やってみた・やってはいけない chatGPT in プログラミング教育

下條真司

2025/1/14

きっかけ

ChatGPTを使ったプログラミング教育のパラダイムシフト：100日連続アプリ作成の達成

中央大学　経済学部　伊藤 篤
（株）ゼンリンデータコム　大塚 あみ

2024.9.3
NWS研究会

https://www.nii.ac.jp/event/upload/20240903-5_ito_ootsuka.pdf

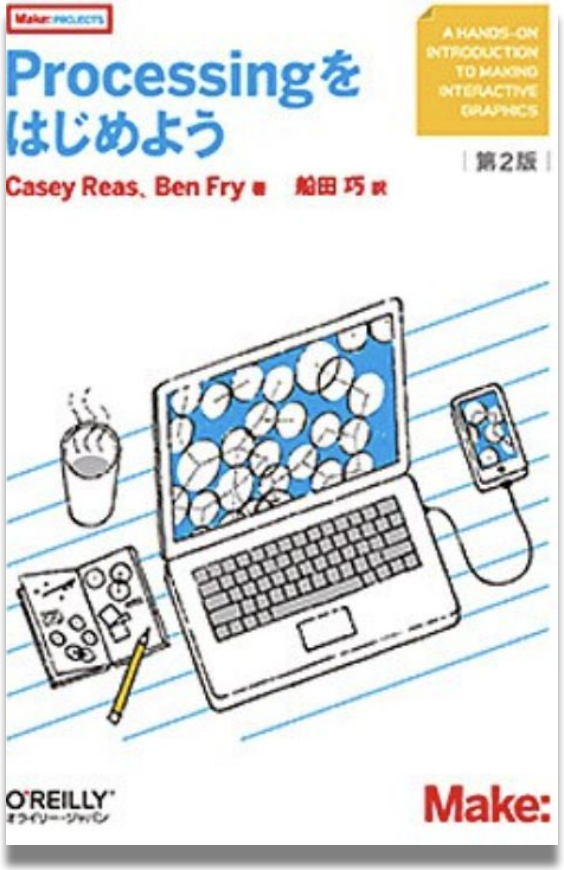
https://ascii.jp/elem/000/004/185/4185964/

対象

青森大学ソフトウェア情報学部　1年生　プログラミング演習II

　Iでprocessingの一通りを学習

　ベーシック（9），ノーマル，アドバンスト　にクラス分け



プログラミングを使ってやりたいことを書いてください.	プログラミングで苦手なところ，わかりにくいことを書いてください
pythonを使いAiを上手く使えるようにしたい	コードがわからない
特にやりたいことはないけど、ある程度はできるようになりたい	簡単な部分は理解できるけど、難しくなったり、前期の最後の部分などがわからなかった。
ゲーム作成	変数の意味が理解しきれていない　組み合わせが理解できていない　基礎がわかっていない
探索RPG作りたい簡単なやつ	何が何の変数（？）関数（？）がわからない 関数（？）のあとに続けて書くやつがわからない
特にやりたいことがないんですけど、プログラミングが苦手なので、ある程度できるようになりたいです。	コードの書き方やコードの意味がいまいちまだ分からないのでそこを克服したいです。
ロボット動かす	コードを書くことが苦手
スムーズに事が進むようにしたい	プログラミングの文面
いろんなチートを作ってみたい	英語が理解できでいない

やったこと―chatgpt（無料版）を利用して

アニメーションを作ってみよう

シューティングゲームを作ってみよう

ボードゲームを作ってみよう

リンクを共有して，提出

コーディングにchatgptを使う注意点 (by chatgpt)

技術的な注意点

1. **コードの検証を必ず行う**
   - ChatGPTが生成したコードは必ず実行してテストする。不正確なコードやエラーが含まれる可能性がある。
2. **最新の情報を確認する**
   - ライブラリやフレームワークの仕様が変わっている場合があるので、公式ドキュメントで確認する。
3. **エラーのデバッグ**
   - ChatGPTはエラーの文脈を正確に理解しない場合があるため、生成されたコードでエラーが発生したら自分で問題を解析する。
4. **セキュリティリスクの確認**
   - セキュリティに関わるコード（認証、データ処理など）は特に慎重に検証し、セキュリティベストプラクティスを守る。
5. **依存関係に注意**
   - 生成されたコードが使用するライブラリやツールが自分の環境にインストールされているか確認する。

教育や学習での注意点

6. **生成コードの理解を促す**
   - 学生や初心者がコードをそのまま使うのではなく、コードの意図や仕組みを理解するよう促す。
7. **コーディングスタイルの統一**
   - チームや授業で使う場合、スタイルガイドに沿ってコードを整える。
8. **依存しすぎない**
   - ChatGPTに頼りすぎると、自己解決能力や学習意欲が低下する可能性があるので、適度な使用を心がける。
9. **アクティブな参加を促進**
   - 学生に対して生成されたコードをそのまま使わせるのではなく、改良や分析、デバッグ作業に取り組ませる。

倫理的・法的な注意点

10. **著作権に注意**
    - ChatGPTが生成するコードは一般的な知識をもとにしているが、特定のコードが著作権侵害に該当する可能性がある。
11. **プライバシー保護**
    - 機密情報や個人情報を含むデータを入力しない。
12. **責任の所在**
    - 生成コードによる問題が発生した場合の責任は利用者にある。最終的な判断は自分で行う。

プログラミングにおける利用例: processing　5章

processingでフクロウを描くプログラムを書いて．ただし，フクロウを描く部分は関数にして．

Processingでフクロウを描くためのプログラムを関数化して作成しました。以下のコードでは、`drawOwl()` という関数を使ってフクロウを描きます。

java　コードをコピーする

```
void setup() {
  size(400, 400);
  background(255);
  // フクロウを描画
  drawOwl(200, 200, 150);
}

// フクロウを描く関数
void drawOwl(float x, float y, float size) {
  // 体
  fill(150, 100, 50);
  ellipse(x, y, size, size * 1.4);

  // 頭
  fill(150, 100, 50);
  ellipse(x, y - size * 0.6, size * 0.7, size * 0.7);

  // 目
  fill(255);
  ellipse(x - size * 0.2, y - size * 0.7, size * 0.2, size * 0.2);
  ellipse(x + size * 0.2, y - size * 0.7, size * 0.2, size * 0.2);

  // 黒目
  fill(0);
  ellipse(x - size * 0.2, y - size * 0.7, size * 0.1, size * 0.1);
  ellipse(x + size * 0.2, y - size * 0.7, size * 0.1, size * 0.1);

  // くちばし
  fill(255, 165, 0);
  triangle(x, y - size * 0.6, x - size * 0.05, y - size * 0.5, x + size * 0.05, y -

  // 足
  fill(255, 165, 0);
  rect(x - size * 0.2, y + size * 0.7, size * 0.1, size * 0.2);
  rect(x + size * 0.1, y + size * 0.7, size * 0.1, size * 0.2);

  // 羽
  fill(120, 80, 40);
  ellipse(x - size * 0.4, y, size * 0.3, size * 0.7);
  ellipse(x + size * 0.4, y, size *
```

```
}

void draw() {
  // 空のdrawで自動更新を無効化
}
```

説明：

1. `drawOwl()` 関数は、指定された座標（`x`, `y`）と大きさ（`size`）を元にフクロウを描きます。
2. `setup()` 関数では、ウィンドウサイズを指定し、フクロウを一度だけ描画します。
3. `draw()` 関数は空にして、フクロウが描かれた後に余計な更新が行われないようにしています。

フクロウを描くための基本的な要素は、体、目、くちばし、足、そして羽です。

もう少しずんぐりとしたフクロウにして

ずんぐりとしたフクロウを描くように、体や羽、頭の比率を調整しました。以下のコードはその変更を反映しています。

java　コードをコピーする

```
void setup() {
  size(400, 400);
  background(255);
  // ずんぐりとしたフクロウを描画
  drawOwl(200, 200, 150);
}

// ずんぐりとしたフクロウを描く関数
void drawOwl(float x, float y, float size) {
  // 体（幅を広くし、全体的に丸みを強調）
  fill(150, 100, 50);
  ellipse(x, y, size * 1.1, size * 1.2);

  // 頭（体に対して少し大きめに設定）
  fill(150, 100, 50);
```

主な変更点:

- 体の幅を広げ、縦横の比率を少し丸くしました。
- 頭を大きくし、全体的にバランスを取るために、目や足も少しずつ大きめにしました。
- 羽は全体がずんぐりと見えるように小さめに設定しました。

この調整によって、フクロウがよりずんぐりとした印象になります。

耳をつけて

耳をつけて、フクロウらしさを強調したバージョンのコードを作成しました。耳はフクロウの頭の上に三角形で表現しています。

java コードをコピーする

```java
void setup() {
  size(400, 400);
  background(255);
  // ずんぐりしたフクロウを描画
  drawOwl(200, 200, 150);
}

// ずんぐりしたフクロウを描く関数（耳つき）
void drawOwl(float x, float y, float size) {
  // 耳
  fill(150, 100, 50);
  triangle(x - size * 0.25, y - size * 0.75, x - size * 0.35, y - size * 0.9, x -
  triangle(x + size * 0.25, y - size * 0.75, x + size * 0.35, y - size * 0.9, x +
```

繰り返し

このフクロウを横に１０体，縦に５体ずつ合計５０体整列して書いて．

50体のフクロウを横に10体、縦に5体並べるようにコードを修正しました。`for`ループを使って、フクロウを規則的に整列させます。

java　コードをコピーする

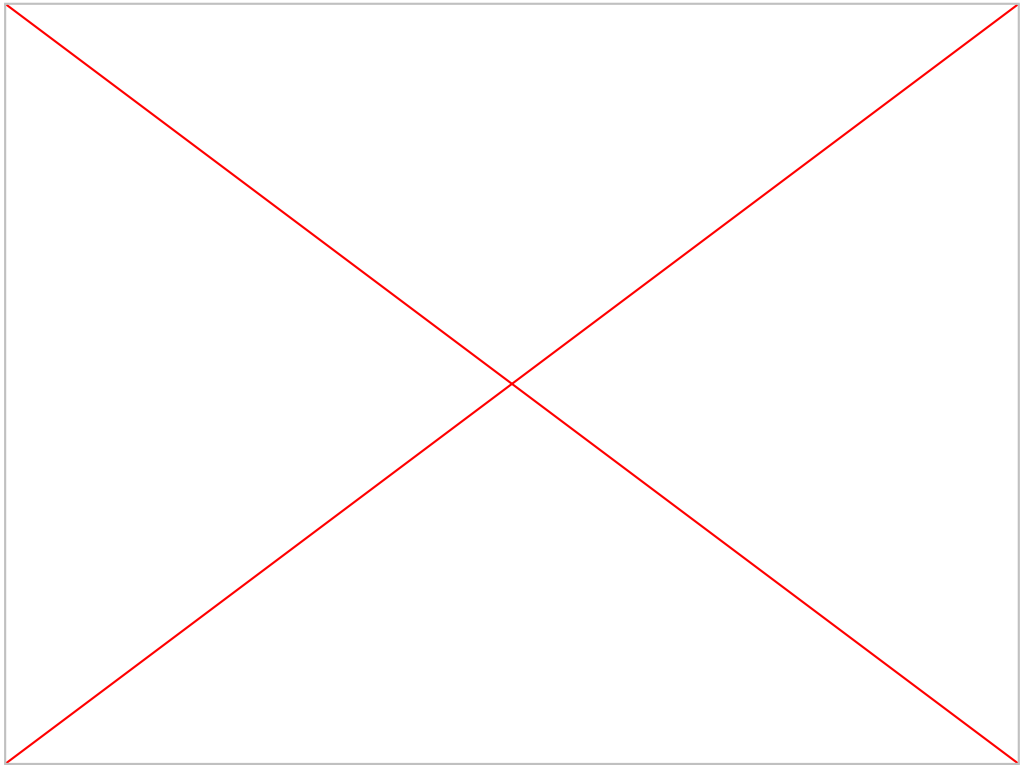
```
void setup() {
  size(800, 600);  // ウィンドウサイズを適切に拡大
  background(255);

  // フクロウの配置
  float owlSize = 60;  // フクロウの大きさ
  for (int i = 0; i < 10; i++) {      // 横に10体
    for (int j = 0; j < 5; j++) {     // 縦に5体
      float x = (i + 1) * (owlSize * 1.5);  // 横方向の位置調整
      float y = (j + 1) * (owlSize * 2);    // 縦方向の位置調整
      drawOwl(x, y, owlSize);               // フクロウを描画
    }
  }
}

// ずんぐりしたフクロウを描く関数（耳つき）
void drawOwl(float x, float y, float size) {
  // 耳
  fill(150, 100, 50);
  triangle(x - size * 0.25, y - size * 0.75, x - size * 0.35, y - size * 0.9, x -
  triangle(x + size * 0.25, y - size * 0.75, x + size * 0.35, y - size * 0.9, x +

  // 体（幅を広くし、全体的に丸みを強調）
  fill(150, 100, 50);
  ellipse(x, y, size * 1.1, size * 1.2);

  // 頭（体に対して少し大きめに設定）
  fill(150, 100, 50);
```

説明：

- `for`ループを使って、フクロウを横に10体、縦に5体並べています。
- `owlSize`でフクロウの大きさを指定し、それに基づいて各フクロウの配置を計算しています。
- `x`と`y`の位置は、フクロウ同士が適切な間隔を持つように調整しています。

これで、整列した50体のフクロウが描画されます。

組み合わせてみよう(関数: A君）

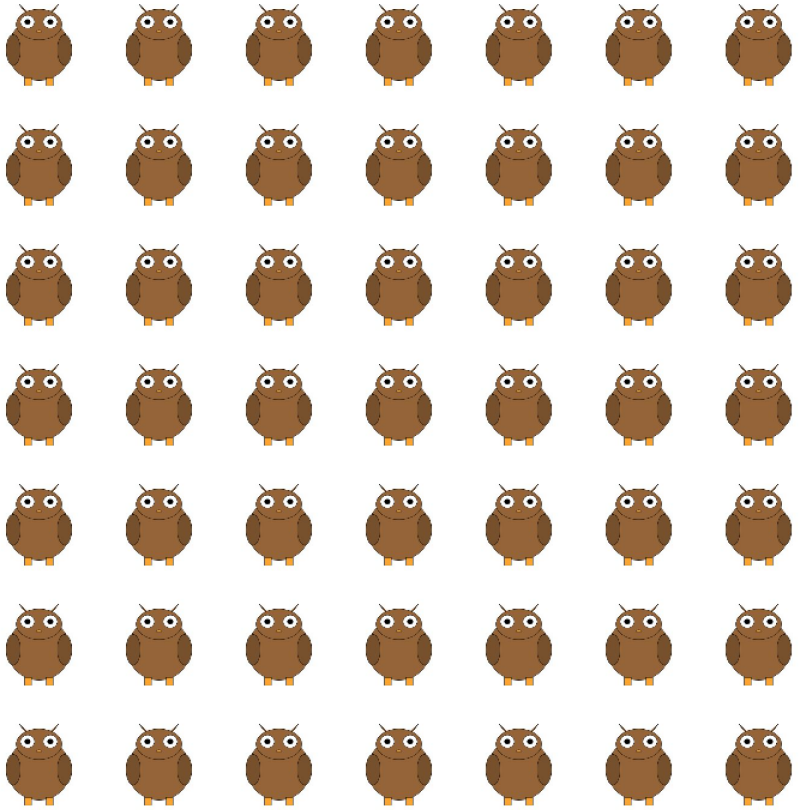


アニメーションからシューティングゲーム（B君）

プロセシングでフクロウを縦と横に一定の間隔をあけて整列させてほしい　できるだけ多く

Processingでフクロウの画像を縦横に一定の間隔をあけて整列させるために、フクロウの画像を用意し、それをグリッド状に配置するコードを作成します。たとえば、`loadImage()`を使ってフクロウの画像を読み込み、ループを使って縦横に繰り返し配置します。

