



Girls Who Code At Home

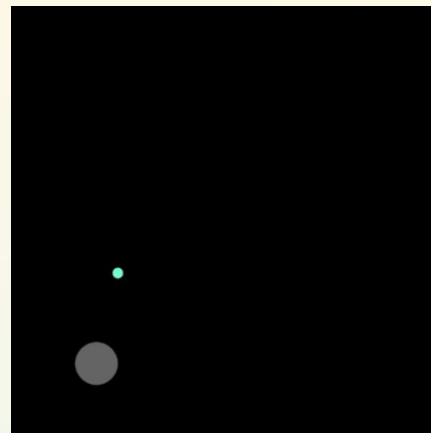
流星キャッチャーゲーム:パート 5

ランダムイザーを追加する

学習概要

この最終章では、ゲームをよりチャレンジングに、そしてより楽しくするための方法を探ります。[random\(\)](#)関数を使って流星の挙動に複雑性を持たせる方法を学びます。最後に、私たちの拡張機能を1つまたは複数試して、プロジェクトをカスタマイズしてください。[ここ](#)をクリックすると、このアクティビティが終了するまでに学ぶ内容をプレビューできます。

アクティビティに取り掛かる前に、「流星キャッチャーゲーム」の[パート 1](#)、[パート 2](#)、[パート 3](#)、[パート 4](#)を既に完了している必要があります。



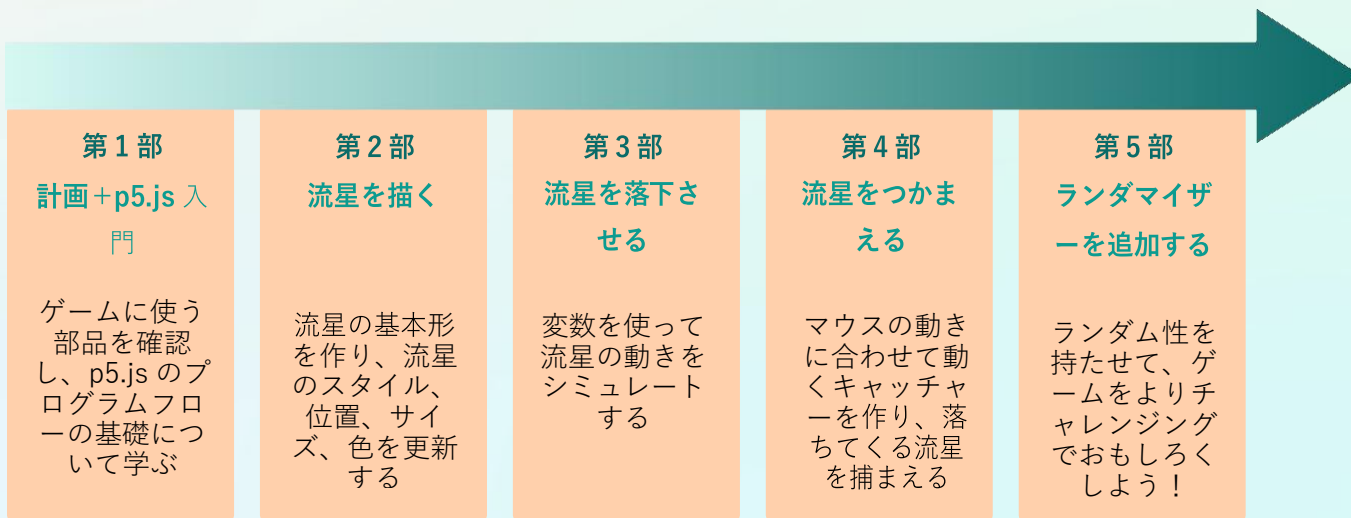
学習目標

このアクティビティが終わるころには、次のことができるようになります。

- [random\(\)](#) 関数を使用して、ゲームの難易度を上げたり下げたりする方法を説明する。

マテリアル

- [p5.js オンラインエディタ](#)
- [流星キャッチャーゲーム サンプルプロジェクト](#)
- [流星キャッチャーゲーム パート 5 リファレンスガイド](#)



流星キャッチャーゲームの[動画チュートリアル](#)のパート 5 を参照しながら、すすめることもできます。

Women in Tech Spotlight: Lisette Titre-Montgomery (リゼット・ティトル=モンゴメリーさん)



