



Girls Who Code en casa

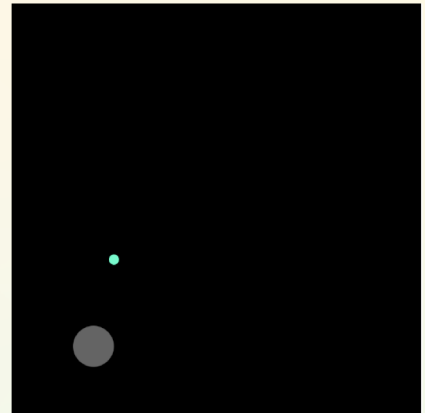
Juego del Meteor Catcher: Parte 5

Agregar un aleatorizador

Descripción de la actividad

En esta parte final, exploraremos una manera de hacer que tu juego sea más desafiante y más divertido. Aprenderás a usar la función [random\(\)](#) para agregar complejidad al comportamiento de tu meteorito. Por último, personaliza tu proyecto probando una o más de nuestras extensiones. Haz clic [aquí](#) para obtener una vista previa de lo que aprenderás al final de la actividad.

Ya deberías haber completado la [Parte 1](#), [Parte 2](#), [Parte 3](#), y [Parte 4](#) de la **Serie de juegos Meteor Catcher** antes de embarcarte en esta actividad.



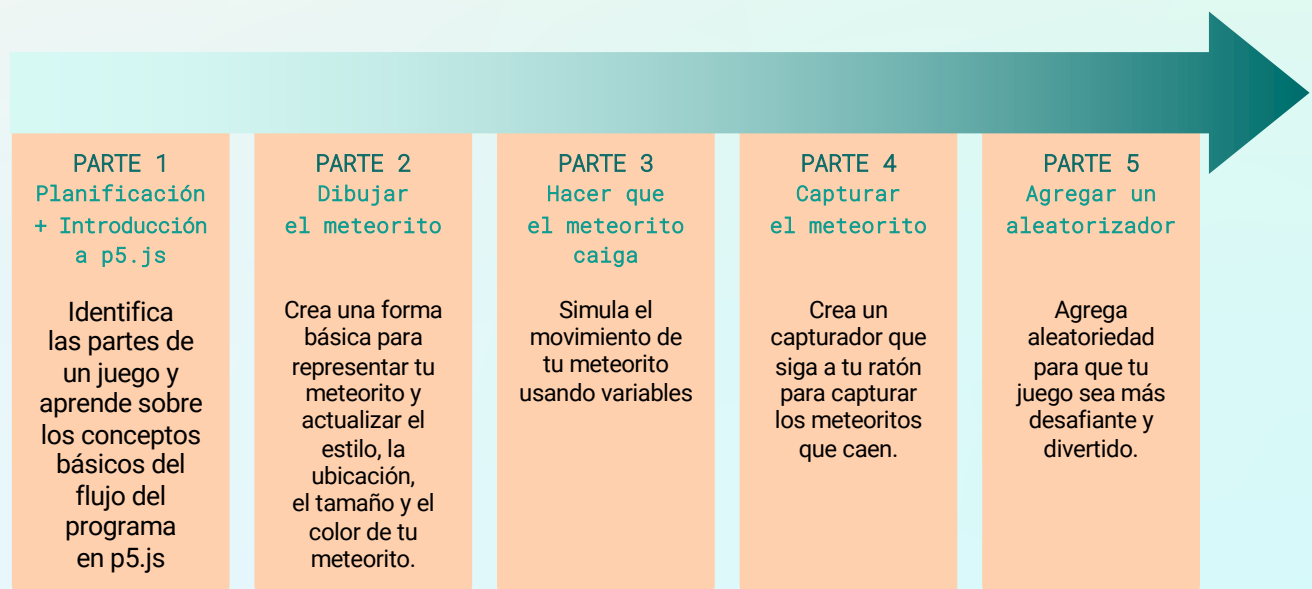
Objetivos del aprendizaje

Al finalizar esta actividad, serás capaz de:

- ☐ explicar cómo se puede usar la función `random()` para aumentar o disminuir la dificultad de un juego.