以下はそのサンプルコードです。

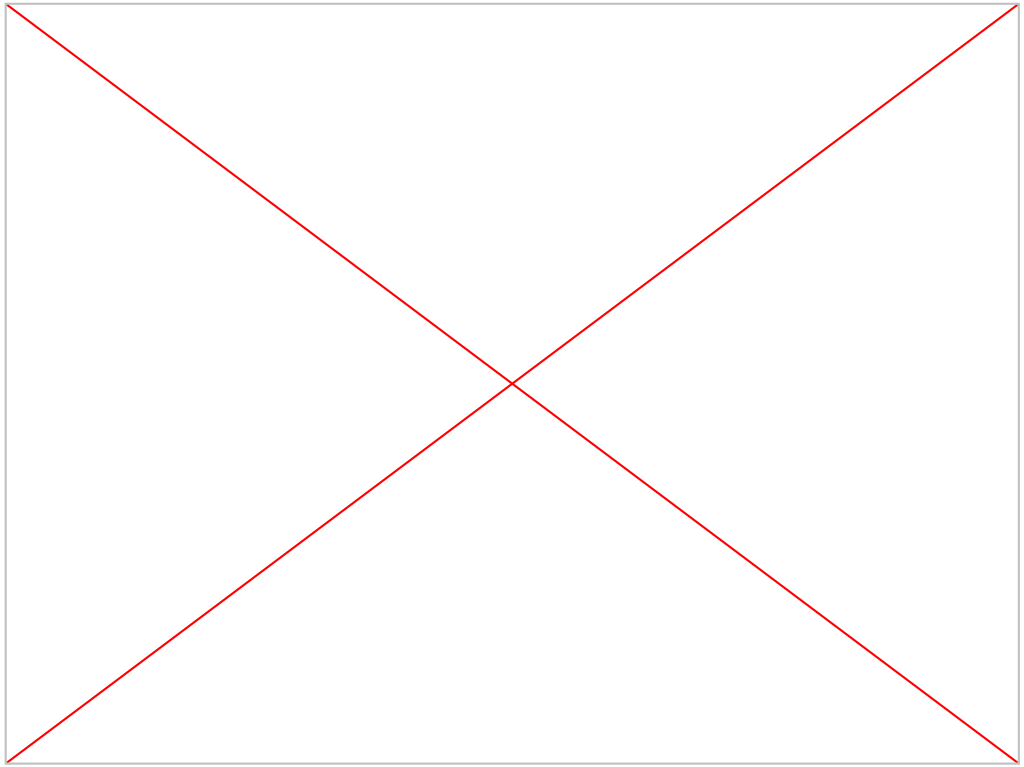


アニメーションからシューティングゲーム（B君）

この生成したフクロウを一個一個アニメーションで出していくプログラムをかいて

フクロウを一つずつアニメーションで表示するには、フクロウの画像を描画する座標を順番に増やしていくロジックを追加すればよいです。表示するフクロウの数をカウントして、その数に応じて順次描画するようにします。

以下はフクロウを1つずつアニメーションで表示するサンプルコードです。

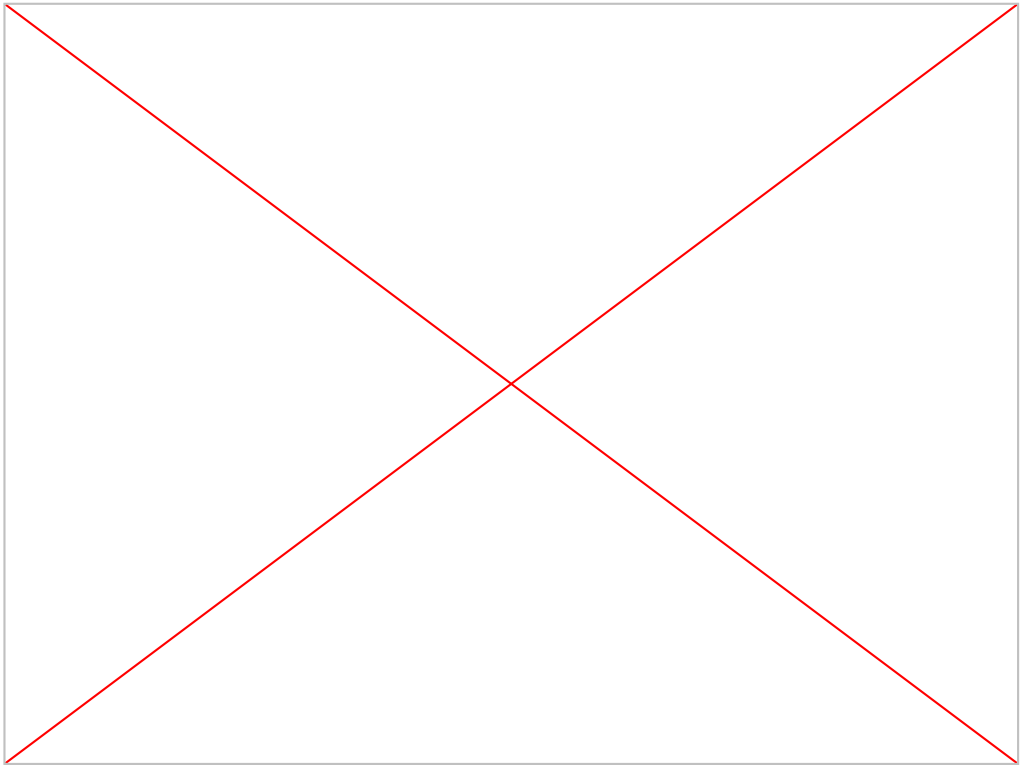


アニメーションからシューティングゲーム（B君）

このフクロウを打ち落とすシューティングゲームをこのコードを利用して作ってください

このフクロウを打ち落とすシューティングゲームを作るために、以下の要素を追加して実装します。

1. **フクロウを動くターゲットにする**：フクロウが動くようにし、打ち落とせるターゲットとして扱います。
2. **プレイヤーの操作**：マウスクリックでフクロウを打ち落とす機能を追加。
3. **スコア**：プレイヤーがフクロウを打ち落とすとスコアが増加する仕組みを追加。
4. **ゲームの終了条件**：一定時間内に全てのフクロウを打ち落とせなければゲームオーバーとする。



シューティングゲームを作って(A君）

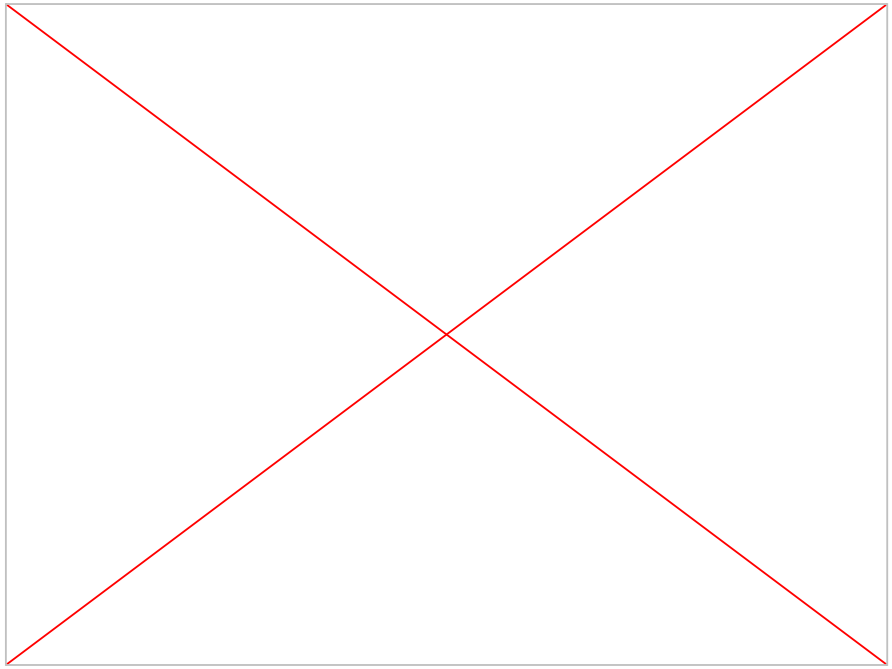
processinngで動作するシューティングゲームを作成してほしい

Processingで動作するシンプルなシューティングゲームの基本コードを提供します。このゲームでは、プレイヤーが左右に移動し、スペースキーで弾を発射して敵を倒すというシンプルな仕組みを持っています。

シューティングゲームの基本構成

- プレイヤー：画面下部に位置し、左右の移動が可能。
- 弾：スペースキーを押すと発射され、敵に当たると消滅。
- 敵：画面上部からランダムな位置で出現し、下に向かって移動。

以下は、Processingで実装したコードです。



機能説明：

- **プレイヤー移動**: `a` キーで左、`d` キーで右に移動します。
- **弾の発射**: スペースキーを押すと弾が発射されます。
- **敵の生成と移動**: 画面上部から敵が出現し、下に移動します。敵は画面外に出たら再度上から出現します。
- **当たり判定**: 弾が敵に当たると、弾と敵が消え、敵は再び上から出現します。

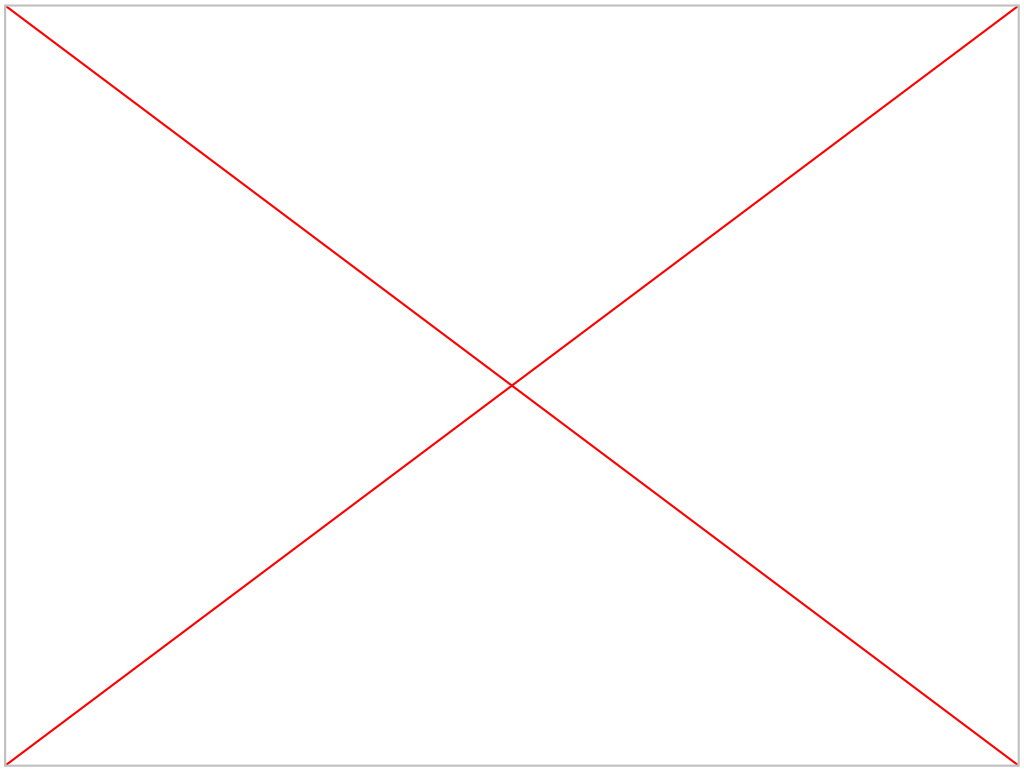
このコードを実行することで、シンプルなシューティングゲームが楽しめます。より複雑なゲームにしたい場合は、敵のパターンを増やしたり、スコア機能を追加するなどしてカスタマイズできます。

シューティングゲームを作って(A君）

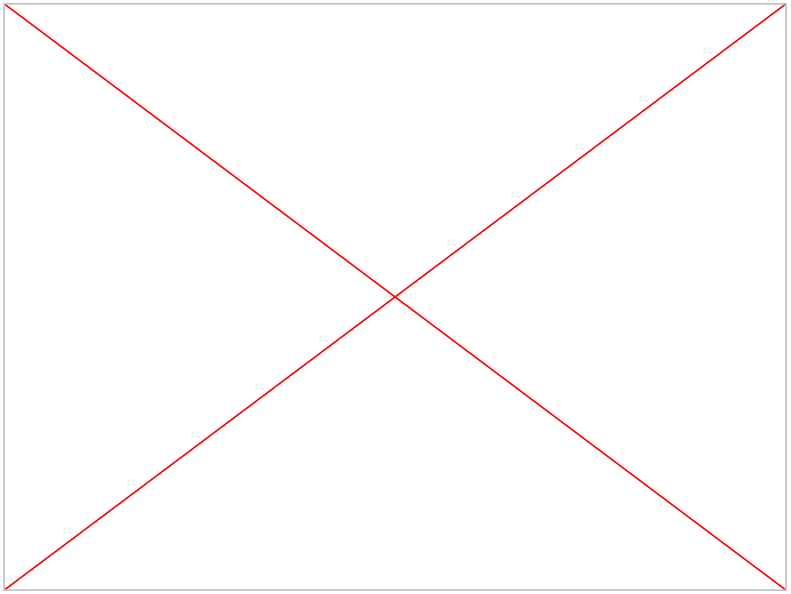
敵が自陣に着地したらゲームオーバー敵を5体倒したらゲームクリアとしてスコアボードを作成してほしい

ゲームに「ゲームオーバー」と「ゲームクリア」の概念を追加し、スコアボードを作成するために、次の機能を追加します。

- **敵が自陣（プレイヤーの領域）に着地したらゲームオーバー。**
- **敵を5体倒したらゲームクリア。**
- **スコアボード**：倒した敵の数を表示。
- **ゲームオーバー・ゲームクリア時の画面表示。**



元ネタは？（C君）



元となるコードがあるurlを教えて

私が提供したシューティングゲームのコードは、特定のURLからコピーしたものではなく、私の知識に基づいて新たに作成したサンプルです。Processingにおける基本的なシューティングゲームの実装は、過去に多くのリソースやチュートリアルで紹介されていますが、元々のコードがある特定のURLというのは存在しません。

ただし、Processingの公式サイトや、一般的なシューティングゲームに関するチュートリアルが公開されている場所をいくつか紹介できます：

1. **Processing公式チュートリアル**
   Processingの公式サイトでは、ゲーム作成のための基本的なチュートリアルがいくつか紹介されています。基本的なグラフィックスやインタラクションの作成方法が学べます。
   URL: https://processing.org/
2. **The Coding Train (Daniel Shiffman)**
   Daniel Shiffman氏の「The Coding Train」チャンネルでは、Processingを使った様々なプロジェクトのチュートリアルが公開されています。シューティングゲームやアニメーション、インタラクティブな作品に関する動画も多く、参考になります。
   URL: https://thecodingtrain.com/
3. **OpenProcessing**

設計を入れる（オブジェクト指向：D君）

オブジェクトをプロセッシングで作って

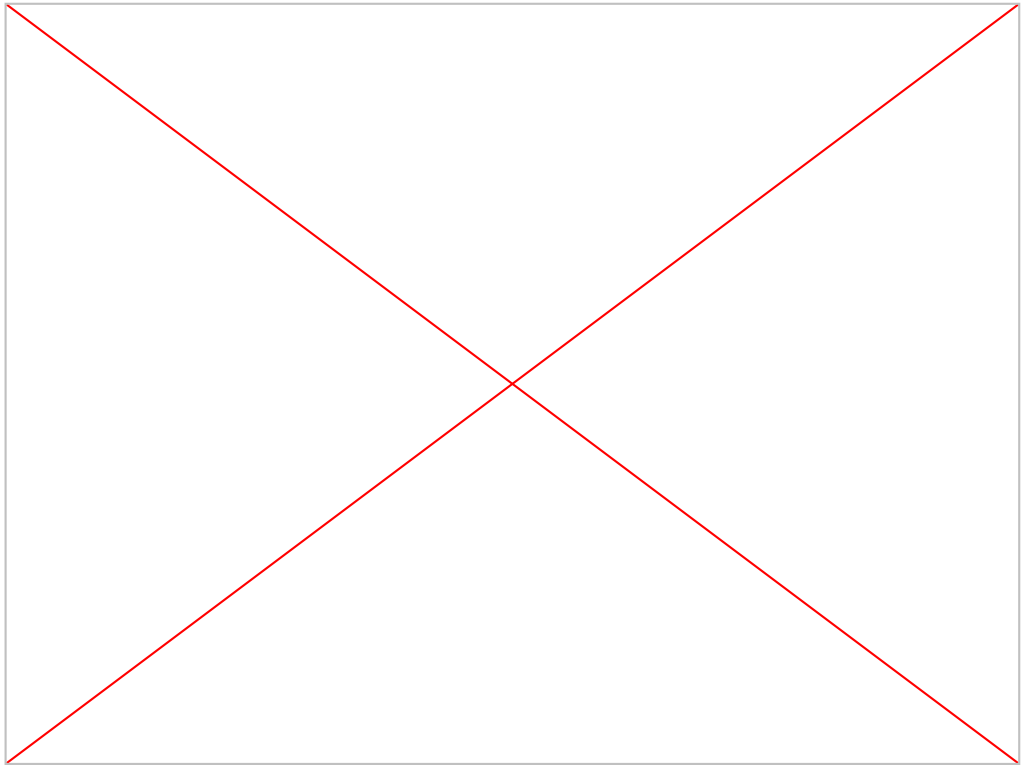
Processingでオブジェクト指向のコードを書くと、コードを整理しやすくなり、再利用も簡単になります。ここでは、Processingで円を表す「Circle」クラスを作り、それを使用して複数の円を描画する例を紹介します。