画像の出展はこちら: [リゼット・ティトル=モンゴメリー](#)

Dance Central で音楽を聴いたり、The Sims でキャラクターを作ったりするとき、そこに至るまでの技術について考えたことはありますか？リゼット・ティトル=モンゴメリーは、市場で最も人気のあるゲームを支えるエンジニア兼デザイナーとして働いています。17 年以上のゲーム業界での経験と 13 ゲームタイトルを世に送り出した実績を持つ彼女は、ビデオゲーマーと開発者のギャップを埋めることに成功しています。

また、メンタリングやホワイトハウスとのインクルーシブな雇用のための取り組みを通じて、ゲーム業界にさらなる多様性をもたらすことにも力を注いでいます。そして、カリフォルニア州オークランドを拠点とする Gameheads というプログラムでは、K-12 教育におけるゲームを活用したカリキュラムを提唱し、生徒の STEAM 教育とキャリアへの関心を高める活動を続けています。

この [ビデオ](#) では、リゼットが様々なゲーム制作で直面した課題と、それをどのように克服したかを紹介しています。

リゼットと彼女の作品についてもっと知りたいですか？

- [彼女の個人ウェブサイト](#) では、様々な作品やプロジェクトについて詳しく知ることができます。
- リゼットの経歴やゲーム業界に入った経緯はこちらの [記事](#) をご覧ください。
- [リゼットのプロフィール](#) を見て、彼女のキャリアのハイライトと、ゲーム業界を目指す学生へのアドバイスをご覧ください。

考えてみましょう

コンピュータサイエンティストであることは、単にコーディングが得意というだけではありません。リゼットと彼女の仕事が、偉大なコンピュータ科学者が築くことに注力している強み、すなわち勇気、レジリエンス、創造性、目的にどのように関連しているのか、時間をかけて考えてみてください。



勇気

リゼットは最初の仕事をするまで、ゲーム業界で働いたことがありませんでした。彼女はどのようなステップを踏んで、この仕事を成功させたのでしょうか？

あなたの回答を家族や友人と共有しましょう。他の人にもリゼットについて知ってもらい、議論に参加してもらいましょう

ステップ 1：ゲームの課題を振り返る（2～4 分）

今、私たちはゲームの重要な部分をすべてカバーしています。つまり、構成要素、コアメカニク、ゲーム、スペースです。しかし、このゲームはまだあまり面白くありません。その主な理由は、難易度が低いためです。ゲームデザイナーにとって、適切な難易度を見つけることは重要です。難易度を上げるために使える道具の 1 つは（たくさんありますが）、ランダム性の要素を含めることです。これは、プレイヤーが次に何が起こるか予想できないように、新しい情報で驚かせ、それに適応させるものです。

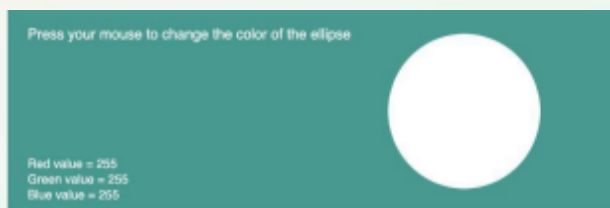
ゲームにランダム性を持たせて、よりチャレンジングにするにはどこがいいでしょうか？ 1 分間で考えて、あなたのアイデアとこのあと紹介するアイデアとを比べてみてください。



リファレンスガイドの ページ 2 をつかい、自分の考えを確認してみましょう！

ステップ 2：random()関数の探索(5 分)

コードにランダム性を持たせるために、**random()**関数を使用することができます。この関数は、指定した範囲の値の間でランダムに浮動小数点数を返す（出力する）関数です。浮動小数点とは、数値が小数を含む可能性があることを意味します。これにより、より広域に分布した数値を得ることができます。この[スケッチの例](#)では、キャンバス内でマウスをクリックするたびに、**fill()**関数の赤、緑、青の値としてランダムな値が生成される様子をご覧ください。



random()関数の構文を見てみましょう。

JAVASCRIPT	DESCRIPTION
<code>random(min, max);</code>	<ul style="list-style-type: none">→ random: 関数名→ ():括弧を使用して、プログラムに関数を呼び出す必要があることを知らせます。時には、関数のパラメータや入力を括弧の中に入れることもあります。→ min: ランダム範囲の下限值（この数値を含む）→ max: ランダム範囲の上限値（この数値を含まない）→ ∴: JavaScript のコード行はすべてセミコロンで終わらなければなりません