Materiales

- [Editor en línea p5.js](#)
- [Proyecto de muestra del juego Meteor Catcher](#)
- [Guía de referencia de la parte 5 del juego Meteor Catcher](#)



Artículo destacado sobre “Mujeres en tecnología”: Lisette Titre-Montgomery



Fuente de la imagen:

[Lisette Titre-Montgomery](#)

Cuando bailas con Dance Central o creas personajes en Los Sims, ¿te preguntas alguna vez por la tecnología que te llevó allí? Lisette Titre-Montgomery trabaja como ingeniera y diseñadora detrás de los videojuegos más populares del mercado. Con más de diecisiete años de experiencia en la industria de los videojuegos y trece títulos lanzados, ha cerrado con éxito la brecha entre el jugador de videojuegos y el desarrollador.

Lisette también dedica su tiempo a aportar más diversidad a la industria de los videojuegos a través de la tutoría y las iniciativas de contratación inclusivas con la Casa Blanca. Continúa siendo defensora de los planes de estudio basados en juegos en educación K-12 para involucrar a los estudiantes en la educación y carreras de STEAM con un programa llamado [Gameheads](#), con sede en Oakland, CA.

Mira este [video](#) para obtener más información sobre Lisette y algunos de los desafíos que enfrentó al trabajar en varios juegos y cómo los superó.

¿Deseas obtener más información sobre Lisette y su trabajo?

- Consulta su [sitio web personal](#) para obtener más información sobre algunos de sus trabajos y proyectos.
- Lee este [artículo](#) sobre los antecedentes de Lisette y cómo llegó a la industria del videojuego.
- Busca el [perfil de Lisette](#) para ver algunos aspectos destacados de su carrera y sus consejos para los estudiantes interesados en comenzar en la industria del videojuego.

Reflexión

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



VALENTÍA

Lisette nunca había trabajado en la industria de los videojuegos hasta su primer trabajo. ¿Qué medidas tomó para tener éxito en su función?

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

Paso 1: Reflexionar sobre el desafío de tu juego (2 a 4 minutos)

En este momento, tenemos todas las partes críticas de nuestro videojuego cubiertas: los componentes, la mecánica principal, el juego y el espacio. Pero el videojuego aún no es muy divertido, principalmente porque no es lo suficientemente desafiante. Encontrar el nivel de dificultad adecuado es clave para los diseñadores de juegos. Una herramienta que puedes usar para aumentar la dificultad (y hay muchas) es incluir un elemento de aleatoriedad. Esto evita que el jugador anticipe lo que vendrá después sorprendiéndolo con nueva información a la que debe adaptarse.

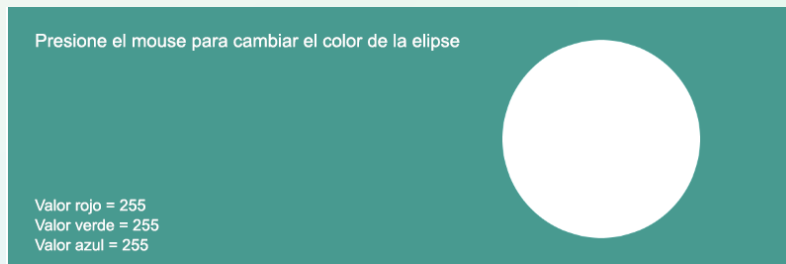
¿Dónde podemos agregar aleatoriedad al juego para que sea más desafiante? Tómame un minuto para pensarlo y luego compara tus ideas con las que aparecen a continuación.



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

Paso 2: Explorar la función `random()` (5 minutos)

Podemos usar la función `random()` para agregar aleatoriedad a nuestro código. Esta función devuelve, o emite, un número de punto flotante aleatorio entre un rango específico de valores. Punto flotante significa que el número puede contener un decimal. Esto permite un espectro de números mucho más grande. Explora este [bordado de ejemplo](#) donde cada clic del ratón dentro del lienzo genera un valor aleatorio para los valores rojo, verde y azul en la función `fill()`.



Veamos la sintaxis de la función `random()`.

JAVASCRIPT	DESCRIPCIÓN
<code>random(min, max);</code>	<ul style="list-style-type: none">→ <code>random</code>: El nombre de la función.→ <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.→ <code>mín</code>: El extremo inferior del rango aleatorio que deseas que incluya este número como una opción.→ <code>máx</code>: El extremo superior del rango aleatorio que deseas que excluya este número como una opción.→ <code>;</code>: todas las líneas de código en JavaScript deben terminar con un punto y coma.

