

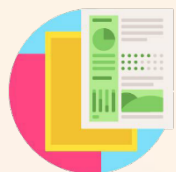


Girls Who Code At Home

流星キャッチャーゲーム: Part 1

リファレンスガイド

流星キャッチャーゲーム: Part 1 – リファレンスガイド



このドキュメントには、アクティビティで出題されるいくつかの問題に対するすべての回答が掲載されています。アクティビティに沿って進み、このアイコンを見たら、立ち止まって自分の考えを確認してみてください。

ステップ1: ゲームのパーツを特定する

流星キャッチャーゲームのパーツ

- 最後のステップで遊んだゲーム、「流星キャッチャー」の**ゴール**を説明してください。ゲームに勝つためには、プレイヤーやチームは何をしなければならないのでしょうか？
「流星キャッチャー」のゴールは、できるだけ多くの流星をキャッチすることです。今のところ、このゲームにはあまり明確なゴールはありません。基本となるゲームを作った後に、自分なりにカスタマイズできるように、自由をしています。
- 流星キャッチャーの**構成要素**には、流星、キャッチャー、壁・床、プレイヤーがあります。各要素には特有の性質（大きさ、色、形など）と動作（その構成要素から連想される動詞）があり、ゲームのシステムに寄与しています。例えば、流星の性質は丸で、動作は画面の上から下へ落ちるというものです。

下の表の各構成要素の性質と動作について、2～3分考えてみてください。

構成要素	性質	動作
ゲームのプレイに欠かせないものとは？	構成要素の属性や特徴はなんですか？	それは何をするものですか？ どんな動詞を連想しますか？
流星	<ul style="list-style-type: none">→ 丸→ ティール（鴨の羽色）→ 10～40ピクセルの任意の直径	<ul style="list-style-type: none">→ 画面上部から下部へ落ちる→ ランダムな速度で移動する→ 画面上部の異なる位置で起動する→ キャッチャーと交わることができる→ 地面と交わることができる（例：画面の下）
キャッチャー	<ul style="list-style-type: none">→ 丸→ 透明な白色→ 直径 40 ピクセル	<ul style="list-style-type: none">→ マウスに従って動く→ 流星と交差することで、流星を "キャッチ"、または "コレクション" できる
壁	<ul style="list-style-type: none">→ 横400ピクセル→ 縦400ピクセル	<ul style="list-style-type: none">→ 流星と交差する→ 一番下の壁は、流星に触れると新しい流星が出現します
プレイヤー	<ul style="list-style-type: none">→ 宇宙が好き！流星群も！	<ul style="list-style-type: none">→ マウスを動かして、落ちてくる流星をキャッチします

- ゲームの**舞台**について説明してください。どこで行われますか？(空間が複数のものになることもあることに注意してください。例えば、チェスはチェス盤の上で行われますが、リビングルーム、公園、カフェテリアなどでも行われます。)

このゲームの舞台は、ウェブページ上の400×400の正方形のウィンドウです。ストーリーの内容としては、実際の宇宙空間が舞台になっています。

- **チャレンジ**を設定しましょう。プレイヤーがゴールにたどり着くまでに、どんな難関がありますか？

落ちてくる流星をキャッチすることがチャレンジです。流星はそれぞれ異なる位置から、異なる速度で落下し、新しい流星は画面に現れるたびに異なるサイズになります。

- ゲームの**コアメカニク**を説明してください。ゲームをプレイするために、プレイヤーはどのような中心となる行動や動きをする必要がありますか？

プレイヤーはマウスを画面上で動かして、流星を集めます。

- ゲームのルールの一覧を書きます。ルールは、ゲームで何ができて、何ができないかを定めるものです。プレイヤー、構成要素、舞台などに適用することができます。

- ◆ 流星は画面の上から下へ落ちていく。
- ◆ 一度に落ちてくる流星は1つだけ。
- ◆ キャッチャーはマウスに従う。
- ◆ プレイヤーのキャッチャーが流星をキャッチするためには、流星と交わる必要がある。
- ◆ 流星をキャッチすると消え、画面上部に新たな流星が描かれる。
- ◆ 流星が画面下に触れると消え、画面上部に新たな流星が描かれる。

