordado](#) y usa el ratón para calcular el centro, el ancho y la altura del círculo.

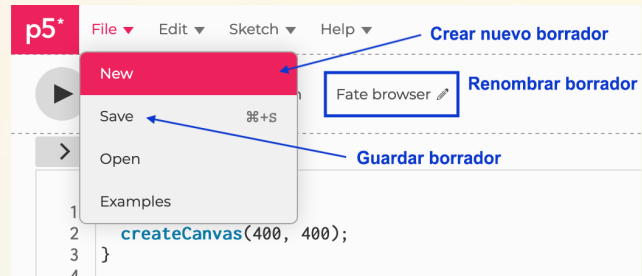


No olvides revisar tus ideas con la Guía de referencia en la página 2.

## Paso 3: Crear el bordado de tu proyecto (5 a 10 minutos)

En las secciones restantes, comenzaremos a escribir el código para tu juego. Primero, debemos crear tu bordado principal del proyecto.

- ☐ **Inicia sesión en el editor en línea p5.js.** El editor te da automáticamente un bordado en blanco con el código de inicio. Como alternativa, puedes crear un nuevo bordado dirigiéndote a Archivo > Nuevo.
- ☐ **Haz clic en el ícono del lápiz para nombrar el bordado** a algo que puedas reconocer fácilmente como Meteor Catcher Game v1. *Nota: Esto se encuentra en el área de información de bordados debajo de la barra de herramientas.*
- ☐ **Luego, ve a Archivo y haz clic en Guardar.** También puedes guardar con el método abreviado del teclado Command S (Mac) o Control S (Windows). Asegúrate de navegar dentro del editor antes de usar estos accesos directos.
- ☐ **Creas un comentario multilínea en la parte superior de tu bordado con la siguiente información:**
  - ☐ **Título del programa:** Debe ser igual al nombre del bordado.
  - ☐ **Versión del programa:** ¿Es esta la primera versión o la segunda? Si realizas grandes cambios, es una buena práctica crear una nueva versión.
  - ☐ **Autor:** Por (su nombre y la inicial del apellido).
  - ☐ **Descripción:** Una oración o dos sobre lo que hace.



En la parte superior de tu bordado, incluirás un comentario que brinda información básica sobre el bordado. Usa **comentarios de código** para recordar cómo funciona algo, el razonamiento para una decisión o una tarea de seguimiento.

- ➔ Los comentarios de una sola línea usan una barra diagonal doble hacia adelante, `//`.
- ➔ Los comentarios de varias líneas usan una barra diagonal hacia adelante y un asterisco, `/*`, para abrirla y un asterisco y una barra diagonal hacia adelante, `*/`, para cerrarla.

```
// Este es un comentario de una sola línea

/*
Esto es un
comentario de varias líneas
*/
```

## Paso 4: Dibujar el meteorito (3-5 minutos)

Ahora que hemos guardado nuestro bordado del proyecto, creemos nuestro primer componente del juego: ¡el meteorito! Primero aprenderemos a dibujar la figura en un lugar específico, luego la rellenaremos con color para cambiar la apariencia.

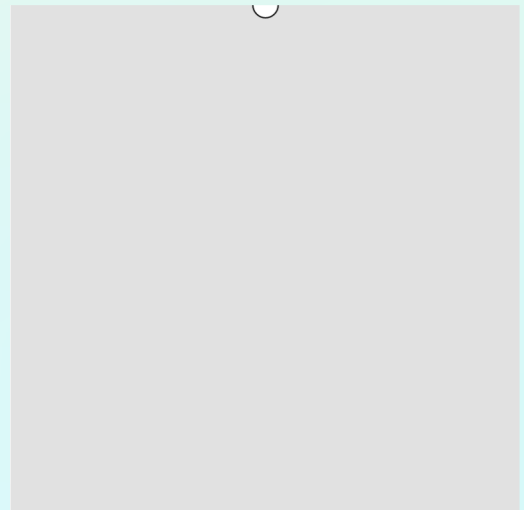
### Forma y ubicación

Piensa en el último paso en el que aprendimos sobre el sistema de coordenadas. Según lo que sabemos ahora, si queremos dibujar una figura, tenemos que decirle a la computadora que dibuje cada píxel individualmente. Esto se vuelve bastante tedioso muy rápidamente, así que p5 creó funciones prefabricadas que dibujan formas por nosotros. Todo lo que debemos hacer es definir la ubicación donde queremos la forma, luego dar el ancho y la altura. Examinemos la sintaxis para ver un círculo a continuación:

JAVASCRIPT	DESCRIPCIÓN
<pre>ellipse(x, y, width, height);</pre>	<ul style="list-style-type: none"><li>→ <b>ellipse</b>: El nombre de la función. Elipse es otra palabra para oval. <a href="#">p5.js Referencia</a></li><li>→ <b>()</b>: Utilizamos paréntesis para indicar a nuestro programa que necesita llamar a la función. A veces incluimos parámetros o entradas en la función dentro de nuestros paréntesis.</li><li>→ <b>x</b>: La coordenada x en el centro de la elipse.</li><li>→ <b>y</b>: La coordenada y en el centro de la elipse.</li><li>→ <b>width</b>: Establece el ancho de la elipse en píxeles.</li><li>→ <b>height</b>: Establece la altura de la elipse en píxeles.</li><li>→ <b>,</b>: Usamos comas para separar los diferentes parámetros o entradas en las funciones.</li><li>→ <b>;</b>: Todas las líneas de código en p5.js deben terminar con punto y coma.</li></ul>

Usar la función `ellipse()` para dirigir el meteorito a la pantalla:

- ☐ Agrega la función `ellipse()` a tu bordado dentro de la función `draw()`.
- ☐ Colócalo en el centro del lienzo (su posición del eje x) en la parte superior (su posición del eje y). Consulta el gráfico anterior si necesitas un repaso.
- ☐ Establece el tamaño del círculo, o elipse, en 20 píxeles de ancho y 20 píxeles de alto.
- ☐ Agrega un comentario para recordarle que este es el meteorito.
- ☐ Ejecuta tu código presionando el botón de reproducción para probarlo.



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

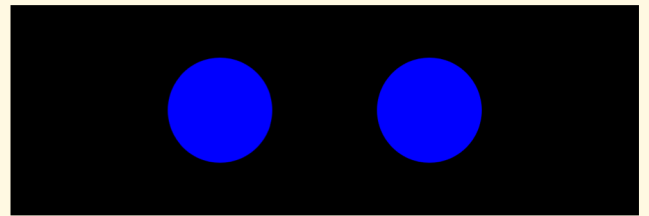
## Paso 5: Llenar el meteorito con color (5 a 8 minutos)

Para agregar color a las formas, podemos usar la función `fill()`. Esto llenará cualquier forma dibujada después del comando con el color especificado. En el ejemplo a continuación, ambos círculos son azules porque aparecen después de la función `fill()`.

### JAVASCRIPT

```
function setup() {  
  createCanvas(600, 200);  
}  
  
function draw() {  
  background(0);  
  
  // Hacer que ambos círculos sean azules  
  fill(0, 0, 255);  
  ellipse(width / 3, height / 2, 100, 100);  
  ellipse(width / 3*2, height / 2, 100, 100);  
}
```

### RESULTADO



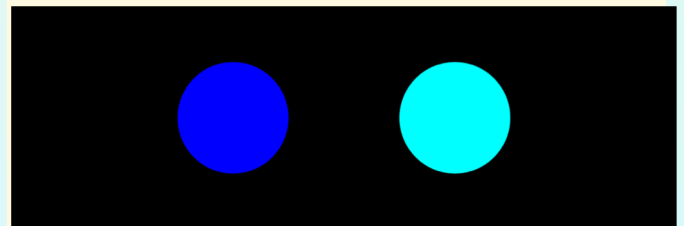
Pero, ¿qué sucede si queremos que la segunda figura sea verde azulado en lugar de azulada? Debemos agregar otra función `fill()` antes de esa forma:

### JAVASCRIPT

```
function setup() {  
  createCanvas(600, 200);  
}  
  
function draw() {  
  background(0);  
  
  // Círculo azul  
  fill(0, 0, 255);  
  ellipse(width / 3, height / 2, 100, 100);  
  // Círculo verde azulado  
  fill(0, 255, 255);  
  ellipse(width / 3*2, height / 2, 100, 100);  
}
```

### RESULTADO

Este cambio daría como resultado un círculo azul y un círculo verde azulado



El método `fill()` utiliza el modo de color RGB que utiliza una combinación de luz roja, verde y azul para crear un espectro de colores.