Paso 3: Actualizar tu pseudocódigo (2-4 minutos)

Antes de agregar nuestras nuevas líneas de código, hagamos una lluvia de ideas sobre lo que queremos que suceda primero en el idioma humano. Escribe el pseudocódigo para cada función `random()`. Intenta ser lo más específica posible y utiliza los rangos de valores especificados. Cuando hayas terminado, marca tu respuesta a continuación:

- **Ubicación inicial del meteorito.** Rango de valores: 0 a 400
- **Velocidad del meteorito.** Rango de valores: 0,5 a 4
- **Tamaño del meteorito.** Rango de valores: 10 a 30



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

Paso 4: Agregar tu código (5-8 minutos)

Ahora necesitamos traducir tu pseudocódigo en un código real. Todos los códigos que escribimos se incluirán en ambas sentencias condicionales debajo de la afirmación existente. En este momento, solo volvemos a dibujar el meteorito en la parte superior de la pantalla si se cruza con el captador o la pared inferior. Agregar estos enunciados en la sentencia condicional aleatorizará la ubicación, la velocidad y el tamaño del nuevo meteorito.

Localiza la primera sentencia condicional. Agrega estas nuevas líneas de código bajo `meteorY = 0`;

- ☐ Escribe la línea de código que aleatoriza la ubicación inicial del meteorito a un valor entre 0 y 400.
- ☐ Escribe la línea de código que aleatoriza la velocidad del meteorito a un valor entre 0,5 y 4.
- ☐ Escribe la línea de código que aleatoriza el tamaño del meteorito a un valor entre 10 y 30.

Ubica la segunda sentencia condicional.

- ☐ Copia las afirmaciones que acabas de escribir para la primera sentencia condicional y cópialas bajo `meteorY = 0`; en la segunda sentencia condicional.

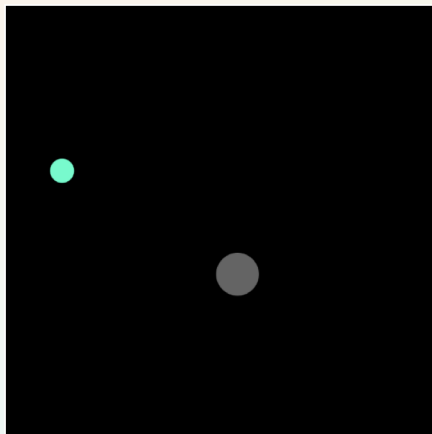
Es posible que hayas notado que las líneas de código dentro de las sentencias condicionales se ven iguales. ¡Eso es cierto! Puedes combinar estas sentencias condicionales con el operador lógico "or" - ||. También puedes combinarlos al anidarlos juntos. Optamos por separarlas para que las extensiones fueran un poco menos confusas. Si deseas combinar estas sentencias condicionales, ¡adelante!

Paso 5: Probar tu código (5 a 10 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 su bordado.

Ejecutar tu código y probar lo siguiente:

- ☐ Trata de hacer que tu meteorito y tu capturador se entrecrucen. Cuando se cruzan, el meteorito actual debe desaparecer y un meteorito “nuevo” debe aparecer en una posición x aleatoria, con un nuevo tamaño y una nueva velocidad.
- ☐ Espera a que el meteorito se intersecte con la parte inferior de tu lienzo. Cuando se cruzan, el meteorito actual debe desaparecer y un meteorito “nuevo” debe aparecer en una posición x aleatoria, con un nuevo tamaño y una nueva velocidad.



Haz clic [aquí](#) para ejecutar el bordado del ejemplo.

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

- ¿Tu código está dentro de las llaves correctas?
- ¿Hay punto y coma al final de cada línea de código?
- ¿Tus funciones están en la ubicación correcta? ¡Recuerda que el orden es importante en el flujo del programa!
- ¿Son correctos los parámetros de tu función `random()`?