ステップ 2: ゲームの疑似コードを書く

任意の変数を宣言する

これを一度だけ実行する

キャンパスの大きさを400ピクセル×400ピクセルに設定する

これをループ毎に行う

背景色を設定する

流星を描く

流星を落下させる

マウスに従うキャッチャーを描く

流星とキャッチャーの距離を求める/計算する

流星とキャッチャーが交差しているかどうかをテストする。もし交差していたら、
画面上部の流星をランダムな位置に再描画し、

新しい速度を与える

新しい直径を設定する

流星と底面の壁が交差しているかどうかをテストする。もし交差していたら、
画面上部の流星をランダムな位置に再描画し、

新しい速度を与える

新しい直径を設定する

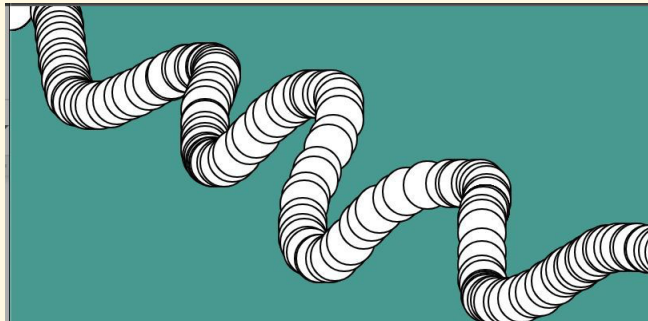
覚えておくこと:

プログラムを書く方法は1つではないので、

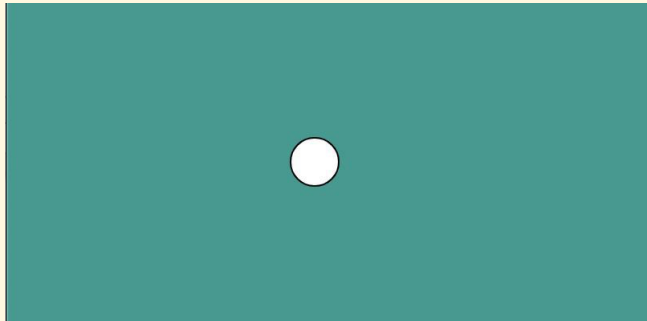
あなたが違う疑似コードを書く方法もたくさんあります。

ステップ5: プログラムフローを学ぶ

スケッチI



スケッチ2



スケッチ1では、`setup()`の中に`background()`関数があります。つまり、背景は1回だけ描かれます。`ellipse()`関数は`draw()`の中にあるので、p5はプログラムがループするたびに、画面上のマス位置に新しい円を描いています。このように表現することができます：背景を塗りつぶす、円を描く、円を描く、円を描く、円を描く、円を描く、円を描く、円を描く、円を描く、円を描く…

```
// Sketch 1
function setup() {
  createCanvas(400, 400);
  background(220);
}

function draw()
{ ellipse(mouseX,
  mouseY,50,50);
}
```

スケッチ2では、`draw()`の中に`background()`関数があります。これは、プログラムがループするたびに背景と円を描くことを意味します。したがって、最初のスケッチと同じようにプログラムが新しい円を描いているにもかかわらず、カーソルを動かすと円が空間内を滑らかに移動しているように見えるのです。このように表現することができます：背景を塗りつぶす、円を描く、背景を塗りつぶす、円を描く、背景を塗りつぶす、円を描く、背景を塗りつぶす、円を描く、背景を塗りつぶす…

```
// Sketch 2
function setup() {
  createCanvas(400, 400);
}

function draw()
{
  background(220);
  ellipse(mouseX, mouseY, 50, 50);
}
```

ステップ 6: 理解度の確認

餃子をたくさん作る時でも、**setup()**のアクションは、一回だけ起こせば大丈夫です。**draw()**のアクションは、餃子を作るたびに実行される必要があります。複数の餃子を作るので、これらのアクションを **draw()** に配置します。

```
setup() {  
  具の材料を計量する  
  具の材料を混ぜる  
  餃子の皮を用意する  
}  
  
draw() {  
  中身をスプーンで包む  
  包みを閉じる  
  餃子をフライパンに入れる  
  餃子を調理する  
  フライパンから餃子を取り出す  
  餃子を食べる  
}
```