ステップ 3: 擬似コードの更新 (2-4 分)

新しいコードを追加する前に、まず人間の言葉で何が起きてほしいかをブレインストーミングしてみましょう。それぞれの `random()` 関数の疑似コードを書いてください。できるだけ具体的に、指定された値の範囲を使うようにしてください。完成したら、あなたの答えを以下でチェックしてください。

- 流星のスタート位置。値の範囲 0~400
- 流星の速度。値の範囲 0.5~4
- 流星の大きさ。値の範囲 10~30



リファレンスガイドのページ 2 でコードを確認することを忘れないでください。

ステップ 4: コードの追加 (5~8 分)

では、疑似コードを実際のコードに変換していきましょう。書くコードはすべて、既存の記述の下にある両方の条件文の中に入ります。今は、流星がキャッチャーや底面の壁と交差した場合にのみ、画面の上部に流星を再描画していますね。この条件文に `random()` のステートメントを追加することで、新しい流星の位置、速度、大きさがランダムになります。

最初の条件文の位置を確認します。 `meteorY = 0;` の下に以下の新しい行を追加してください。:

- ❑ 流星のスタート位置を 0 から 400 の間の値でランダム化するコード行を書きましょう。
- ❑ 流星の速度を 0.5 から 4 の間の値でランダム化するコードを書きます。
- ❑ 流星の大きさを 10 から 30 の間の値でランダム化するコード行を書きます

2 つ目の条件文の位置を確認します。

- ❑ 先ほど、一つめの条件文で書いた記述をコピーして、2 つ目の条件文の `meteorY = 0;` の下にコピーしてください。

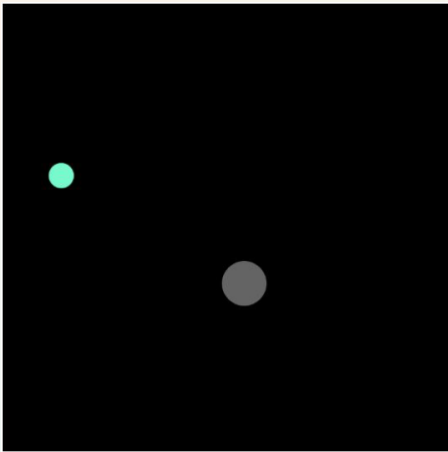
条件文の中のコード行が同じに見えることにお気づきでしょうか？ そうなんです。これらの条件文は、論理演算子「`or`」- `||` で結合することができます。また、それらを入れ子にして組み合わせることもできます。私たちは、拡張機能をもう少しわかりやすくするために、これらを分離することを選択しました。もし、これらの条件式を組み合わせたいのであれば、どうぞ自由に挑戦してみてください！

ステップ 5: コードのテスト (5-10 分)

ここまでに書いたものをテストして、プログラムが思い通りに動くことを確認しましょう。再生ボタンをクリックして、スケッチを実行してください。

コードを実行し、以下のようにテストしてください:

- ❑ 流星とキャッチャーが交差するようにします。それらが交差すると、現在の流星が消え、「新しい」流星がランダムな x 位置に、新しいサイズと新しい速度で出現するはずです。
- ❑ 流星とキャンバスの底が交差するのを待ちます。それらが交差すると、現在の流星が消えて、「新しい」流星がランダムな x 位置に、新しいサイズと新しい速度で現れるはずです。



サンプルスケッチを実行するには、[ここ](#)をクリックしてください。

思い通りに動作しませんでしたか？では、デバッグのコツをご紹介します。

- あなたのコードは正しい波括弧 { } の中にありますか？
- コードの各行の末尾にセミコロン ; をつけていますか？
- 関数は正しい場所に配置されていますか？プログラムの流れは順序が重要であることを忘れないでください。
- `random()` 関数のパラメータは正しいですか？