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

Paso 7: Probar tu código y recibir comentarios (5 a 10 minutos)

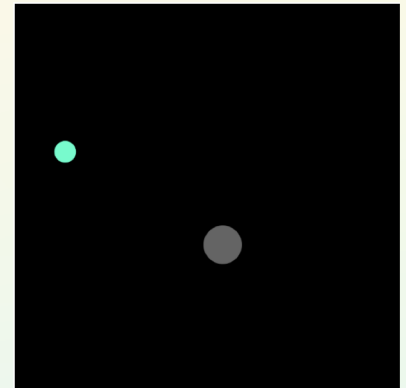


Has desbloqueado un hito importante: has terminado de construir la parte central de tu proyecto. Antes de avanzar, probaremos tu código para asegurarnos de que tu juego esté funcionando correctamente. Esta también es una gran oportunidad para obtener comentarios sobre tu juego de amigos y familiares. Comparte tu proyecto y esta lista de verificación con tu comprobador.

Lista de verificación

Ejecuta tu bordado, luego usa la lista de verificación a continuación para probar tu código:

- ☐ Tienes un lienzo de 400 por 400. El fondo se rellena con el color que elijas.
- ☐ Tu juego comienza con un meteorito (es decir, una elipse) que cae lentamente desde el centro de la parte superior del lienzo. El meteorito no tiene contorno y está relleno con un color de tu elección.
- ☐ Hay un capturador (es decir, una elipse) que es de color blanco y semitransparente. El centro del capturador sigue el movimiento del ratón.
- ☐ El programa imprime la distancia entre el meteorito y el capturador a la consola.
- ☐ Si el meteorito y el capturador se cruzan, el juego se actualiza para dibujar un nuevo meteorito.
- ☐ Si el meteorito y la parte inferior del lienzo se cruzan, el juego se actualiza para dibujar un nuevo meteorito.
- ☐ Los nuevos meteoritos comienzan en la parte superior del lienzo en una ubicación aleatoria en el eje x con una velocidad aleatoria y un tamaño aleatorio.



Haz clic [aquí](#) para ejecutar el bordado del ejemplo.

Consejos de depuración

Si tu bordado no funciona como deseas, verifica primero la consola para ver si hay un error. Puedes intentar arreglar tu código o intentar buscar en Google para encontrar la respuesta. También puedes revisar [este recurso](#) desde p5.js.

- ¿Tu código está dentro de las llaves correctas?
- ¿Hay punto y coma al final de cada línea de código?
- ¿Escribiste correctamente los nombres de la función y de la variable?
- ¿Tus funciones están en la ubicación y secuencia correctas? ¡Recuerda que el orden es importante en el flujo del programa!
- ¿Son correctos los valores de los parámetros en tus funciones? ¿Específicamente las funciones `dist()` y `random()`?
- Imprime los valores en la consola con `print()` y luego marca cualquier afirmación `if`.

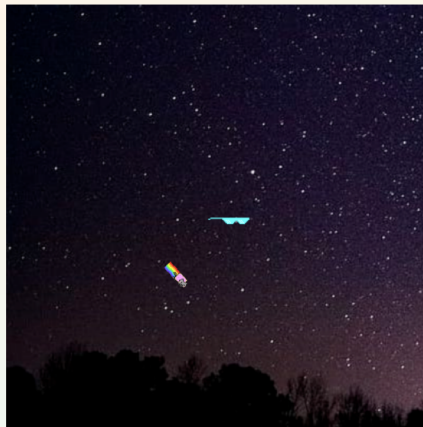
¡Lo hiciste! Terminaste de construir los cimientos de tu proyecto. En la siguiente parte de esta actividad, tendrás la oportunidad de personalizar aún más tu proyecto con algunas extensiones opcionales.



Paso 8: Extensiones (5 a 45 minutos)

Extensión 1: Cambia la historia con gráficos (15 a 45 minutos)

En este momento, el juego se trata de espacio y meteoritos, ¡pero no tiene por qué ser así! Puedes alterar la narrativa del juego agregando nuevos gráficos o imágenes. En el diseño de videojuegos, esto se conoce como personalización de la interfaz. La interfaz de un juego es la apariencia de los componentes y el fondo del juego. Puedes cambiar la interfaz del juego sin cambiar la mecánica, pero esto puede cambiar enormemente el significado del juego y cómo se siente el jugador al respecto.