ステップ1：Circleクラスの作成

まず、`Circle`というクラスを定義します。このクラスでは、円の位置やサイズ、色などのプロパティを設定し、描画するためのメソッドも用意します。

java　コードをコピーする

```
class Circle {
  float x, y;    // 円の中心座標
  float radius;  // 円の半径
  color c;       // 円の色
```



いろいろな形のオブジェクトでゲームをプロセッシングで作成して

Processingを使って、シンプルなオブジェクト指向のゲームを作成してみましょう。ここでは、いろいろな形状（円や四角形、三角形など）のオブジェクトを用いた簡単な回避ゲームを例にします。

ゲームの概要

- プレイヤー（青い円）をマウスで操作して、ランダムに動く敵（異なる形のオブジェクト）に当たらないように避け続けます。
- 敵に当たるとゲームオーバーになります。
- 時間が経つごとに敵の数が増えることで、難易度も徐々に上がります。

コード例

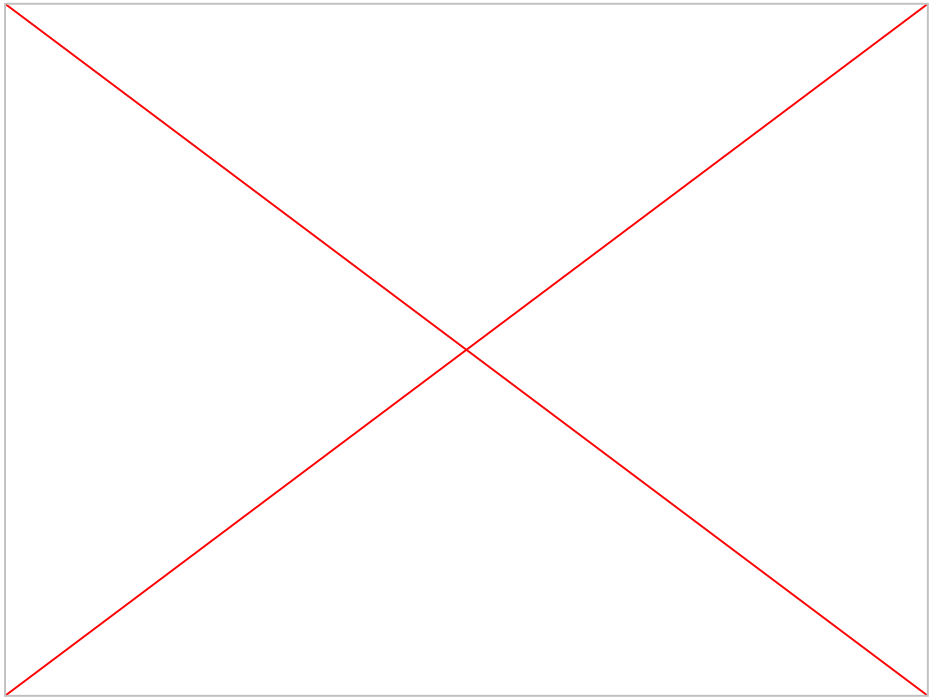
ステップ1：`Player`と`Enemy`クラスの作成

設計を入れる（オブジェクト指向：E君）

上記のコードを難しいのにして

より難易度が高いゲームを作成するため、以下のような要素を追加します：

1. **敵の動きをパターン化**: ランダムではなく、プレイヤーを追尾する敵や、曲線移動する敵など、異なる動きのパターンを持たせます。
2. **プレイヤーに制限**: プレイヤーが移動できる範囲を制限し、徐々にエリアが狭まるなどのプレッシャーを追加。
3. **追加の障害物**: 敵以外にプレイヤーに当たるとペナルティを受ける障害物や、プレイヤーをスローにするトラップを追加。
4. **スコアリングシステム**: 生き延びた時間に応じてスコアを増やし、スコアに応じて敵の出現頻度を上げる。



設計を入れる（バスケットボール：B君）

ゲーム設計

- プレイヤーはバスケットボールをシュートして、ゴールに入れることを目指します。
- シュートの方向や強さを調整して、正確にゴールを狙います。
- 一定の時間内で何回ゴールを決められるか、または特定の得点に達するまでプレイします。

ゲームの主要要素

バスケットボール

ボールの位置

ボールの移動（物理法則：重力や初速など）

ボールの大きさ

シュートの速度や角度を計算

ゴール（バスケットリング）

ゴールの位置（画面上部の固定位置）

ゴールのサイズ

ゴール判定（ボールが通過したときに得点をカウント）

制限時間（例 30秒、60秒など）

プレイヤー操作

マウスクリックまたはキーボード操 作でシュートの方 向と強さを調整

シュートボタンを押した際にボールを発射

シュートの角度や強さの調整方法

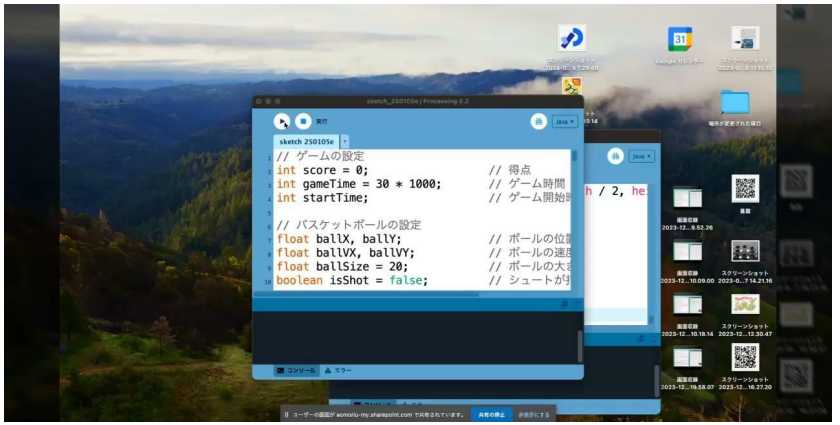
得点管理

ゴールが決まるたびに得点を加算

画面上に現在の得点を表示

ゲーム終了時に総得点を表示

タイマー



設計を入れる（バスケットボール：B君）リファクタリング

このコードのボールの部分をオブジェクトにしてほしい

ボールに関するコードをオブジェクト指向プログラミングの形式にリファクタリングしてみましょう。`Ball` クラスを作成し、ボールの位置や速度、動作（移動、リセット、描画）を管理します。

以下は、`Ball` クラスを導入したリファクタリング版のコードです。

リファクタリングポイント

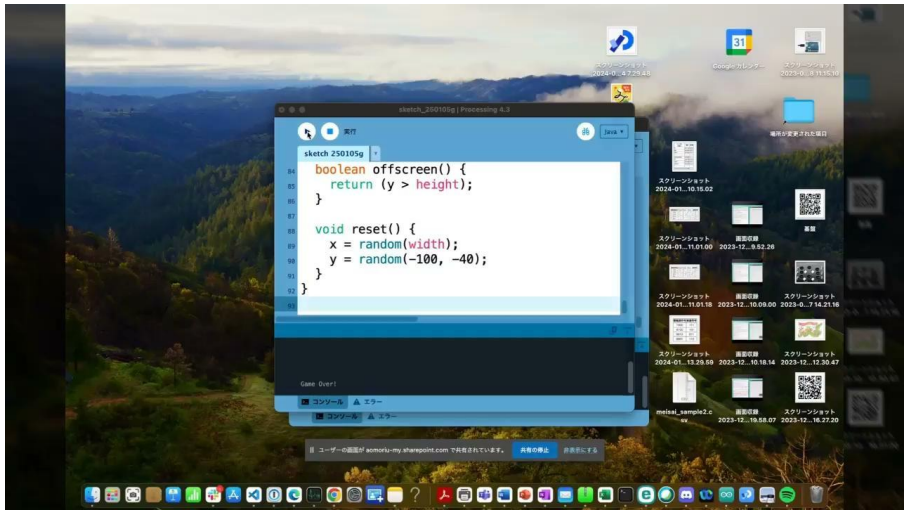
1. **Ball クラスの作成**:
   - `x`, `y`, `vx`, `vy`, `size`, `isShot` など、ボールに関連する変数を `Ball` クラスにまとめました。
   - `update()` メソッドでボールの移動と重力の影響を管理。
   - `reset()` メソッドでボールの位置と速度を初期化。
   - `shoot()` メソッドでボールをシュートするロジックを追加。
2. **ゴール判定の追加**:
   - `isScored()` メソッドで、ボールがゴールに入ったかどうかの判定を行うようにしました。
3. **ball オブジェクトの使用**:
   - `ball` オブジェクトを使用してボールの描画や移動、判定などを行います。

これで、ボールの動作が `Ball` クラスに分離され、コードが整理されました。

コードをコピーする

```
int score = 0;                  // 得点
int gameTime = 30 * 1000;       // ゲーム時間（30秒）
int startTime;                  // ゲーム開始時刻

// バスケットボールクラス
class Ball {
  float x, y;                   // ボールの位置
  float vx, vy;                 // ボールの速度
  float size = 20;              // ボールの大きさ
  boolean isShot = false;       // シュートが打たれたかどうか

  Ball() {
    reset();
  }

  void update() {
    if (isShot) {
      x += vx;
      y += vy;
      vy += 0.5; // 重力

      // ボールが地面または画面外に出た場合のリセット
      if (y > height || x > width || x < 0) {
        reset();
      }
    }
  }

  void reset() {
```

シンプルな設計から改善（D君）



プロセシングでオブジェクト与えてゲームを作れ

Processingを使って簡単なゲームを作るための基本的なアイデアを紹介します。ここでは、オブジェクト指向プログラミングを用いて、プレイヤーが動くことができ、敵を避けるゲームのサンプルを作ります。

1. ゲームの基本構造

以下のコードは、Processingでシンプルなゲームを作成するための例です。プレイヤーは矢印キーで移動し、敵を避ける必要があります。

```
  void reset() {
    x = random(width);
    y = random(-100, -40);
  }
}
```

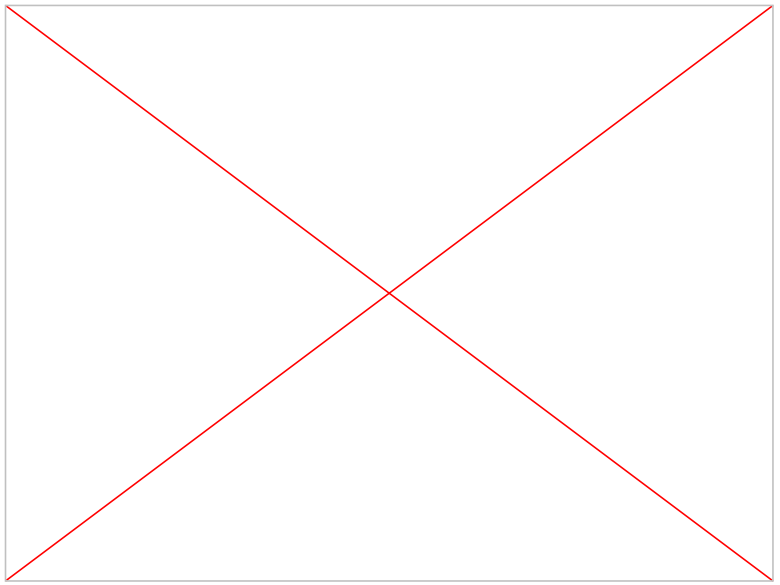
これに敵用のオフジェクトを作れ

Processingでのゲームに敵用のオブジェクトを追加するには、クラスを使用して敵の挙動や特性をカスタマイズできます。以下は、敵の動きをさらに多様化し、オブジェクト指向プログラミングの良さを活かした改善例です。

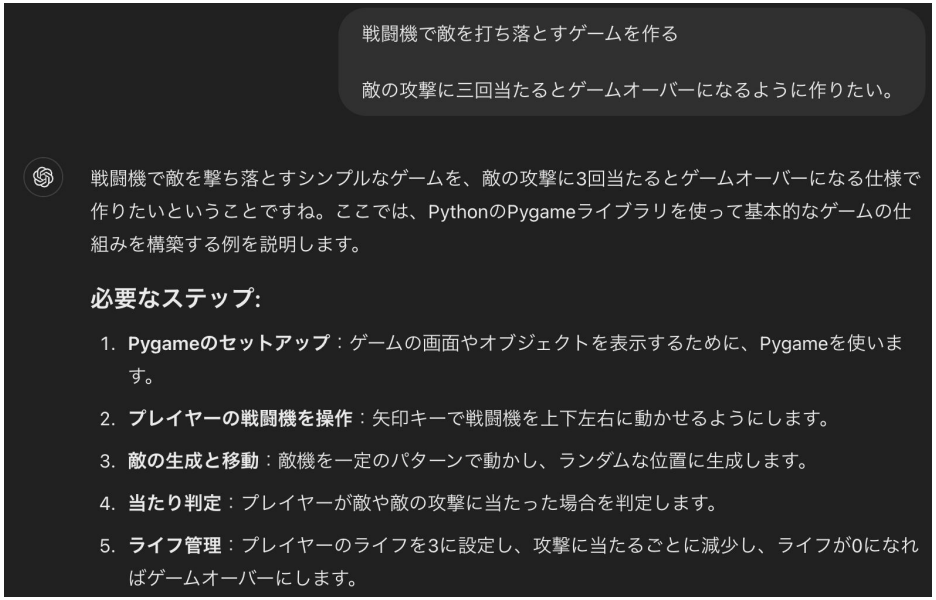
敵クラスの拡張

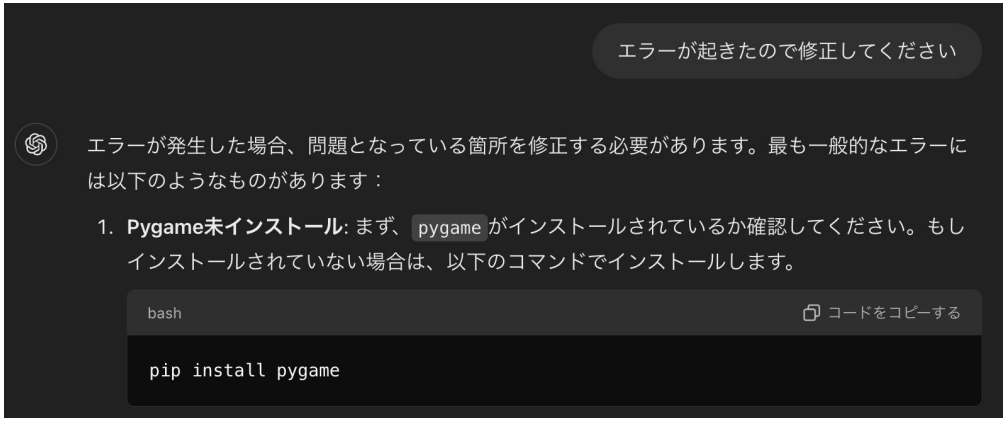
まず、敵クラスにランダムな速度を追加して、各敵が異なる速さで落下するようにします。また、敵が画面上で横に移動する機能も追加して、ゲームプレイをよりダイナミックにします。

以下は、改良された`Enemy`クラスを含む全体のコードです。



戦闘機ゲーム（C君）


戦闘機で敵を打ち落とすゲームを作る
敵の攻撃に三回当たるとゲームオーバーになるように作りたい。
戦闘機で敵を撃ち落とすシンプルなゲームを、敵の攻撃に3回当たるとゲームオーバーになる仕様で作りたいということですね。ここでは、PythonのPygameライブラリを使って基本的なゲームの仕組みを構築する例を説明します。
必要なステップ:
1. Pygameのセットアップ：ゲームの画面やオブジェクトを表示するために、Pygameを使います。
2. プレイヤーの戦闘機を操作：矢印キーで戦闘機を上下左右に動かせるようにします。
3. 敵の生成と移動：敵機を一定のパターンで動かし、ランダムな位置に生成します。
4. 当たり判定：プレイヤーが敵や敵の攻撃に当たった場合を判定します。
5. ライフ管理：プレイヤーのライフを3に設定し、攻撃に当たるごとに減少し、ライフが0になればゲームオーバーにします。


エラーが起きたので修正してください
エラーが発生した場合、問題となっている箇所を修正する必要があります。最も一般的なエラーには以下のようなものがあります：
1. Pygame未インストール: まず、pygameがインストールされているか確認してください。もしインストールされていない場合は、以下のコマンドでインストールします。
bash
コードをコピーする
pip install pygame

戦闘機ゲーム（C君）エラー

processingで作成してください

Processingで同様の戦闘機ゲームを作るには、`setup()`と`draw()`のメソッドを使って、画面描画やプレイヤーの移動、敵の生成と当たり判定を行います。