リファレンスガイドの **3** ページでコードを確認することを忘れないでください。

ステップ7：コードのテストとフィードバックの受信（5-10 分）

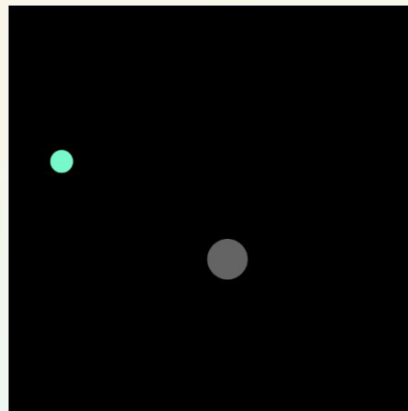


プロジェクトのコア部分の構築が完了し、大きなマイルストーンに到達しました。次に進む前に、あなたのゲームが正しく動作していることを確認するため、コードをテストします。これは、友人や家族からあなたのゲームに対するフィードバックを得る絶好の機会でもあります。あなたのプロジェクトとこのチェックリストをテスター（テストをしてくれる人）と共有してください。

チェックリスト

スケッチを実行し、チェックリストでコードをテストしてください。:

- ❑ 400×400 のキャンバスがあります。背景はあなたの選んだ色で塗りつぶされています。
- ❑ ゲームは、キャンバス上部の中央から流星（つまり楕円）がゆっくりと落下してくるところから始まります。流星には輪郭がなく、あなたの選んだ色で塗りつぶされます。
- ❑ 白色で半透明のキャッチャー（楕円形など）があります。キャッチャーの中心は、マウスの動きに追随します。
- ❑ あなたのプログラムは、流星とキャッチャーの間の距離をコンソールに表示します。
- ❑ 流星とキャッチャーが交差したら、ゲームは新しい流星を描画します。
- ❑ 流星とキャンバスの底が交差したら、ゲームは新しい流星を描画します。
- ❑ 新しい流星はキャンバスの上部から、x 軸上のランダムな位置で、ランダムな速度とランダムな大きさに始まります。



サンプルスケッチを実行するには、[ここ](#)をクリックしてください。

デバッグのヒント

もしあなたのスケッチが思い通りに動かないなら、まずコンソールをチェックして、エラーがあるかどうか確認してください。あなたは、コードを修正するか、答えを見つけるためにインターネット検索を試みることができます。また、p5.js の[このリソース](#)を確認することもできます

- ➔ あなたのコードは正しい波括弧 { } の中にありますか？
- ➔ コードの各行の末尾にセミコロン ; をつけていますか？
- ➔ 変数名や関数名のスペルは正しいですか？

- ➔ 関数は正しい位置と順序で配置されていますか？プログラムの流れは順序が重要であることを忘れないでください。
- ➔ 関数のパラメータ値は正しいですか？具体的には、`dist()` と `random()` の関数です。
- ➔ `print()` でコンソールに値を出力し、`if` 文のチェックを行います。

やりましたね！プロジェクトの土台作りが完了しました。このアクティビティの次のパートでは、あなたのプロジェクトをさらに個性的にするために、オプションな

拡張機能を使って、いくつかの工夫を追加することができます。



ステップ 8：拡張機能（5～45 分）

拡張機能 1：グラフィックでストーリーを変える（15 分～45 分）

今、このゲームは宇宙と流星に関するものですが、そうである必要はありません。新しいグラフィックや画像を追加することで、ゲームの物語を変化させることができます。ゲームデザインでは、これを「スキニング」と呼んでいます。ゲームのスキンとは、ゲームの構成要素や背景の外観のことです。メカニクスを変えずに「リスキン（外観を変える）」を行うこともできますが、これによりゲームの意味やプレイヤーの感じ方を大きく変えることができます。



この拡張機能のスケッチ例は [こちら](#) からご覧いただけます。

この拡張機能では、1 つまたは複数の図形と背景に置き換えてみてください。ここでは、この拡張機能を完成させるために必要な基本的な手順を説明します。

- ❑ 背景が透明な **.png** 画像ファイルを探します。選択した画像によっては、Google Drawing(Google 図形描画)でサイズを変更し、**.png** 画像ファイルとしてエクスポートする必要があるかもしれません。
- ❑ **.png** ファイルをスケッチにアップロードします。
- ❑ 画像ファイルを格納する変数を宣言します。
- ❑ **setup()**内の各画像変数を **loadImage()**関数 で初期化します。
- ❑ **draw()** の **image()** 関数を使用して、画像を画面に描画します。必ず正しい x と y の位置変数を使用してください
- ❑ **background()**関数を使い、背景画像変数を渡しましょう。