Haz clic [aquí](#) para ver un bordado de ejemplo de esta extensión

En esta extensión, intenta reemplazar una o más formas y el fondo con imágenes. Estos son los pasos básicos que necesitas para completar esta extensión:

- ☐ Busca archivos de imagen `.png` con fondo transparente. Es posible que debas cambiarles el tamaño en Google Draw y luego exportarlos como un archivo de imagen `.png`.
- ☐ Carga tus archivos `.png` en tu bordado.
- ☐ Declara una variable para almacenar el archivo de imagen.
- ☐ Inicializa cada variable de imagen en `setup()` con la función `loadimage()`.
- ☐ Dibuja las imágenes en la pantalla usando la función `loadimage()` en `draw()`. Asegúrate de usar las variables de posición `x` e `y` correctas.
- ☐ Pasa la variable de imagen de fondo a través de la función `background()`.

Recursos de extensión

A continuación, encontrarás algunos recursos útiles que utilizamos para crear esta extensión. Esto te ayudará a comenzar, pero recuerda que hay muchos más recursos usando el motor de búsqueda.

- [p5.js Web Editor: Carga de archivos multimedia - Tutorial p5.js](#) de The Coding Train (empieza alrededor de 0:59)
- [Cargar y mostrar un ejemplo](#) de imagen de p5.js. Nota: Ignora el texto que dice Para ejecutar este ejemplo localmente, necesitarás un archivo de imagen y un servidor local en ejecución. Esto no se aplica al editor en línea.
- [loadImage\(\)](#)
- [image\(\)](#)
- [imageMode\(\)](#)
- [background\(\)](#)

Este es un enlace a nuestro [código de solución para esta extensión](#). Recomendamos que lo pruebes tú misma primero y que uses este recurso cuando realmente te quedes atascada o desees revisar tu trabajo.

Extensión 2: Seguir la puntuación (5-15 minutos)

En algún momento del tutorial, es posible que te hayas preguntado: “¿Pero qué hay de los puntos?!”. Llevar un registro de la puntuación de un jugador no es necesario para que sea un juego, pero es una forma de darles a los jugadores comentarios inmediatos sobre su juego. Puedes crear una variable para llevar un registro de la puntuación, luego sumar o incrementar el valor de esa variable cada vez que el jugador gane puntos. También podrías hacer que el jugador comience con una puntuación, y luego restar o disminuir un valor cuando realice una acción específica.



Haz clic [aquí](#) para ver un bordado de ejemplo de esta extensión

En esta extensión, intenta crear tu propio sistema de puntuación simple y luego dibuja la puntuación en la pantalla como texto. Estos son los pasos básicos que necesitas para completar esta extensión:

- ☐ **Crea una variable para seguir un registro de la puntuación y ponla a cero.** Pista: ¿Dónde crees que querrías colocar esta variable si quisieras acceder a ella en cualquier lugar de tu bordado?
- ☐ **Incrementa esta variable si el meteorito y el capturador se cruzan.** Pista: ¿En qué parte de tu código verificas si el meteorito y el capturador se cruzan?
- ☐ **Dibuja el valor de la variable en la pantalla como texto.**

Recursos de extensión

A continuación, encontrarás algunos recursos útiles que utilizamos para crear esta extensión. Esto te ayudará a comenzar, pero recuerda que hay muchos más recursos usando el motor de búsqueda.

- [Ejemplo de disminución de incremento](#) de p5.js
- [Ejemplo de palabras](#) de p5.js
- [textSize\(\)](#)
- [text\(\)](#): Puedes combinar [cadenas](#) (es decir, una serie de caracteres de texto) con variables en la función `text()`.
- [textAlign\(\)](#)
- [string](#)

Este es un enlace a nuestro [código de solución para esta extensión](#). Recomendamos que lo pruebes tú misma primero y que uses este recurso cuando realmente te quedes atascada o desees revisar tu trabajo.

Extensión 3: Agregar un estado de ganancia (10-20 minutos)

Ya sean individuales o colaborativos, los juegos a menudo tienen un estado ganador. Los estados ganadores ocurren cuando los jugadores han superado con éxito el desafío principal del juego, como el mayor número de goles o de monedas de oro en 30 segundos o la finalización de una misión espacial. Hay muchas maneras diferentes de ganar un juego.