- `(0,0,255)` mostraría un color **azul**
- `(0,255,255)` daría un color **verde azulado**.



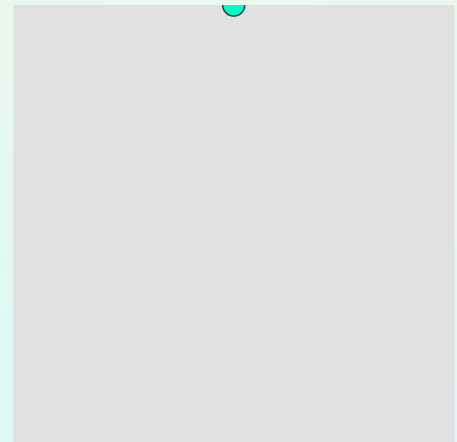
### Paso 5: Llenar el meteorito con color (cont.)

Comprueba la sintaxis para la función `fill()` utilizando el modo de color RGB a continuación. El modo de color RGB utiliza combinaciones de luz roja, verde y azul para crear una gama de colores digitales. Puedes asignar un valor a cada color entre 0 y 255. Por ejemplo, (255, 0, 0) sería rojo, (0, 0, 0) sería negro y (255, 136, 0) sería naranja. Puedes jugar con diferentes valores y colores usando herramientas como [selectores de color](#) o herramientas de paleta como [Colors](#).

JAVASCRIPT	DESCRIPCIÓN
<pre>fill(redValue, greenValue, blueValue);</pre>	<ul style="list-style-type: none"><li>→ <code>fill</code>: El nombre de la función.</li><li>→ <code>()</code>: Utilizamos paréntesis para indicar a nuestro programa que necesita llamar a la función. A veces incluimos parámetros o entradas en la función dentro de nuestros paréntesis.</li><li>→ <code>redValue</code>: El valor rojo entre 0 y 255.</li><li>→ <code>blueValue</code>: El valor azul entre 0 y 255.</li><li>→ <code>greenValue</code>: El valor verde entre 0 y 255.</li><li>→ <code>;</code>: Todas las líneas de código en p5.js deben terminar con punto y coma.</li></ul>

Usar la función `fill()` para agregar color a tu meteorito:

- ☐ Elegir un color para tu meteorito y tomar nota de los valores RGB.
- ☐ Llamar a la función `fill()` con los valores RGB de tu meteorito (recuerda que la ubicación es importante).
- ☐ Presionar el botón de reproducción para ejecutar tu programa cuando hayas terminado. El color de tu meteorito deberá cambiar al que especificaste.



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

## Paso 6: Cambiar el esquema (3 a 5 minutos)

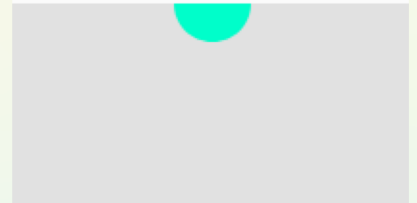
Quizá hayas notado que hay un contorno negro alrededor de tu meteorito. Podemos eliminar este esquema utilizando la función `noStroke()`. Llamar a esta función significa que ninguna de las figuras de tu bordado tendrá contornos. Si deseas habilitar los contornos, debes llamar a la función `stroke()` por encima de esa forma.

Puedes utilizar otras funciones como `stroke()` y `strokeWeight()` para controlar el color y el grosor de los bordes de la forma.

Eliminar el contorno utilizando la `noStroke()`:

- ☐ Agregar la función `noStroke()` debajo de la función `background()`. Ya que no queremos contornos en ninguna de nuestras figuras, lo colocaremos en la parte superior.
- ☐ Ejecutar el programa cuando haya terminado.