以下に、Processingで基本的な戦闘機ゲームを作成する例を示します。敵の攻撃に3回当たるとゲームオーバーになる仕様になっています。

Processingコード例

java　コードをコピーする

```
int screenWidth = 800;
int screenHeight = 600;

int playerX, playerY;  // プレイヤーの位置
int playerSize = 50;   // プレイヤーのサイズ
int playerLife = 3;    // プレイヤーのライフ

int[][] enemies;       // 敵の配列
int numEnemies = 8;    // 敵の数
int enemySize = 50;    // 敵のサイズ

void setup() {
  size(screenWidth, screenHeight);
  playerX = width / 2;
  playerY = height - 80;
```

エラーが出ました修正してください。

エラーが出た場合、原因を特定して修正する必要があります。以下に、考えられる一般的なエラーとその修正を示します。特にProcessingの範囲外参照やデータ型の間違いなどが原因でエラーが発生することが多いので、それらに注意して見直します。

次のようにいくつかの改良を加えた修正版コードです：

1. `checkCollision()`**の範囲チェック**を改善。
2. **変数の初期化**と**配列の定義**を明示的に行う。
3. **プレイヤーの動き**をリファクタリングして、画面外に出ないように調整。

修正版のコード

size(screenWidth, screenHeight);がおかしいので修正してください

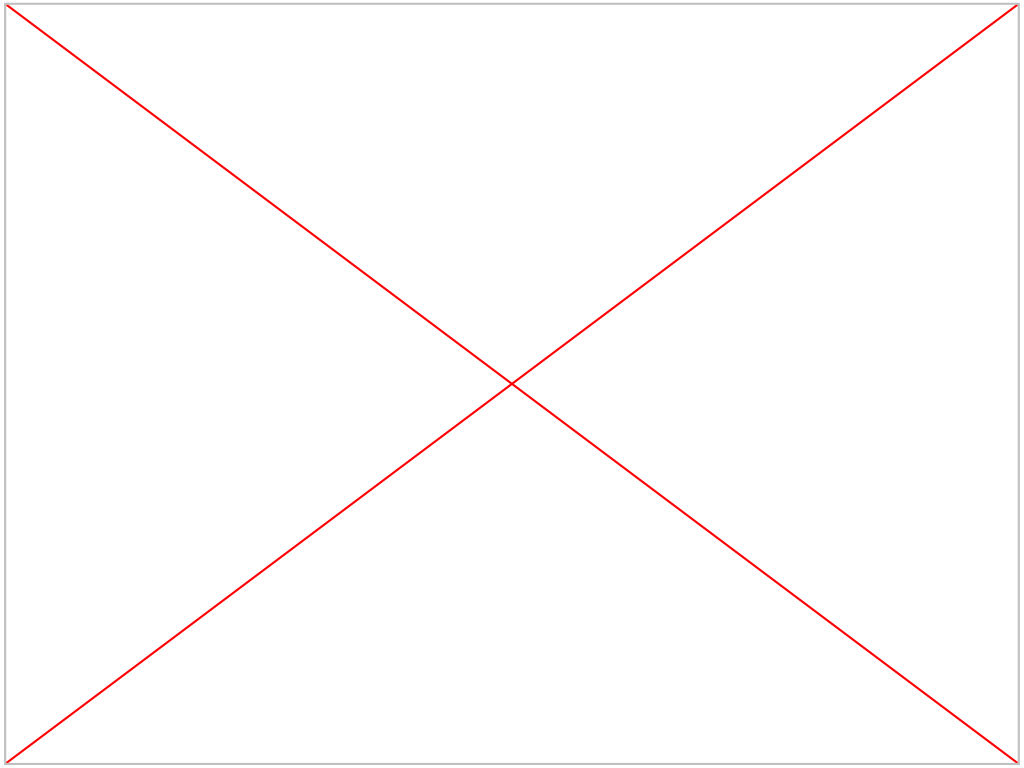
`size()`の呼び出しが、Processingの新しいバージョン（3以降）では`settings()`メソッド内に記述される必要があります。以下のように修正してみてください。

修正版コード

java　コードをコピーする

```
int screenWidth = 800;
int screenHeight = 600;

int playerX, playerY;  // プレイヤーの位置
int playerSize = 50;   // プレイヤーのサイズ
int playerLife = 3;    // プレイヤーのライフ

int[][] enemies;       // 敵の配列
int numEnemies = 8;    // 敵の数
int enemySize = 50;    // 敵のサイズ

void settings() {
  size(screenWidth, screenHeight);
}
```

戦闘機ゲーム（C君）



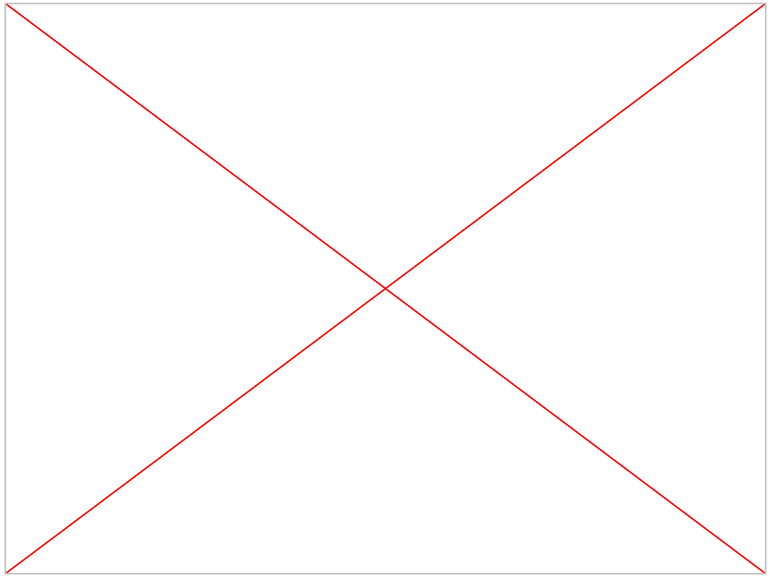
物理シミュレーションfisicaを利用して（A君）

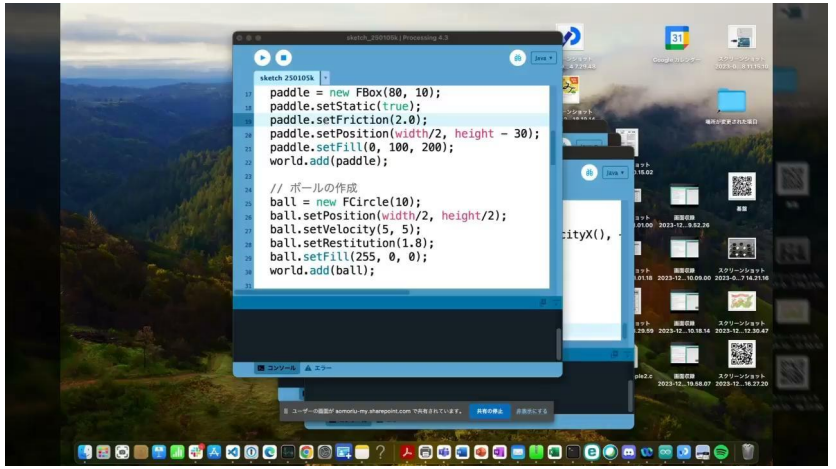
Fisicaとprocessingをつかいブロック崩しを作成してほしい

「Fisica」を使用して「Processing」でブロック崩しゲームを作成するための基本コードを紹介します。「Fisica」は物理エンジン「Box2D」のラッパーで、「Processing」上で2D物理シミュレーションを手軽に扱えます。このコードではボール、パドル、ブロックに物理的な動きを与え、ボールがブロックに当たると破壊されるシンプルなブロック崩しゲームを作ります。

準備

1. Fisica ライブラリが必要です。Fisica をダウンロードして「Processing」のライブラリに追加してください。
2. 「Fisica」ライブラリは、Fisica ライブラリのダウンロードから入手できます。





fisicaを使ったアニメーション（F君）

processingのfisicaを使って動く人のアニメーションを作ってほしい

`Processing`で`Fisica`ライブラリを使って動く人のアニメーションを作成するには、`Fisica`を使って物理的な動きを再現し、キャラクターを関節ごとに連結して体の各部分が動くようにします。以下は、簡単な人の形を模したオブジェクトを`Fisica`で動かすためのサンプルコードです。

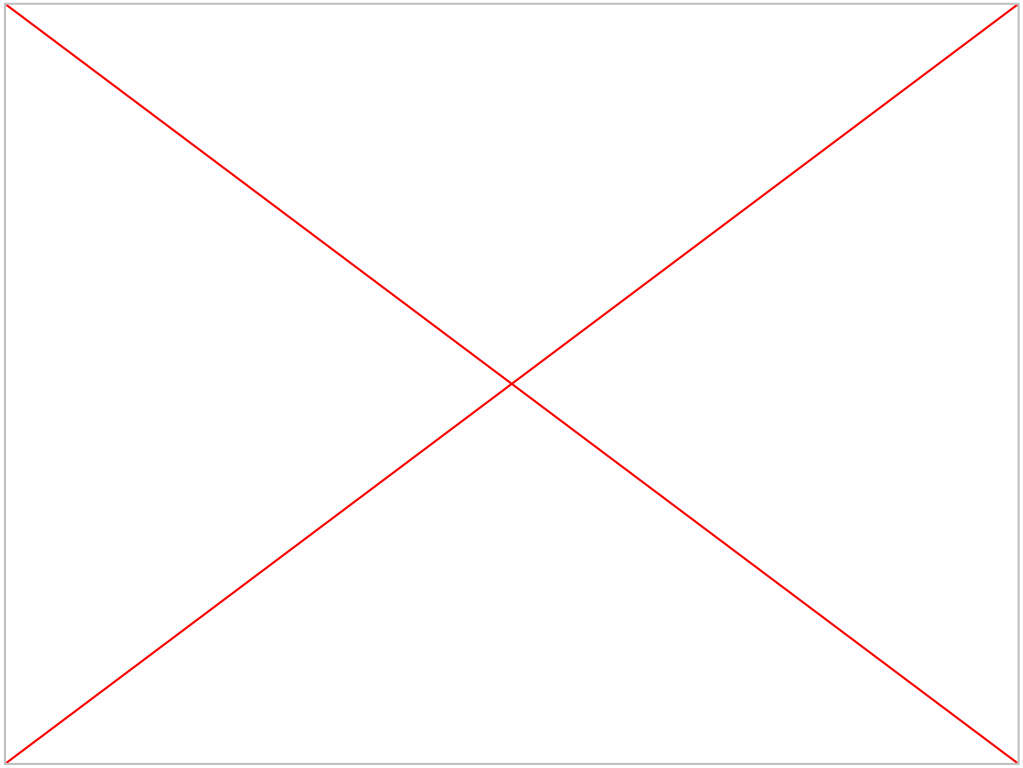
前提条件

まず、Fisicaライブラリをインストールし、`Fisica`を`Processing`に追加してください。`Fisica`ライブラリがまだインストールされていない場合は、`Processing`の「Sketch」>「Import Library」からインストールできます。

processingのfisicaを使って無限に動く人のアニメーションを作ってほしい

`Processing`の`Fisica`ライブラリを使って無限に動く人のアニメーションを作るには、物理エンジンを利用して体の各パーツを連結し、無限に動くループアニメーションを作成できます。例えば、体を円と線で表現し、腕や脚を物理的に動かしながら、歩行のような動きをさせることが可能です。

以下は、`Processing`と`Fisica`ライブラリを使って、歩行のような動きをする人を描く基本的なサンプルコードです。`Fisica`ライブラリをインストールしておく必要があります。



fisicaを使ったアニメーション（E君）試行錯誤

fisicalibaryとprocessingを使って雪が降ってるアニメーションを作って

Fisicaライブラリと Processingを使って雪が降るアニメーションを作るためには、雪の粒（フレーク）を物理的に落下させる設定が必要です。以下のコードでは、Fisicaライブラリで物理シミュレーションを行いながら、雪の粒が重力に従って降るようにしています。

準備

Fisicaライブラリがインストールされていない場合は、まずインストールしてください。

1. Fisica公式サイトからダウンロード
2. ダウンロードしたJARファイルをProcessingの `libraries` フォルダに追加

少しずつ雪が積もるようにして

雪が少しずつ積もるようにするためには、画面下に落ちた雪の粒を床として追加し、段々と高さを増やしていくことができます。この方法により、雪の粒が重なって積もっていくような効果が出せます。

以下は、落ちた雪の粒が積もるように改良したコードです。

下に積もる雪を粒から壁みたいにして

雪が積もる部分を粒から壁のようにするには、一定の高さに達した時に雪の粒を一体化させ、地面のように広がる「積雪の壁」を作成する方法が効果的です。この壁が徐々に高くなることで、積雪が増していくように見えます。

以下のコードでは、画面下に達した雪の粒がある程度たまると、幅いっぱいの「積雪ブロック」にまとめ、少しずつ高さを上げるようにしています。

The function getAge() does not exist.

`getAge()` がない場合、代わりにフレームカウントや他のタイマー変数を使って周期的な揺れを追加できます。以下のコードでは、`sin()` 関数を使い、`frameCount` に基づいて雪が揺れるように変更しました。

色を濃くしたり薄くしたりして立体感を出して

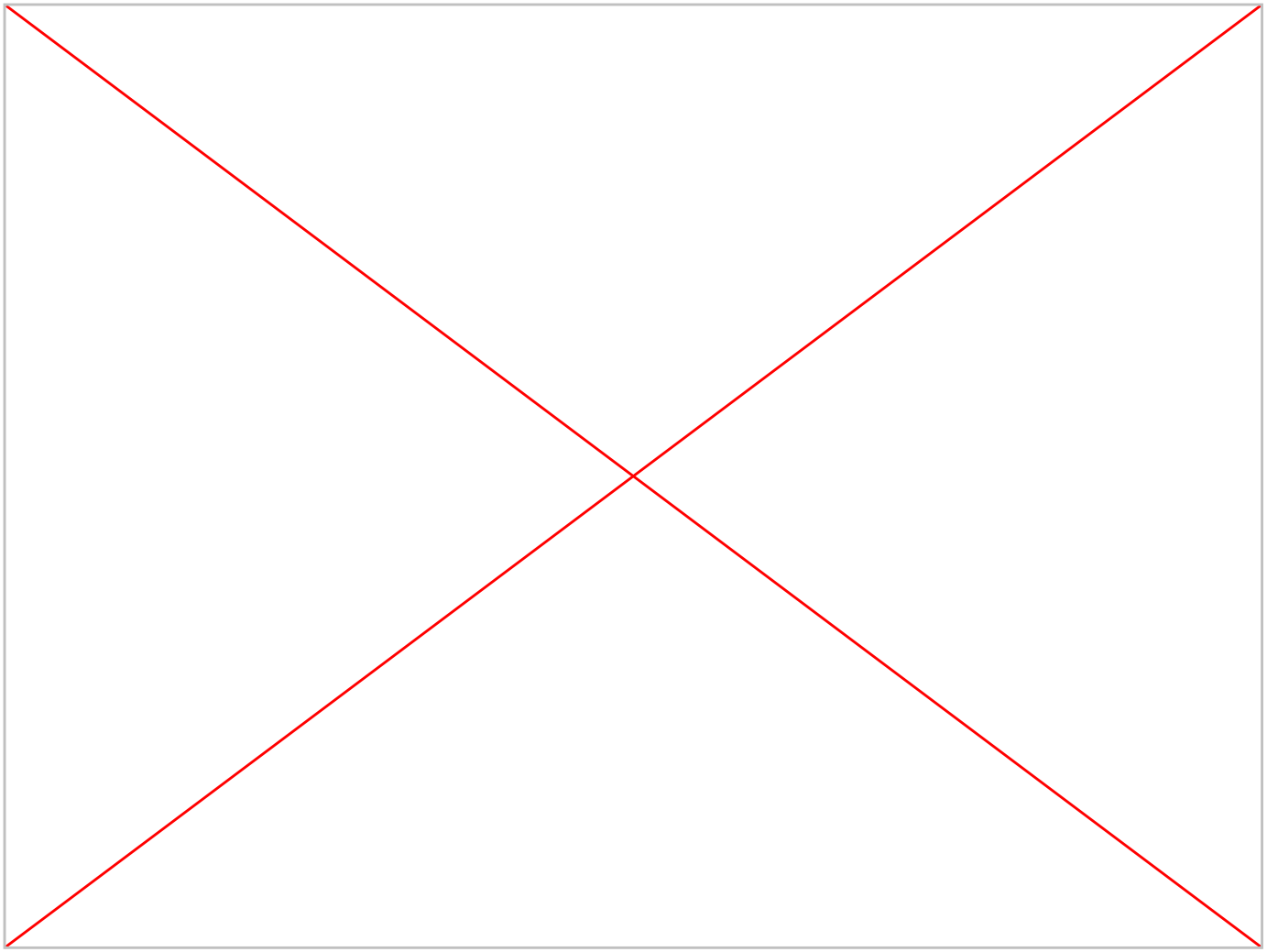
雪の粒の色を変化させて立体感を出すには、各雪の粒にランダムな色の濃淡を与えたり、フレームごとに少しずつ色を変化させたりする方法があります。ここでは、雪の粒に個別の色の濃さを設定し、`applyWobble()` メソッド内でその色を周期的に変化させるようにします。