Haz clic [aquí](#) para ver un bordado de ejemplo de esta extensión

En esta extensión, crea un sistema para probar y alertar a un jugador si ha ganado, luego reinicia el juego para que pueda jugar nuevamente. Antes de comenzar, piensa en lo que significa “ganar” en tu juego. *Por ejemplo, ¿es la cantidad de veces que se juega, la cantidad de puntos, etc.?* En el ejemplo de extensión [aquí](#), usamos la cantidad de puntos anotados para determinar si un jugador ha ganado. Si han ganado y presionan el ratón, el juego se reinicia. Esta extensión requiere que crees una función propia, por lo que si deseas obtener más información sobre cómo definir funciones, ¡esta es una buena oportunidad!

Estos son los pasos básicos que necesitas para completar esta extensión:

- ❑ Crea una función para reiniciar el juego restableciendo la posición, la puntuación y las variables de velocidad a sus valores originales.
- ❑ Crea un condicional para probar si la puntuación es igual a la cantidad de puntos necesarios para ganar.
- ❑ Escribe un mensaje en la pantalla para alertar al jugador de que ha ganado y cómo reiniciar el juego.
- ❑ Crea una segunda sentencia condicional dentro de la primera para probar si se presiona el ratón. Llama a la función de reinicio dentro de ella.

Recursos de extensión

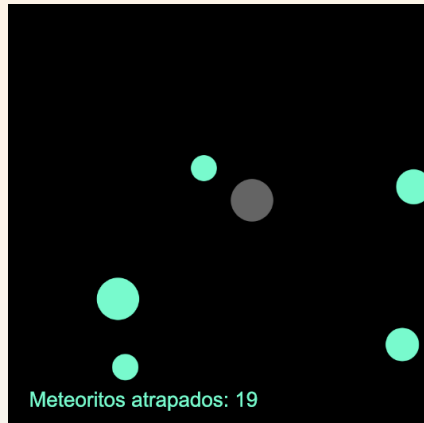
A continuación, encontrarás algunos recursos útiles que utilizamos para crear esta extensión. Esto te ayudará a comenzar, pero recuerda que hay muchos más recursos usando el motor de búsqueda.

- [Conceptos básicos de la función: video tutorial p5.js](#) de The Coding Train
- [función](#)
- [mouseIsPressed](#)
- [Ejemplo de disminución de incremento](#) de p5.js
- [Ejemplo de palabras](#) de p5.js
- [textSize\(\)](#)
- [text\(\)](#): Puedes combinar [cadenas](#) (es decir, una serie de caracteres de texto) con variables en la función `text()`.
- [textAlign\(\)](#)
- [string](#)

Este es un enlace a nuestro [código de solución para esta extensión](#). Recomendamos que lo pruebes tú misma primero y que uses este recurso cuando realmente te quedes atascada o desees revisar tu trabajo.

Extensión 4: Agregar más meteoritos (25-40 minutos)

En la última parte de la fase de construcción, exploramos maneras en las que podíamos hacer que el juego fuera más desafiante al aleatorizar las propiedades del meteorito. Otra manera de hacer que el juego sea más desafiante es hacer más meteoritos.



Haz clic [aquí](#) para ver un bordado de ejemplo de esta extensión

Nota: También incluimos un rastreador de puntuación para que puedas ver cuándo un meteorito se ha intersectado, pero no es necesario que lo hagas para esta extensión.

Hay un par de maneras en las que puedes abordar esto.

- **Enfoque n.o 1:** Crea nuevas variables para un segundo meteorito. Luego, duplica todo el código que escribiste para el primer meteorito, pero actualízalo con variables para el segundo meteorito. Si crees que esto suena tedioso, ¡lo es! Funcionará, pero tu código será largo y engorroso.
- **Enfoque n.o 2:** Usa [matrices](#) y bucles [for](#) para agregar más meteoritos. Las matrices te permiten almacenar una lista ordenada de elementos en una sola variable. Es decir, en lugar de solo almacenar un valor para el diámetro de un meteorito, puede almacenar 10, 15, 20 o 300, y puedes acceder a todos ellos usando el número de índice. Los bucles for te permiten recorrer una sección de código varias veces en función de una condición. Podría decirse que los bucles for y las matrices son mejores amigos, porque puedes usar un bucle [for](#) para iterar o recorrer completamente una [matriz](#). Consulta los **recursos de extensión** a continuación para obtener más información.

En esta extensión, intenta usar matrices y bucles for para formar cuatro meteoritos más para un total de cinco. Las matrices te permitirán almacenar las variables meteorX que necesitas para todos los meteoritos en un solo lugar. Los bucles for te permiten recorrer cada una de esas variables y usarla o realizar una acción en ella.