この拡張機能のソリューションコードへのリンクは [こちら](#) です。まずはご自身で試してみてください。本当に困ったときや、自分の作ったコードをチェックしたいときに、このリソースをご利用ください。

拡張機能リソース

以下は、この拡張機能を作成するために使用した、いくつかの有用なリソースです。これらはあなたが作業を始めるのに役立ちますが、検索エンジンを使えばもっとたくさんのリソースを活用することができることも忘れないでくださいね。

[p5.js のウェブエディタ：メディアファイルのアップロード \(p5.js Tutorial\)](#) by The Coding Train (0:59 頃開始)

p5.js から [画像を読み込んで表示する例](#) です。注：この例をローカルで実行するには、画像ファイルとローカルサーバーが必要です。これはオンラインエディタには適用されません

[loadImage\(\)](#)

[image\(\)](#)

[imageMode\(\)](#)

[background\(\)](#)

拡張機能 2：スコアを追う（5～15 分）

チュートリアルの中で、"でも、点数ってどうなの?"と疑問に思ったかもしれません。プレイヤーのスコアを記録することは、ゲームとして成立させるために必須ではありませんが、プレイヤーにゲームプレイに関するフィードバックを即座に与える方法の 1 つではあります。スコアを記録する変数を作成し、プレイヤーがポイントを獲得するたびに、その変数の値を追加または増加させることができます。また、プレイヤーにある点数からスタートさせ、特定の行動をとったときに点数を減少させることもできます。



この拡張機能のスケッチ例は[こちら](#)からご覧いただけます。

このエクステンションでは、簡単な採点システムを自分で作り、そのスコアをテキストとして画面に描画してみてください。このエクステンションを完成させるために必要な基本的な手順は以下の通りです。:

- ❑ **スコアを記録するための変数を作成し、それをゼロに設定します。** ヒント：この変数をスケッチのどこからでもアクセスできるようにするには、どこにこの変数を置けばいいと思いますか？
- ❑ **流星とキャッチャーが交差したら、この変数を加算します。** ヒント：コードのどこで流星とキャッチャーが交差しているかどうかをチェックしていますか？
- ❑ **変数の値をテキストとして画面に描画します。**

この拡張機能のソリューションコードへのリンクは[こちら](#)です。まずはご自身で試してみてください。本当に困ったときや、自分の作ったコードをチェックしたいときに、このリソースをご利用ください

拡張機能リソース

以下は、この Extension を作成するために使用した、いくつかの有用なリソースです。これらはあなたが作業を始めるのに役立ちますが、検索エンジンを使えばもっとたくさんのリソースを活用することができることも忘れないでくださいね。

[インクリメント・デクリメント例](#) p5.js より

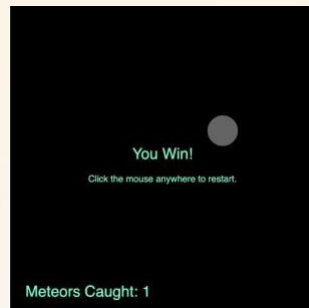
p5.js [単語例](#)

[testSize\(\)](#)

[text\(\)](#) : text() 関数では、[文字列](#) (つまり一連のテキスト文字) と変数を組み合わせることができます。

拡張機能 3：勝利状態の追加（10 分～20 分）

個人であれ共同であれ、ゲームにはしばしば勝つという状態が存在します。この勝つという状態は、プレイヤーがゲームの主要な課題をうまく克服したときに発生します。例えば、30 秒間の最高ゴール数、最高金貨数、宇宙ミッションの完了などです。ゲームにはさまざまな勝ち方があります。