以下は、雪の粒の色が少しずつ濃くなったり薄くなったりするようにしたコード例です。

粒の大きさをもっと大きくして

雪の粒の大きさを大きくするためには、`FSnowflake` クラスをインスタンス化する際に、ランダムな半径を大きい値に設定する必要があります。以下のコードでは、雪の粒のサイズを大きくしました

fisicaを使った
アニメーション
（E君）試行錯
誤



花火（B君）

processingで、きれいな花火をかいて、また、条件として花火が爆発する位置はバラバラで、色もバラバラにしてほしい。また、最後のほうでフィナーレとして花火が連続で何度も上がるような花火を描いてほしいです。

Processingで花火をランダムに打ち上げて、色や爆発位置がバラバラになるようにし、最後にフィナーレとして連続して花火が上がるようにするコードを書いてみますね。

このコードを使って、花火の爆発する位置をもっと高くする花火もいれて

花火が爆発する高さをランダムに変えるようにするには、`Firework`クラスの位置の初期値（`pos`）をランダムに設定します。`height`の上半分に限定して配置するようにし、より高く爆発する花火を加えます。

修正したコードはこちらです。

少しバラバラすぎるのでもっと真ん中くらいにあつめてほしい

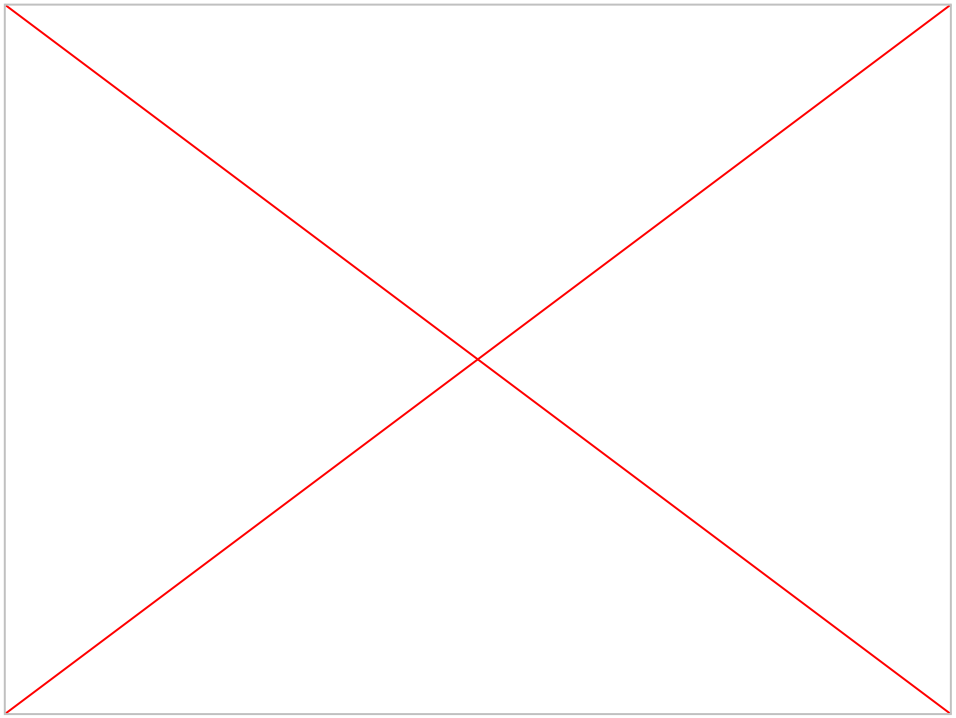
花火の爆発位置を中央付近に集めるために、位置の範囲を画面中央に寄せるように修正します。具体的には、`Firework`クラスの爆発位置（`pos`）を画面の中央付近に設定しつつ、少しランダム性を加えて自然なばらつきを出します。

修正したコードはこちらです。

花火の形がハートや星などもいれてほしい

花火の形をハートや星型にするには、`Firework`クラスで花火の形状を選択し、`Particle`クラスで形状に合わせて粒子を配置します。

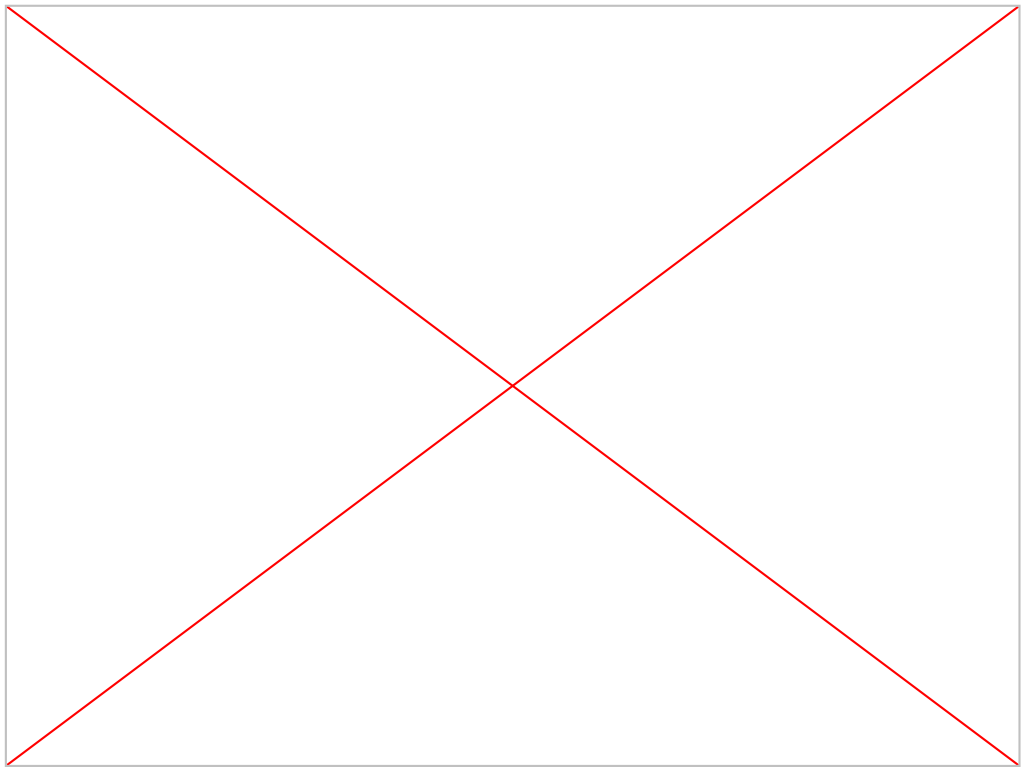
ハート型と星型を追加し、それ以外は通常の円形にランダムで設定するようにしたコードです。



fisicaでひよこ（G君）

ficicaとprocessingを使って、ヒヨコが動くアニメーションを作って

FicicaとProcessingを使って、簡単なヒヨコが動くアニメーションを作成してみましょう。このプログラムでは、物理エンジンを利用してヒヨコが地面を跳ねながら動くようにします。ヒヨコは重力に従い、左右に移動している様子を表現します。



対戦ゲーム＋fisica(E君）

processingでfisicaを使って簡単な対戦ゲームを作って

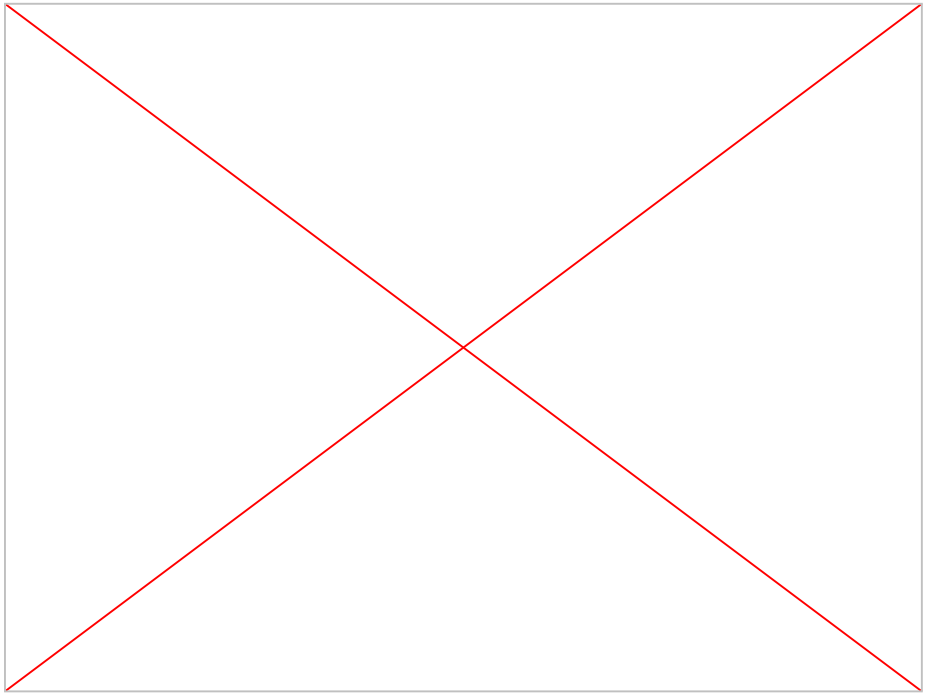
簡単な対戦ゲームを作るには、ProcessingとFisicaライブラリを使って、以下のような基本的な流れでゲームを作れます。たとえば、プレイヤー同士がボールを投げて相手に当てるシンプルなゲームを考えてみましょう。

球をもっと早くして

弾の速度を速くするには、`shootBullet()` 関数で設定している速度 (`vx` や `vy`) を増やせば、弾がより早く移動するようになります。例えば、速度を 2 倍にしてみましょう。

修正後のコードは以下の通りです：

```
// 弾を生成する関数
void shootBullet(float x, float y, float vx, float vy, int fillColor) {
  FCircle bullet = new FCircle(10);
  bullet.setPosition(x, y);
  bullet.setVelocity(vx * 2, vy * 2); // 速度を2倍にする
  bullet.setFillColor(fillColor);
  bullet.setNoStroke();
  bullets.add(bullet);
  world.add(bullet);
}
```



APIで間違える.

```
// キーボード入力での操作
void keyPressed() {
  // プレイヤー1の操作（W, Sで上下移動、Dで弾を右に発射）
  if (key == 'w') player1.setY(player1.getY() - 10);
  if (key == 's') player1.setY(player1.getY() + 10);
  if (key == 'd') shootBullet(player1.getX() + 30, player1.getY(), 5, 0, color(255,

  // プレイヤー2の操作（↑, ↓で上下移動、LEFTで弾を左に発射）
  if (keyCode == UP) player2.setY(player2.getY() - 10);
  if (keyCode == DOWN) player2.setY(player2.getY() + 10);
  if (keyCode == LEFT) shootBullet(player2.getX() - 30, player2.getY(), -5, 0, colo
}
```

なぜsetY()を提案したの.

`setY()` を提案したのは私の誤りです。`Fisica` ライブラリではオブジェクトの位置を設定する際に `setPosition(x, y)` を使用する必要があり、`setY()` のような関数は存在しません。

私が誤って `setY()` を提案した理由は、位置を調整するコードを簡潔にする意図があったかもしれませんが、Fisica ライブラリの仕様に沿っていない提案でした。正しくは以下のように記述する必要があります：

java　コードをコピーする

```
player1.setPosition(player1.getX(), player1.getY() - 10);
```

The function setY(float) does not exist.

`setY()` は Fisica ライブラリの `FBox` クラスには存在しないため、代わりに `setPosition(x, y)` を使用してプレイヤーの位置を更新できます。プレイヤーの現在の X 座標を保持し、Y 座標のみを変更する形に修正します。

fisica対戦型ホッケー（E君，F君）

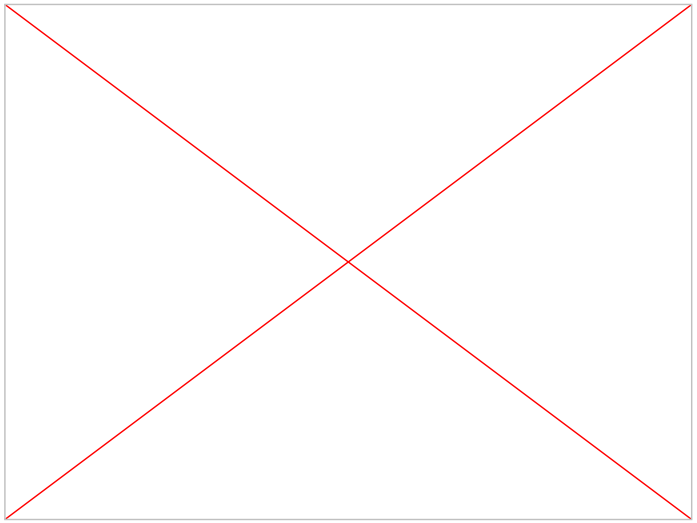
processingとFicicaを使って楽しめる対戦ゲームを作って

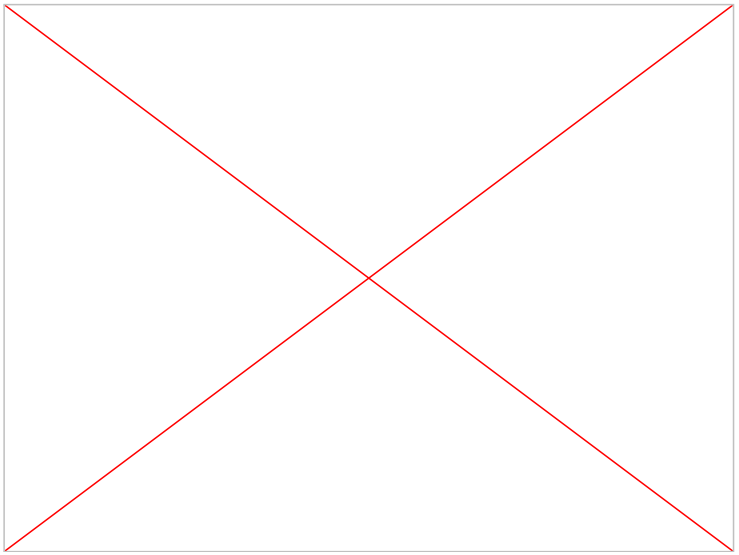
「Processing」と「Fisica」を使って、シンプルな対戦型ゲームを作ってみましょう。Fisicaは、Processingで物理演算を簡単に使えるようにするライブラリです。2人のプレイヤーが球を操作し、相手のゴールにボールを入れて得点するゲームを作ります。

Fisicaのインストールがまだの場合は、Processingのライブラリマネージャから「Fisica」を検索してインストールしてください。

ゲーム概要

- プレイヤー1とプレイヤー2はそれぞれ画面の左右に配置され、上下のキーで円形のプレイヤーを動かします。
- 中央に配置されたボールを相手のゴールに押し込み、得点を目指します。





fisica対戦型ホッケー（蛯澤）

processingのfisicaで対戦型ゲームをつくってほしい

Processingの `fisica` ライブラリを使用して、対戦型ゲームを作成しましょう。このライブラリは、物理演算を簡単に扱えるようにしてくれるため、ゲームオブジェクトの衝突判定や重力効果などを簡単に実装できます。

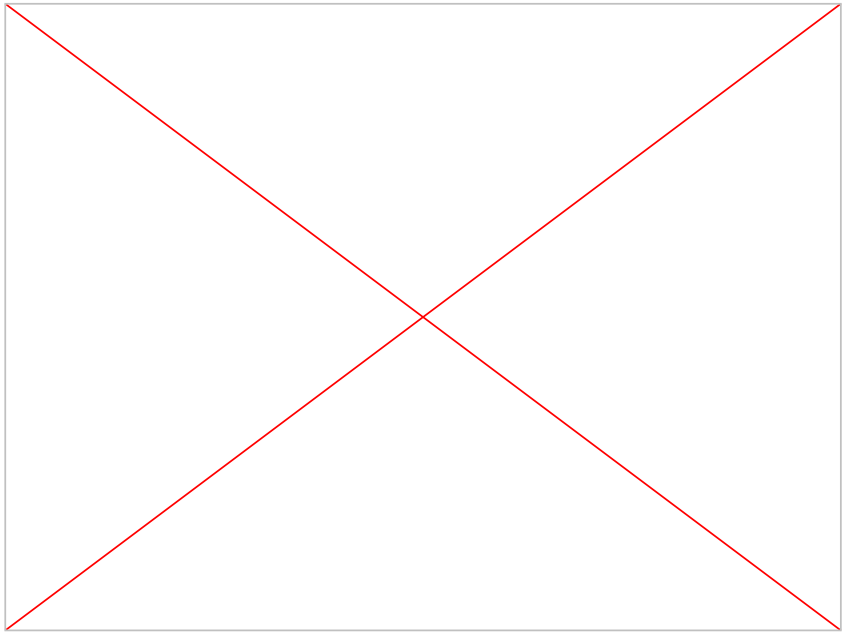
今回は、**2人のプレイヤーがそれぞれボールを操作して対戦するシンプルなゲーム**を作成します。このゲームでは、各プレイヤーが相手のゴールにボールを入れることでポイントを獲得します。キーボード操作を用いて各プレイヤーを動かし、物理エンジンで衝突や移動の処理を行います。