Estos son los pasos básicos que necesitas para completar esta extensión:

- ❑ Declara **matrices** para almacenar la posición X del meteorito, la posición Y del meteorito (todos estos valores deben ser 0), el diámetro del meteorito, la distancia y la velocidad.
- ❑ Crea bucles **for** en **setup()** que completarán automáticamente las siguientes matrices con valores aleatorios iniciales: posición X, diámetro del meteorito y velocidad.
- ❑ Crea un bucle **for** que dibuje los meteoritos en la pantalla.
- ❑ Crea un bucle **for** que recorra todos los meteoritos para que caigan a diferentes velocidades.
- ❑ Dibuja el capturador.
- ❑ Crea un bucle **for** que recorra todos los meteoritos para determinar la distancia entre cada meteorito y el capturador.
- ❑ **Crea un bucle **for** que recorra todos los meteoritos para probar si uno de los meteoritos se ha intersectado con el capturador.** Si se ha intersectado, vuelve a dibujar ese meteorito en la parte superior de la pantalla en una posición X aleatoria y establece una nueva velocidad aleatoria para ese meteorito.

Consejo: Deberás agregar una sentencia condicional dentro del bucle for.

- ❑ **Crea un bucle **for** que recorra todos los meteoritos para probar si uno de los meteoritos se ha intersectado con la parte inferior de la pantalla.** Si se ha intersectado, vuelve a dibujar ese meteorito en la parte superior de la pantalla en una posición X aleatoria y establece una nueva velocidad aleatoria para ese meteorito.

Consejo: Deberás agregar una sentencia condicional dentro del bucle for.

Recursos de extensión

A continuación, encontrarás algunos recursos útiles que utilizamos para crear esta extensión. Esto te ayudará a comenzar, pero recuerda que hay muchos más recursos usando el motor de búsqueda.

→ Videos tutoriales de p5.js de The Coding Train

Nota: Algunos de estos videos se crearon antes de que se creara el editor en línea, ¡pero funcionarán de la misma manera!

◆ [¿Qué es una matriz?](#)

◆ [matrices y bucles](#)

◆ [bucles while y for](#)

→ [Tutorial sobre el flujo del programa](#)

→ [Tutorial sobre la iteración](#)

→ **for** Si tienes problemas para comenzar, intenta usar el primer enfoque y crea un segundo meteorito duplicando el código del primer meteorito. Cada vez que veas uno doble (p. ej. meteorY_1 y meteorY_2 o speed_1 y speed_2) es probable que necesites una matriz y un bucle for para recorrerlo.

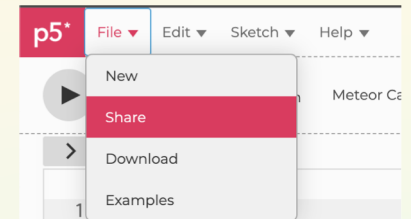
Este es un enlace a nuestro [código de solución para esta extensión](#). Recomendamos que lo pruebes tú misma primero y que uses este recurso cuando realmente te quedes atascada o desees revisar tu trabajo.

Paso 9: Compartir tu proyecto de Girls Who Code en casa (5 minutos)

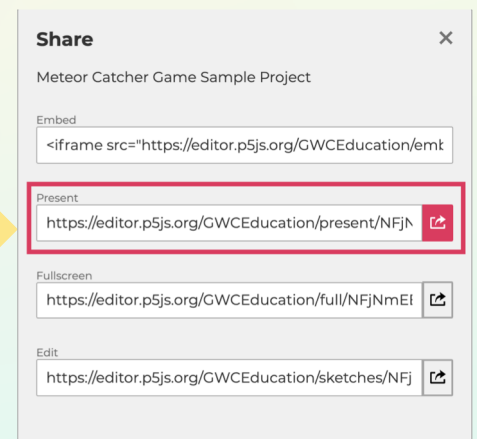
Nos encantaría ver tu trabajo y sabemos que a otros también les gustaría. ¡Comparte tu juego con nosotros! No olvides etiquetar [@girlswhocode](#) [#codefromhome](#), ¡y quizá te destaquemos en nuestra cuenta!

Sigue estos pasos para compartir tu proyecto:

- Primero guarda tu proyecto.
- 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!