この拡張機能のスケッチ例は[こちら](#)からご覧いただけます。

この拡張機能では、プレイヤーが勝利した場合にテストして警告し、ゲームをリセットして再びプレイできるようにするシステムを作成します。始める前に、あなたのゲームにおいて「勝利」が何を意味するのか考えてみてください。例えば、プレー回数なのか、ポイント数なのか、などです。[この拡張例](#)では、プレイヤーが勝ったかどうかを得点数で判断しています。もしプレイヤーが勝利し、画面上でマウスを押せば、ゲームが再開されます。このエクステンションでは独自の関数を作成する必要があります。

ここでは、この拡張機能を完成させるために必要な基本的な手順を説明します。

- ❑ 流星の位置、得点、速度変数を元の値にリセットして、ゲームを再開する関数を作成します。
- ❑ スコアが勝利に必要なポイント数と等しいかどうかをテストする条件文を作成します。
- ❑ 画面に勝ったことを知らせるメッセージとゲームの再開方法を書き込みます
- ❑ マウスが押されたかどうかをテストするために、最初の条件の中に 2 つ目の条件文を作成します。その中で restart 関数を呼び出します。

拡張機能リソース

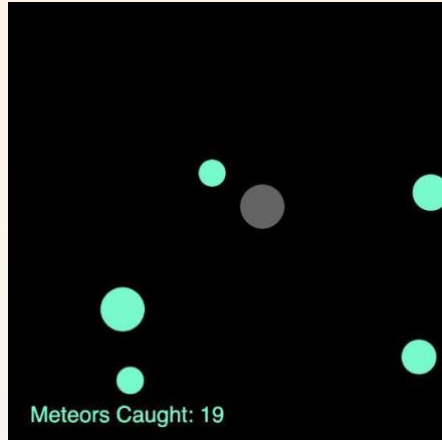
以下は、この Extension を作成するために使用した、いくつかの有用なリソースです。これらはあなたが作業を始めるのに役立ちますが、検索エンジンを使えばもっとたくさんのリソースを活用することができますことも忘れないでくださいね。

- [関数の基本 - p5.js チュートリアルビデオ](#)（The Coding Train より）
- [function](#)
- [mouseIsPressed](#)
- p5.js の[インクリメント・デクリメントの例](#)
- p5.js の[単語例](#)
- [textSize\(\)](#)
- [text\(\) :text\(\)](#) 関数では、[文字列](#)（つまり一連のテキスト文字）と変数を組み合わせることができます。
- [textAlign\(\)](#)
- [文字列](#)

この拡張機能のソリューションコードへのリンクは[こちら](#)です。まずはご自身で試してみてください。本当に困ったときや、自分の作ったコードをチェックしたいときに、このリソースをご利用ください

拡張機能 4：流星を増やす（25 分～40 分）

構築段階の最後には、流星の性質をランダムにすることで、ゲームをよりチャレンジングにする方法を探りました。また、流星をたくさん作ることで、ゲームの難易度を上げることもできます。



この拡張機能のスケッチ例は[こちらから](#)ご覧いただけます。

注：流星がいつ交差したかを確認できるように、スコアトラッカーも搭載していますが、この拡張機能では必須ではありません。

これには、いくつかの方法があります。

- アプローチ #1: 2 つ目の流星用に新しい変数を作成します。そして、1 つ目の流星で書いたコードをすべて複製し、2 つ目の流星用の変数で更新します。この手順が面倒くさいと思いましたか？ はい、あなたは正しいです。この手順はうまくいきますが、コードは長く、面倒になります。
- アプローチ 2：[配列](#)と [for ループ](#)を使って流星を増やしましょう。配列は、要素の順序付きリストを一つの変数に格納することを可能にします。つまり、流星の直径を 1 つだけ格納するのではなく、10 個、15 個、20 個、300 個と格納し、配列のインデックス番号を使って格納された値すべてにアクセスすることができます。for ループは、ある条件に基づいてコードの一部を何度もループさせることができます。for ループを使用すると、[配列](#)を繰り返し処理したり循環させたりすることができるので、for ループと配列は非常に相性がいいと言えるでしょう。詳しくは、以下の拡張機能リソースを参照してください。

この拡張機能では、配列と for ループを使って、さらに 4 つの流星を作り、合計 5 つの流星を作ってみましょう。配列は、すべての流星に必要な meteorX 変数を一箇所に格納することができます。For ループを使用すると、これらの変数を循環させて使用したりアクションを実行したりすることができます。