Ya no podrás ver un esquema en tu meteorito. Esta es una captura de pantalla en primer plano de cómo el meteorito debe verse en la parte superior de tu bordado.



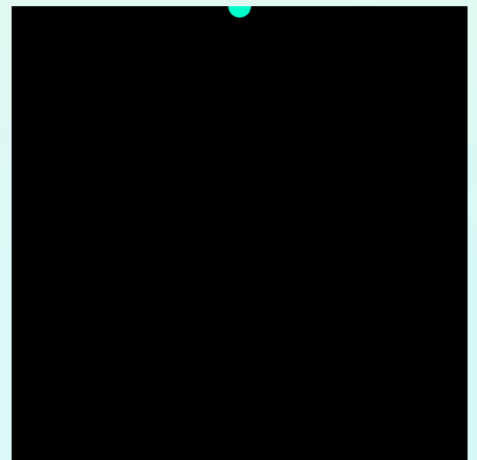
No olvides verificar tu código con la Guía de referencia en la página 4.

## Paso 7: Agregar el color de fondo (3 a 5 minutos)

Pon a prueba tu conocimiento sobre el color. Rellenar el lienzo con la función `background()`. En este momento, solo tiene un valor, pero de manera similar a la función `fill()`, puede tomar parámetros para otros modos de color como RGB.

Cambiar el fondo de tu juego:

- ☐ Elegir un color de fondo y tomar nota de sus valores RGB. (Aquí están los [selectores de color](#) y las herramientas de paleta, como [Colors](#) mencionados anteriormente).
- ☐ Actualizar la función `background()` dentro de `draw()` con sus nuevos valores de color.

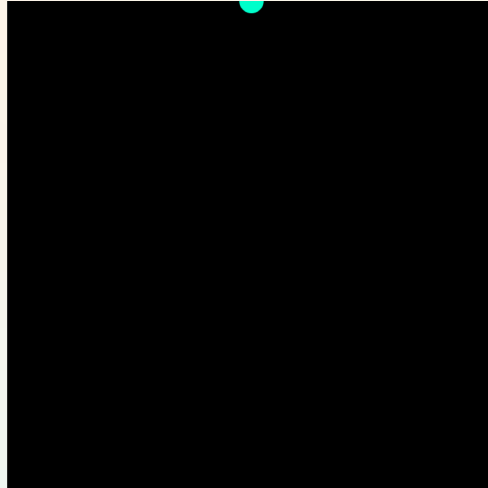


No olvides verificar tu código con la Guía de referencia en la página 4.

## Paso 8: Probar tu código (3 a 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 bordado. Deberías tener:

- Un lienzo de 400 por 400 con un fondo de color.
- Un meteorito circular en la parte superior central del lienzo con un nuevo color y sin borde.



¿No trabaja de la manera que tú deseas? Si tienes un error que impide que el código se compile y se ejecute, p5.js mostrará un mensaje de error en la consola. Cuando algo no funcione correctamente, comienza a resolver el problema.

De lo contrario, 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 la función y de la variable? Recuerda que JavaScript distingue entre mayúsculas y minúsculas.
- ¿Tus funciones están en la ubicación y secuencia correctas? Recuerda que el orden es importante en el flujo del programa.
- ¿Están los valores de tus parámetros dentro del rango correcto para la función? Por ejemplo, ¿hay un valor x de 500 aunque el lienzo tenga 400 píxeles de ancho? ¿Tus valores RGB están entre 0 y 255?

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

## Paso 9: Verificar la comprensión

Describir la ubicación, el tamaño y el color de la forma en el código a continuación:

```
function setup() {  
  createCanvas(100, 100);  
}  
  
function draw() {  
  fill(0, 0, 255);  
  ellipse(50, 50, 5, 5);  
}
```



No olvides revisar tus ideas con la Guía de referencia en la página 5.

## Paso 10: 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 pseudocódigo con nosotros. No olvides etiquetar [@girlswhocode](#) [#codefromhome](#), ¡y quizá la destaquemos en nuestra cuenta!

¡Espera más proyectos de Girls Who Code en casa!