必要な準備

- `Processing` の `fisica` ライブラリをインストールしておいてください。
  - Processingのメニューから **Sketch > Import Library > Add Library...** で `fisica` を検索し、インストールしてください。

ゲームの仕様

1. **2人対戦**: 左のプレイヤーは `WASD` キー、右のプレイヤーは矢印キーで操作。
2. **ボール**: 物理演算に基づいて、プレイヤーがボールを押して相手のゴールに入れる。
3. **得点**: ボールがゴールに入ると得点が加算され、リスタート。
4. **制限時間**: 一定時間経過でゲーム終了、得点の高い方が勝者。



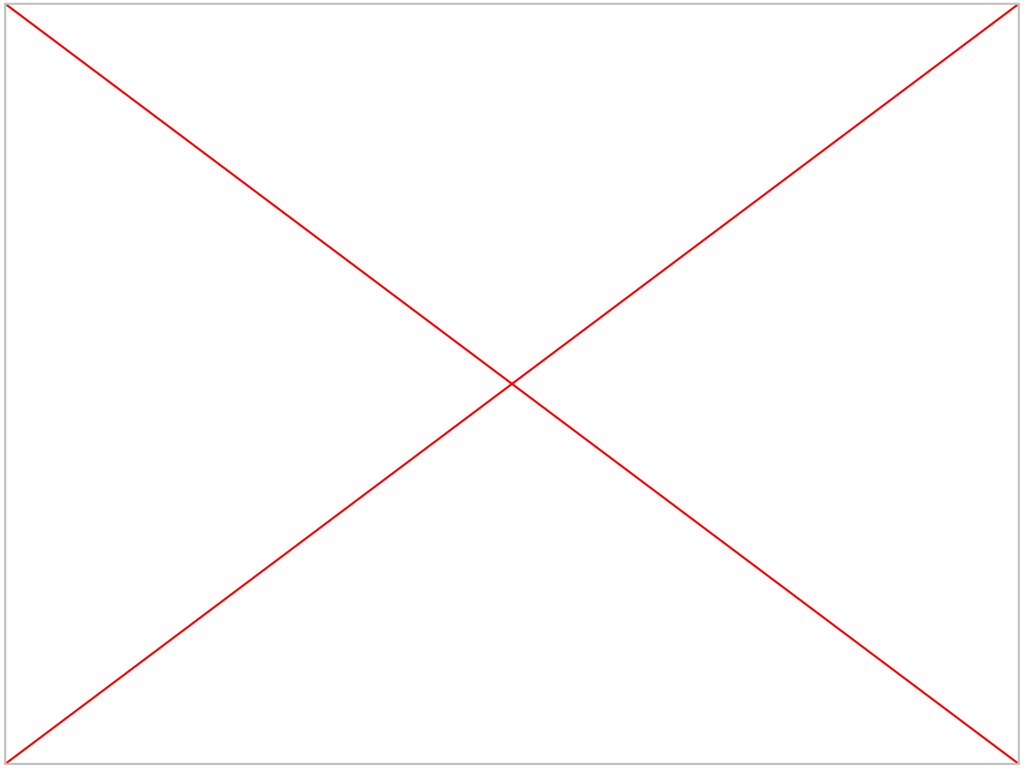
シューティング（A君）

processingを使って1対1のゲームを作ってほしい

「Processing」を使って1対1のゲームを作成したいということですね。ゲームの内容やジャンル（例えば、シューティング、バトル、パズルなど）をお聞きしてもよろしいでしょうか？それとも、基本的なテンプレートを用意して、その後に機能を追加する形で進めますか？

バトル

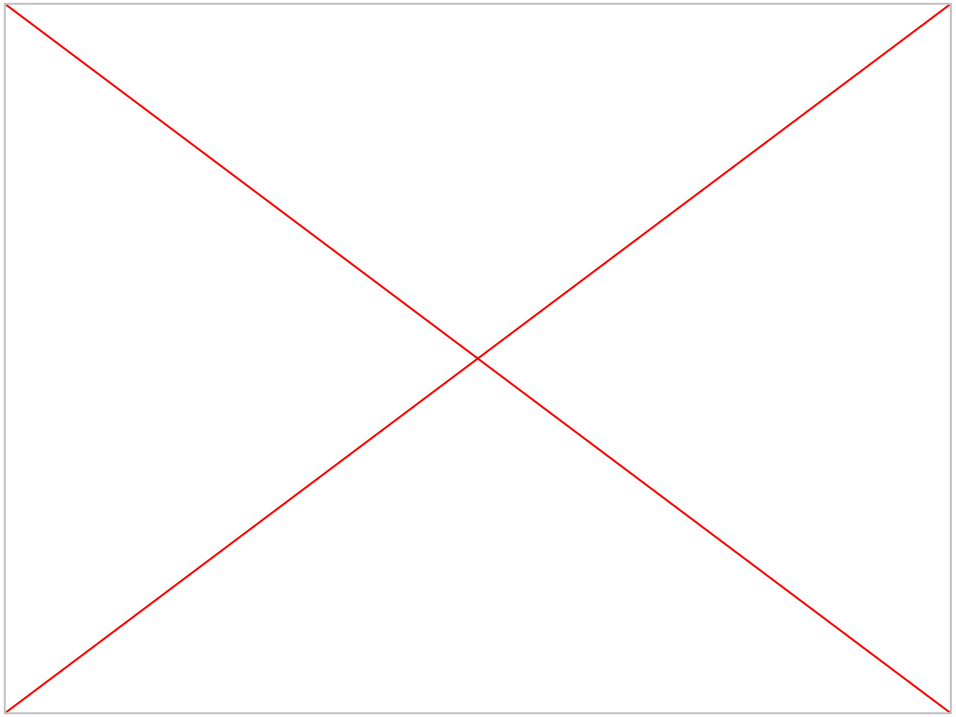
1対1のバトルゲームをProcessingで作成する場合、基本的な要素としては、プレイヤーキャラクターの操作、相手キャラクター、そしてバトル用のインターフェースが必要です。以下に基本的な構成のコードを示します。このコードは、プレイヤーと相手のキャラクターが動き回り、攻撃する簡単なバトルシステムです。



シューティング（A君）

玉を打ち出してシューティング１ｖ１でどちらも操作可能にしてほしい

シューティングバトルを1対1で実装し、両プレイヤーが操作できるようにするには、キャラクターごとに弾を打ち出し、相手を狙うシステムを作ります。以下はその実装例です。これで両方のプレイヤーが弾を発射し、互いに攻撃できるシューティングバトルゲームが作成できます。



シューティング（A君）

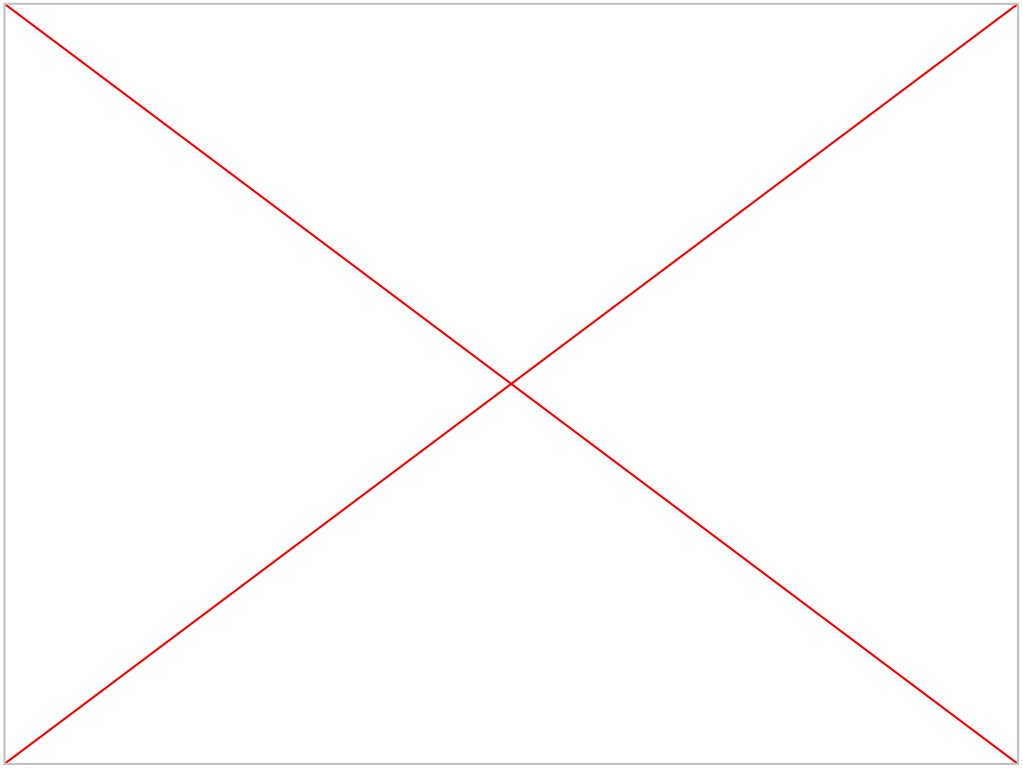
改善案

- **弾のクールダウン**: プレイヤーが弾を連続して撃ちすぎないように、弾発射にクールダウンを設定する。
- **攻撃エフェクト**: 弾が当たった時のエフェクト（爆発など）を追加。
- **スコア表示**: 各プレイヤーのスコアを表示し、ラウンド制にする。

このコードを基にさらに機能を追加して、ゲームの面白さを増やしていくことができます！

弾のクールダウン: プレイヤーが弾を連続して撃ちすぎないように、弾発射にクールダウンを設定する。
攻撃エフェクト: 弾が当たった時のエフェクト（爆発など）を追加。
スコア表示: 各プレイヤーのスコアを表示し、ラウンド制にする。

弾のクールダウン、攻撃エフェクト、スコア表示とラウンド制を追加したゲームのコードを以下に示します。これにより、プレイヤーが弾を連続して撃ちすぎないように制限を加え、弾が当たったときに爆発エフェクトを表示し、スコアを追跡してラウンド制を導入します。



ボードゲーム（双六，オセロ：中村）

プロセッシングを使ってボードゲームを作って

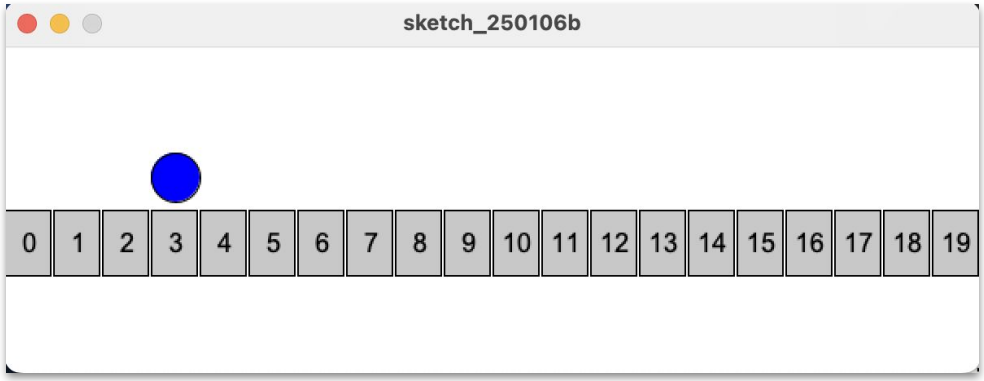
プロセッシングでボードゲームを作るには、まず基本的なゲームのアイデアとルールを決め、それをプログラムで表現するための計画を立てることが大事です。ここでは、簡単な「すごろく（サイコロを振ってゴールを目指す）」のようなゲームを例にして、基本的なボードゲームの作り方を説明します。

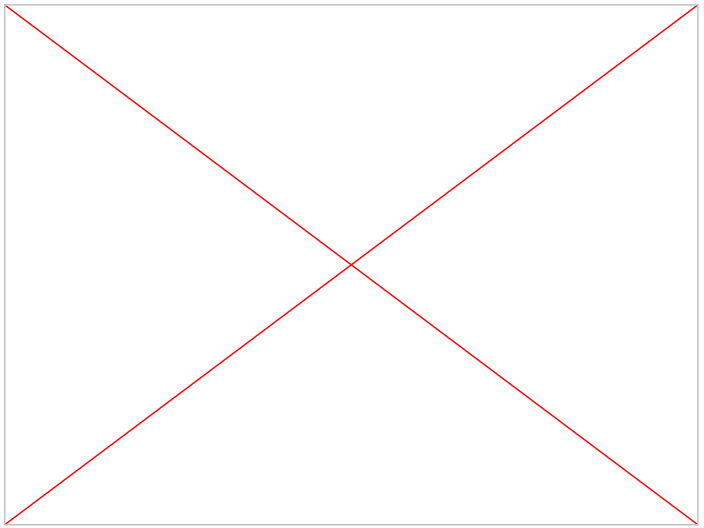
必要な準備

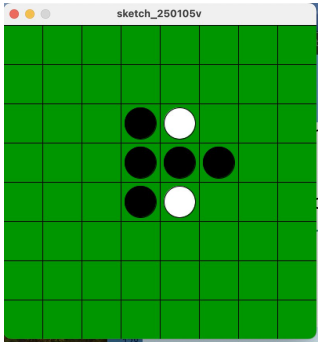
1. **ボードの描画**: ゲーム盤のマス目を描く。
2. **サイコロ機能**: ランダムにサイコロの目を決め、プレイヤーがその目の分だけ進めるようにする。
3. **プレイヤーの位置管理**: プレイヤーの位置をボード上で追跡し、移動するようにする。
4. **ゴール判定**: プレイヤーがゴールに到達したらゲームが終了する。

コード例

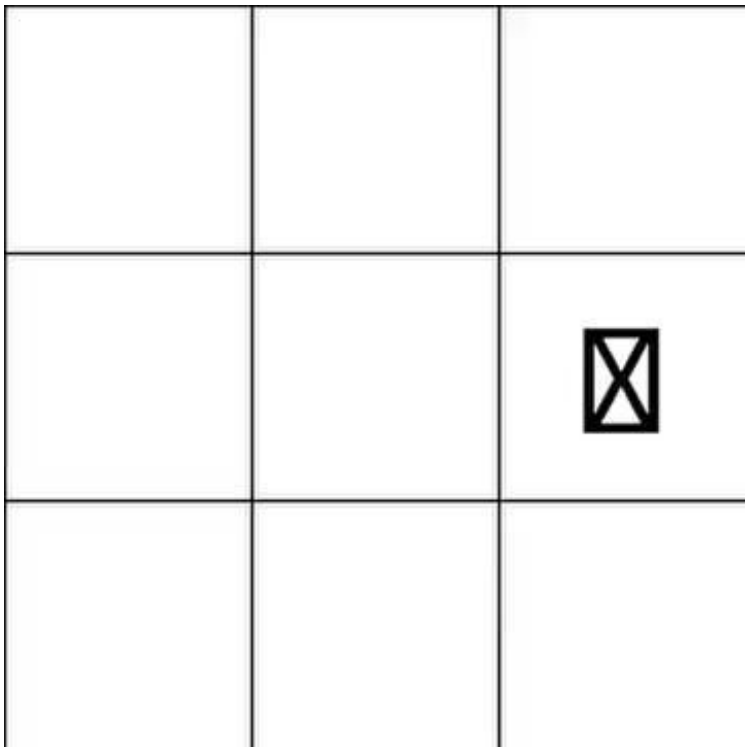
以下はプロセッシングでシンプルなすごろくゲームを実装する例です。

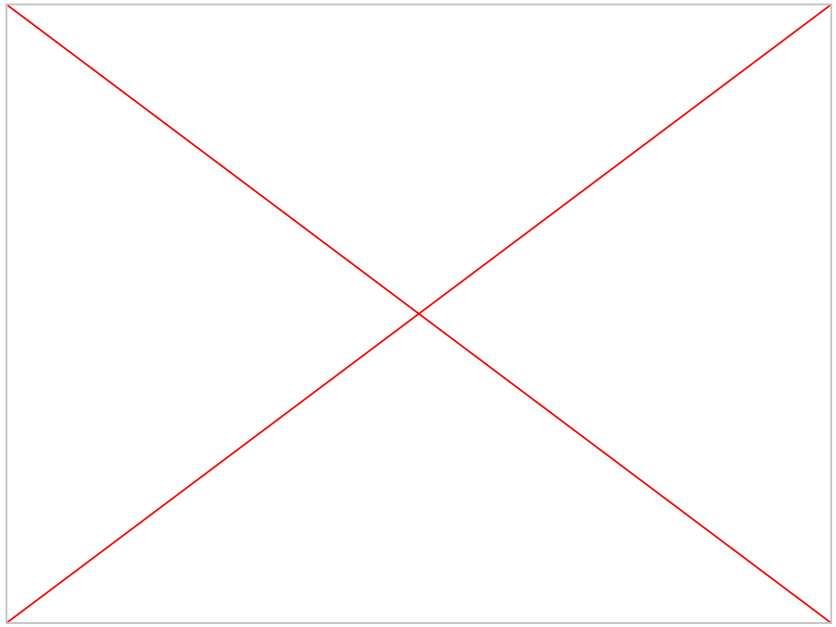






tiktoktoe（A君）チェス（櫻庭）





オセロ（C君）

石を置けるところに印が欲しいです

石を置ける場所にマークを表示するために、`drawBoard`関数で有効な場所に印を描画する処理を追加します。これにより、プレイヤーが石を置ける位置が視覚的に分かるようになります。