ステップ 8：拡張機能（続き）

ここでは、この拡張機能を完成させるために必要な基本的な手順を説明します。

- ❑ 流星の X 位置、Y 位置(この値は全て 0 でなければなりません)、流星の直径、距離、速度を格納する **配列**を宣言してください。
- ❑ **setup()** の中で **for** ループを作成し、以下の配列に X 位置、流星の直径、速度のランダムな初期値を事前投入します。
- ❑ 流星を画面に描画する **for** ループを作成する。
- ❑ 流星を循環させ、異なる速度で落下させる **for** ループを作成する。
- ❑ キャッチャーを描く。
- ❑ 各流星とキャッチャー間の距離を決定する **for** ループを作成します。
- ❑ 流星の **1** つがキャッチャーと交差したかどうかをテストする **for** ループを作成します。交差した場合、その流星を画面上部のランダムな X 位置に再描画し、その流星に新しいランダムな速度を設定します。ヒント：*for* ループの中に条件文を追加する必要があります
- ❑ 流星の **1** つが画面の下部と交差しているかどうかをテストする **for** ループを作成します。交差した場合、その流星を画面上部のランダムな X 位置に再描画し、その流星に新しいランダムな速度を設定します。ヒント：*for* ループの中に条件文を追加する必要があります。

拡張機能リソース

以下は、この Extension を作成するために使用した、いくつかの有用なリソースです。

これらはあなたが作業を始めるのに役立ちますが、検索エンジンを使えばもっと

たくさんのリソースを活用することができることも忘れないでくださいね。

p5.js のチュートリアルビデオ (The Coding Train より)

注：これらのビデオのいくつかは、オンラインエディタが構築される前に作成されたものですが、同じように機能します

[アレイとは何ですか？](#)

[配列とループ](#)

[while and for ループ](#)

[プログラムフロー チュートリアル](#)

[イテレーションチュートリアル](#)

for: もし、うまくいかない場合は、最初のアプローチを使って、

最初の meteor のコードを複製して 2 番目の meteor を作ってみてください。

ダブル (例えば **meteorY_1** と **meteorY_2** や **speed_1** と **speed_2**) を見るたびに、配列

とそれを循環させるための for ループが必要になる可能性があります。

この拡張機能のソリューションコードへのリンクは[こちら](#)です。まずはご自身で試してみてください。

本当に困ったときや、自分の作ったコードをチェックしたいときに、このリソースをご利用ください。

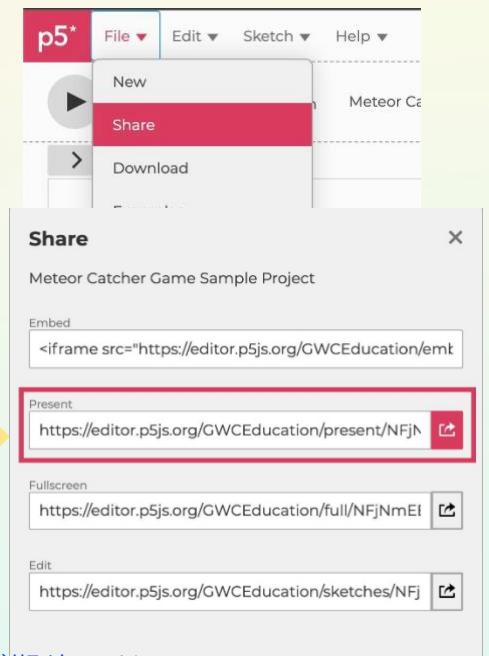
ステップ 9 : Girls Who Code at Home プロジェクトを共有しよう (5 分)

私たちは皆さんの作品を見るのが大好きで、他の人もそうであることを知っています。あなたのゲームを私たちとシェアしてください。また、[@girlsswhocode](#) [#codefromhome](#) のタグをお忘れなく！ 私たちのアカウントで紹介するかもしれません。

以下の手順でプロジェクトを共有することができます。

- 最初にプロジェクトを保存してください。
- **File** メニューのドロップダウンメニューから、「**Share**」を選択します。
- ドロップダウンメニューから「**Link**」を選択します。
- **Present** リンクをコピーして、共有したい場所に貼り付けてください。

Project



Girls Who Code at Home の今後のプロジェクトにご期待ください。