以下のように、`drawBoard`関数にて各セルが有効な手かどうかをチェックし、有効な場合には薄い色の円を描画するようにコードを修正します。

プロセッシングで対戦型のオセロゲームを作ってください。プレイ中に今、お互いに何枚なのか確認できるようなスコアを表示するものも入れてください。

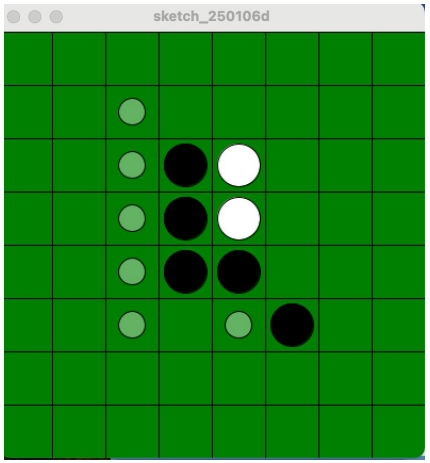
以下は、Processingで対戦型のオセロゲームを作るコードです。スコアは画面の上部にリアルタイムで表示され、お互いの石の数を確認できるようになっています。

もし、どちらかの色が置けない場合はスキップさせる

どちらかのプレイヤーが置ける場所がない場合にターンをスキップする機能を追加するために、以下のコードを修正します。この修正では、次のような流れでターンをスキップします。

1. プレイヤーが石を置こうとするたびに、相手のプレイヤーに置ける場所があるかを確認します。
2. 相手プレイヤーが置ける場所がなければ、もう一度現在のプレイヤーのターンとなります。
3. 両プレイヤーとも置ける場所がない場合にはゲーム終了の判定とします。

以下に、スキップ機能を追加した修正版のコードを示します。



オセロ（B君）

プロセッシングで対戦型のオセロゲームを作ってください。プレイ中に今、お互いに何枚なのか確認できるようなスコアを表示するものも入れてください。

以下は、Processingで対戦型のオセロゲームを作るコードです。スコアは画面の上部にリアルタイムで表示され、お互いの石の数を確認できるようになっています。

追加・修正点の説明

- `canPlay` 関数：指定したプレイヤーが有効な手を持つかどうかを確認します。配置可能な場所があれば `true`、なければ `false` を返します。
- `mousePressed` 内の `canPlay` 呼び出し：相手プレイヤーに有効な手がない場合はターンをスキップし、再度現在のプレイヤーのターンに戻します。
- `displayScore` 内のゲーム終了判定：両プレイヤーとも置ける場所がない場合、ゲームを終了して「ゲーム終了」と表示します。

相手プレイヤーをＡＩで操作して一人でも楽しめるようにして

AIを導入し、相手プレイヤーをコンピュータが操作するようにして一人で遊べるようにします。AIは簡単な戦略に基づいて行動し、最も多くの石をひっくり返せる場所を優先して選びます。

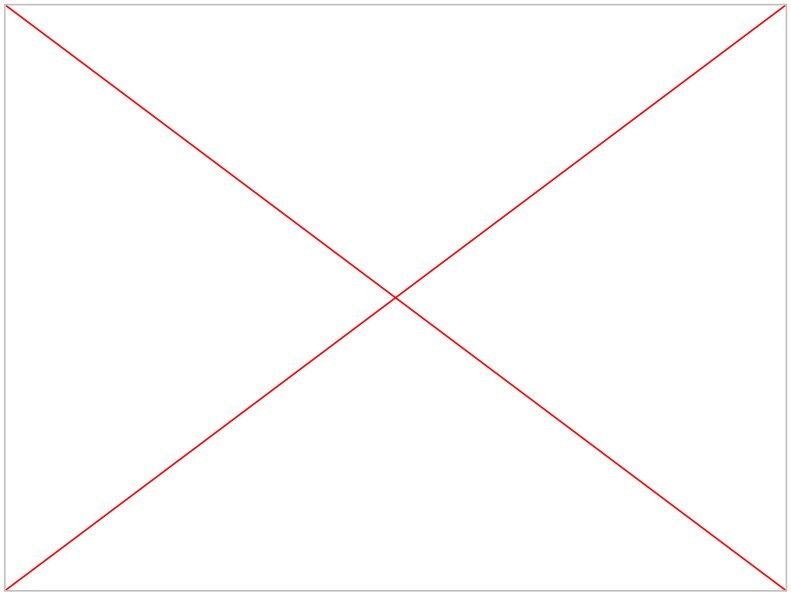
以下は、AIプレイヤーを追加したオセロゲームのコードです。

もし、どちらかの色が置けない場合はスキップさせる

どちらかのプレイヤーが置ける場所がない場合にターンをスキップする機能を追加するために、以下のコードを修正します。この修正では、次のような流れでターンをスキップします。

1. プレイヤーが石を置こうとするたびに、相手のプレイヤーに置ける場所があるかを確認します。
2. 相手プレイヤーが置ける場所がなければ、もう一度現在のプレイヤーのターンとなります。
3. 両プレイヤーとも置ける場所がない場合にはゲーム終了の判定とします。

以下に、スキップ機能を追加した修正版のコードを示します。



lesson learned

chatGPTは鏡

基礎は大事.

- 基本的な（文法レベル）こと
- 設計の概念

チューターとしては優秀

- 先生の言うことが理解できないといけない
- モチベーションを高めるには

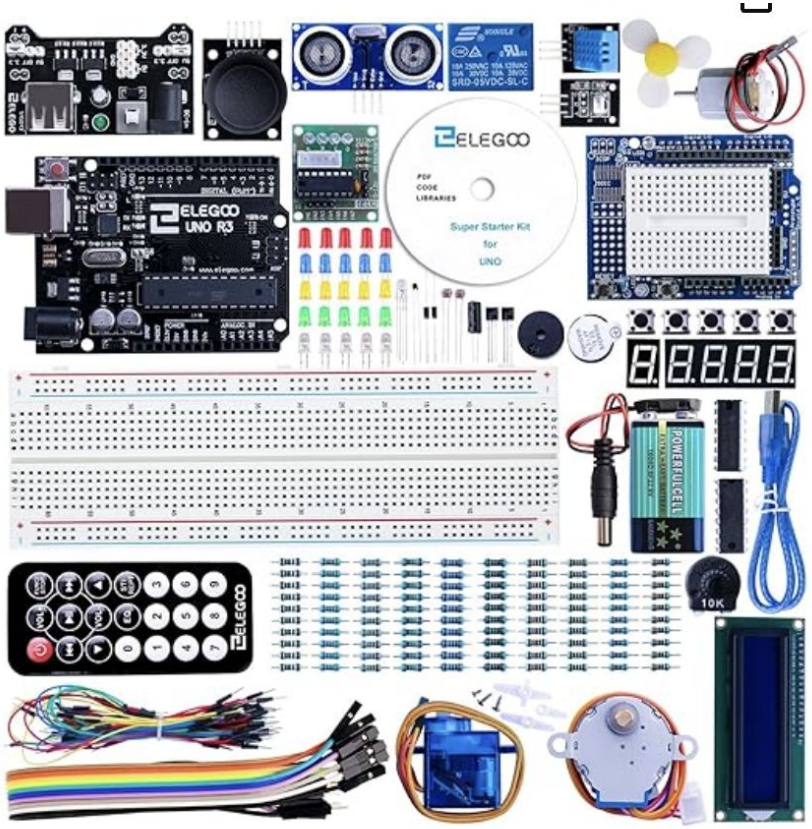
使い所が大事

- あらかじめ計画しておくといい.
- 改善，リファクタリングなど，ピンポイントでテーマを決めてやるといいかも
  - fisicaのゲーム(オブジェクト指向API)
  - processingのリファクタリング
  - シューティングゲームの壁打ち
  - オセロでAI

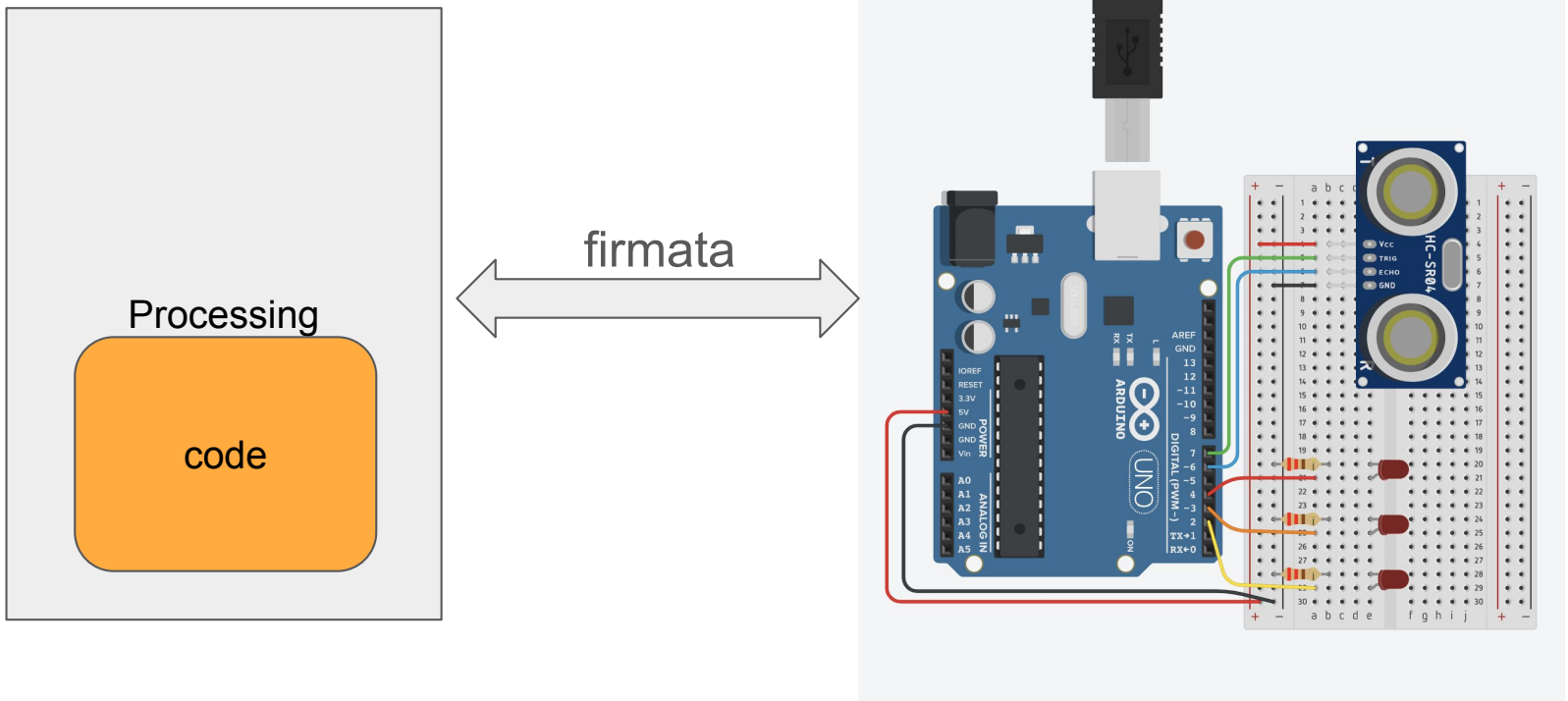
対象

青森大学ソフトウェア情報学部　2年生　プログラミングワークショップ

ArduinoとIoT入門



ArduinoとfirmataライブラリをIEった例



firmata+サーボ (A君）

firmataを使って意味のあるものを作成したいです。手順を教えてください

Firmataを使ってArduinoとProcessingを連携させる手順を説明します。Firmataを使用すると、Arduinoのコードをほとんど書かずにProcessingから直接Arduinoのハードウェアを操作することができます。以下は、基本的なセットアップから簡単なプロジェクトまでの手順です。

1. Arduino側の準備

1. **Arduino IDEのインストール**：Arduino IDEをダウンロードし、インストールします（もしまだインストールしていない場合）。
2. **Firmataライブラリのインストール**：
   - Arduino IDEを開き、`ツール` → `ライブラリを管理` に移動します。
   - 「Firmata」で検索し、「Firmata」ライブラリをインストールします。
3. **StandardFirmataスケッチのアップロード**：
   - Arduino IDEのメニューから `ファイル` → `例` → `Firmata` → `StandardFirmata` を選択します。
   - 「StandardFirmata」スケッチをArduinoにアップロードします。

2. Processing側の準備

3. 簡単なプロジェクト例：LEDの制御

ここでは、ProcessingからArduinoのLEDを制御する例を紹介します。

接続：

- Arduinoの `13番ピン` にLEDを接続します（抵抗を忘れずに）。
- グラウンド(GND)にLEDのマイナス端子を接続します。

Processingのコード：

以下のコードをProcessingに貼り付けてください。

java　コードをコピーする

```
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int ledPin = 13;

void setup() {
```

5. 拡張アイデア

- ボタンを追加して、Processingでその状態をモニタリングする。
- ポテンショメータを接続して、Processingでリアルタイムに値をグラフ表示する。
- サーボモーターを制御する。

興味があるプロジェクトやセンサーがあれば、さらに詳しい手順をお教えします！

LED以外のものを使ってください

LED以外のコンポーネントを使用して、ArduinoとProcessingをFirmataで連携させるプロジェクトを考えます。今回は、**サーボモーター**を使ったプロジェクトを紹介します。Processingからサーボモーターの角度を制御する例です。

手順

1. 準備物

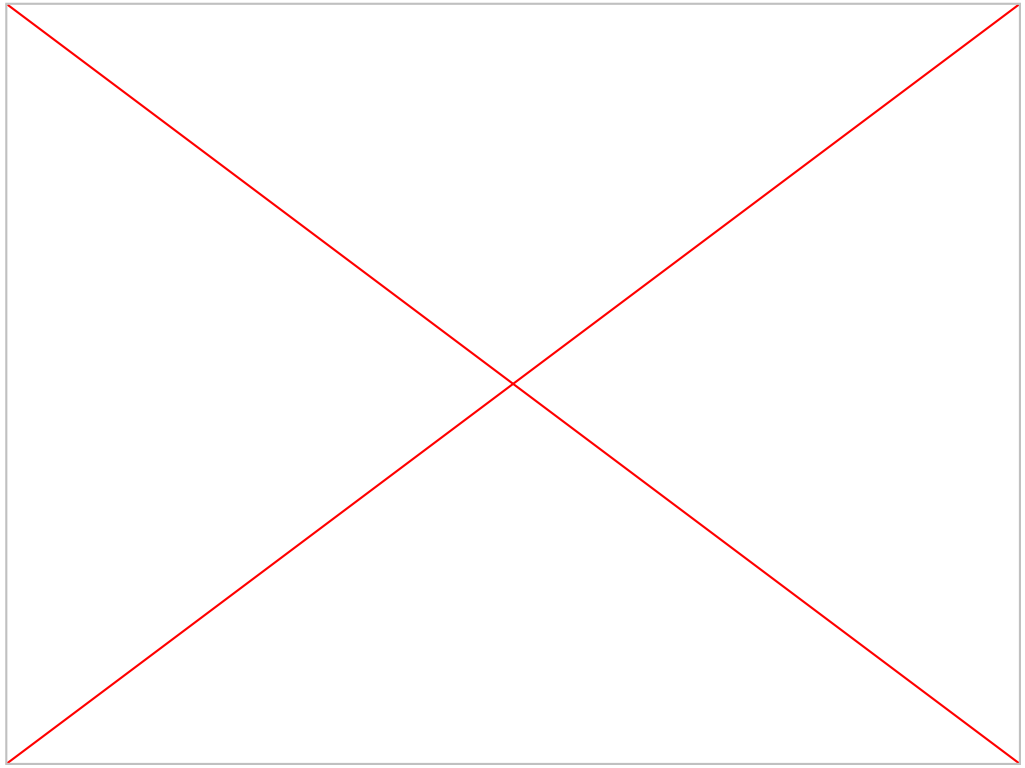
- Arduinoボード（Unoなど）
- サーボモーター
- ジャンパーケーブル
- Arduino IDE
- Processing IDE

2. Arduinoのセットアップ

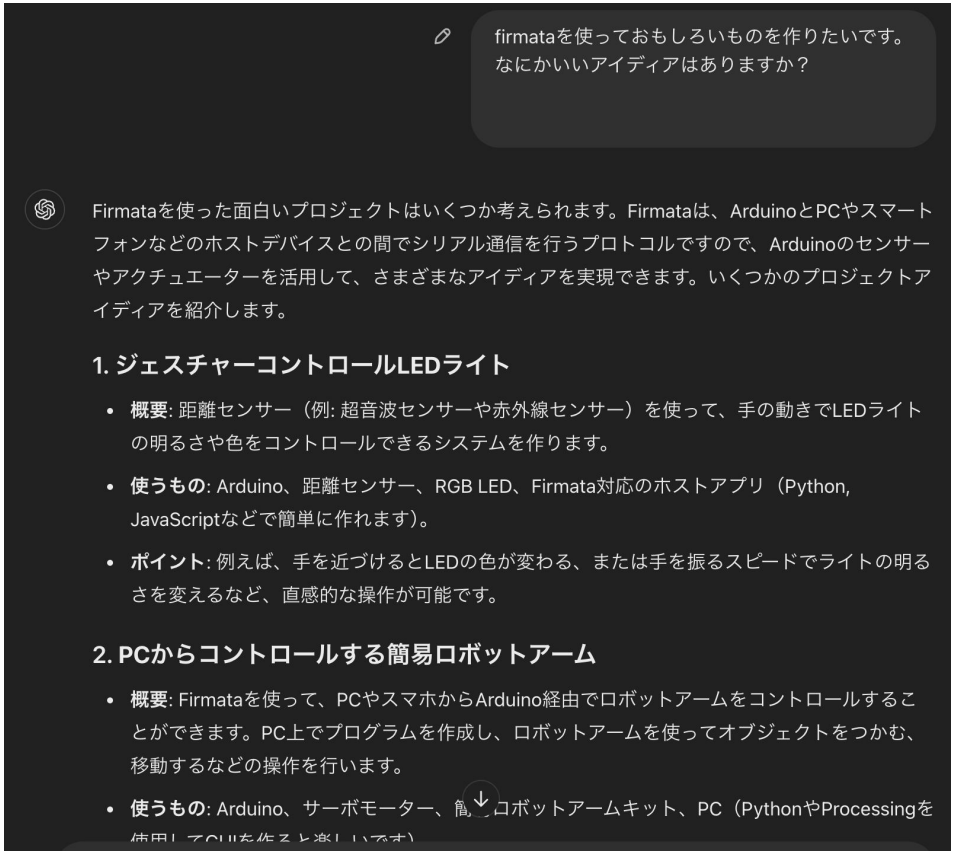
1. **Firmataのインストールとアップロード**:
   - Arduino IDEで「Firmata」ライブラリをインストールします（前述の手順を参照）。
   - 「StandardFirmata」スケッチをArduinoにアップロードします。
2. **サーボモーターの接続**:
   - サーボモーターの**信号線**を `9番ピン` に接続します。
   - サーボの**電源線**を `5V`、**GND線**を `GND` に接続します。

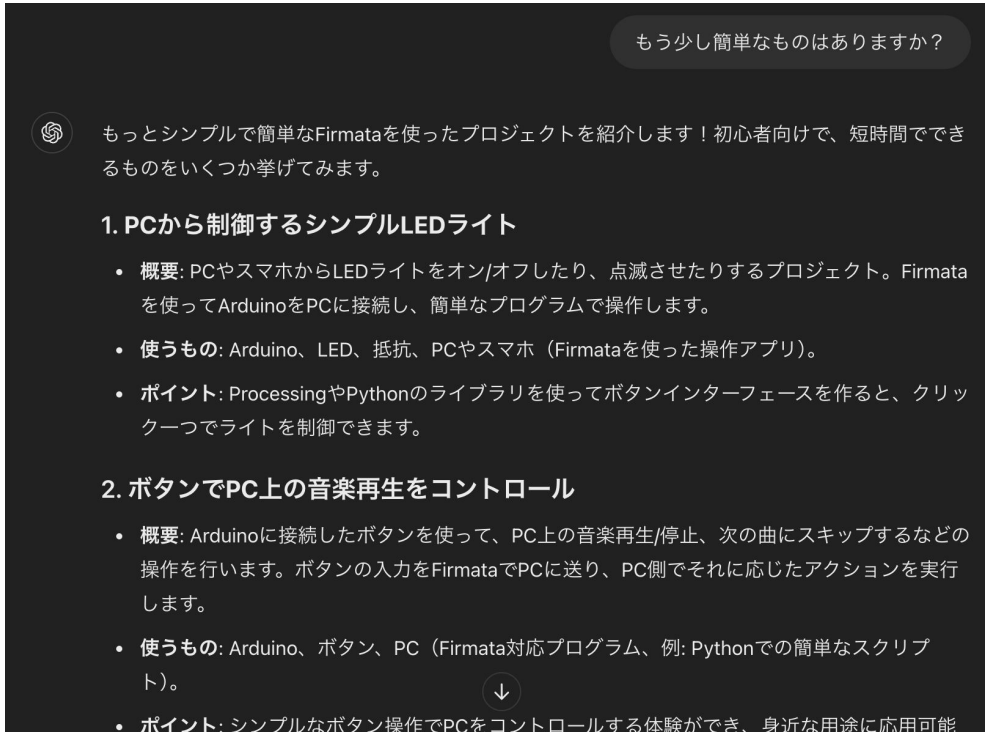
3. Processingのコード

firmata+サーボ (B君）



壁打ち（C君）


firmataを使っておもしろいものを作りたいです。なにかいいアイディアはありますか？
Firmataを使った面白いプロジェクトはいくつか考えられます。Firmataは、ArduinoとPCやスマートフォンなどのホストデバイスとの間でシリアル通信を行うプロトコルですので、Arduinoのセンサーやアクチュエーターを活用して、さまざまなアイディアを実現できます。いくつかのプロジェクトアイディアを紹介します。
1. ジェスチャーコントロールLEDライト
概要: 距離センサー（例: 超音波センサーや赤外線センサー）を使って、手の動きでLEDライトの明るさや色をコントロールできるシステムを作ります。
使うもの: Arduino、距離センサー、RGB LED、Firmata対応のホストアプリ（Python, JavaScriptなどで簡単に作れます）。
ポイント: 例えば、手を近づけるとLEDの色が変わる、または手を振るスピードでライトの明るさを変えるなど、直感的な操作が可能です。
2. PCからコントロールする簡易ロボットアーム
概要: Firmataを使って、PCやスマホからArduino経由でロボットアームをコントロールすることができます。PC上でプログラムを作成し、ロボットアームを使ってオブジェクトをつかむ、移動するなどの操作を行います。
使うもの: Arduino、サーボモーター、亀


もう少し簡単なものはありますか？
もっとシンプルで簡単なFirmataを使ったプロジェクトを紹介します！初心者向けで、短時間でできるものをいくつか挙げてみます。
1. PCから制御するシンプルLEDライト
概要: PCやスマホからLEDライトをオン/オフしたり、点滅させたりするプロジェクト。Firmataを使ってArduinoをPCに接続し、簡単なプログラムで操作します。
使うもの: Arduino、LED、抵抗、PCやスマホ（Firmataを使った操作アプリ）。
ポイント: ProcessingやPythonのライブラリを使ってボタンインターフェースを作ると、クリック一つでライトを制御できます。
2. ボタンでPC上の音楽再生をコントロール
概要: Arduinoに接続したボタンを使って、PC上の音楽再生/停止、次の曲にスキップするなどの操作を行います。ボタンの入力をFirmataでPCに送り、PC側でそれに応じたアクションを実行します。
使うもの: Arduino、ボタン、PC（Firmata対応プログラム、例: Pythonでの簡単なスクリプト）。
ポイント: シンプルなボタン操作でPCをコントロールする体験ができ、身近な用途に応用可能

壁打ち（C君）

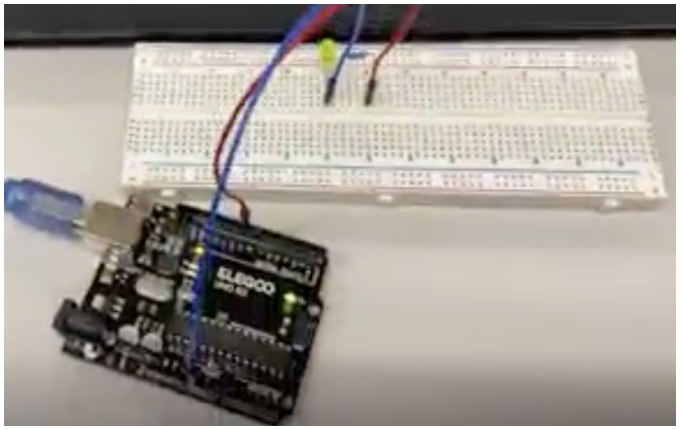
PCから制御するシンプルLEDライト
概要: PCやスマホからLEDライトをオン/オフしたり、点滅させたりするプロジェクト。Firmataを使ってArduinoをPCに接続し、簡単なプログラムで操作します。
使うもの: Arduino、LED、抵抗、PCやスマホ（Firmataを使った操作アプリ）。
ポイント: ProcessingやPythonのライブラリを使ってボタンインターフェースを作ると、クリック一つでライトを制御できます。これを作りたいです。まず何の部品が必要ですか？

PCから制御するシンプルLEDライトプロジェクトのために必要な部品は以下の通りです：

必要な部品

1. **Arduinoボード**（例: Arduino Uno, Nano, Megaなど）
   - Arduinoのボードは、PCとLEDの制御を仲介する役割を果たします。Unoが標準的で初心者向けです。
2. **LED**（1個）
   - 基本的な単色のLED（赤、青、緑など）。初心者には5mmの一般的なLEDが扱いやすいです。
3. **抵抗**（220Ω ～ 330Ω）
   - LEDに流れる電流を制限するための抵抗。220Ωから330Ω程度のものがLEDに適しています。
4. **ブレッドボード**（小型）

おねがいします



トラブルシューティング（D君）

反応しないなぜ

Firmataを使ったコードが反応しない場合、いくつかの原因が考えられます。以下のチェックリストを確認してください。

---

1. ポートの選択

Processingの`Arduino.list()`で取得されるポート一覧が正しいか確認してください。

- `println(Arduino.list());`で表示されるポートを確認し、Arduinoが接続されているポートを特定。
- `Arduino.list()[0]`の`0`を、正しいポートのインデックスに置き換えます。 例:

java　コードをコピーする

```
arduino = new Arduino(this, Arduino.list()[1], 57600); // インデックスを変更
```

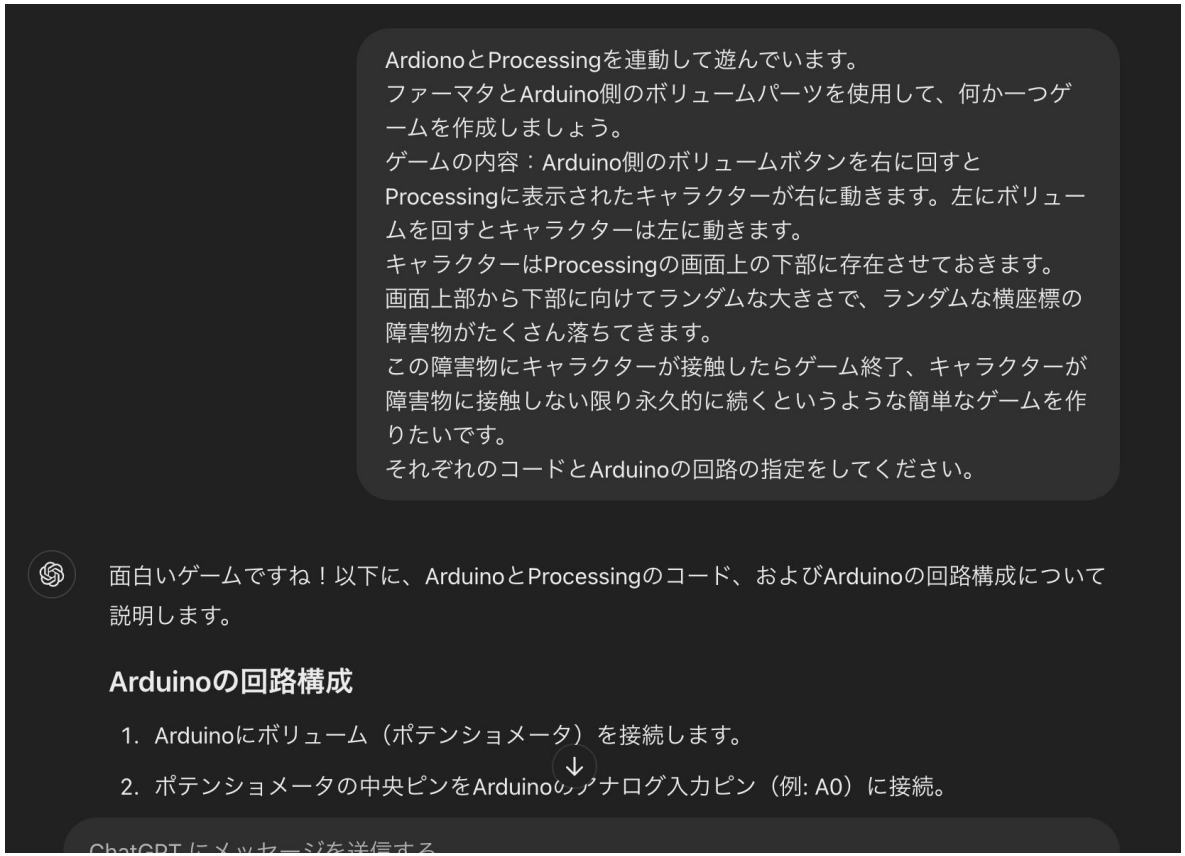
---

2. FirmataのスケッチがArduinoにアップロードされているか

Arduinoに`StandardFirmata`スケッチが正しくアップロードされていることを確認してください。

- Arduino IDEで`ファイル > スケッチ例 > Firmata > StandardFirmata`を選択し、アップロード

丁寧な受け答え(E君）


ArdionoとProcessingを連動して遊んでいます。
ファーマタとArduino側のボリュームパーツを使用して、何か一つゲームを作成しましょう。
ゲームの内容：Arduino側のボリュームボタンを右に回すとProcessingに表示されたキャラクターが右に動きます。左にボリュームを回すとキャラクターは左に動きます。
キャラクターはProcessingの画面上の下部に存在させておきます。
画面上部から下部に向けてランダムな大きさで、ランダムな横座標の障害物がたくさん落ちてきます。
この障害物にキャラクターが接触したらゲーム終了、キャラクターが障害物に接触しない限り永久的に続くというような簡単なゲームを作りたいです。
それぞれのコードとArduinoの回路の指定をしてください。
面白いゲームですね！以下に、ArduinoとProcessingのコード、およびArduinoの回路構成について説明します。
Arduinoの回路構成
1. Arduinoにボリューム（ポテンショメータ）を接続します。
2. ポテンショメータの中央ピンをArduinoのアナログ入力ピン（例: A0）に接続。

丁寧な修正（F君）

ArduinoとProcessing, Firmata, JoyStickを使ってミニゲームを作成してください。

ArduinoとProcessing、Firmata、ジョイスティックを使用してミニゲームを作成するためのシンプルなサンプルプロジェクトを紹介します。ここでは、基本的なミニゲームとして「ジョイスティックでボールを操作してゴールに到達する」というゲームを作成します。

必要なもの

- Arduinoボード（例えばArduino Uno）
- ジョイスティックモジュール
- USBケーブル
- Arduino IDE
- Processing IDE
- Firmataライブラリ

Processingコード

java　コードをコピーする

```
import processing.serial.*;
import cc.arduino.*;
import org.firmata.*;

Arduino arduino;
int xPin = A0; // ジョイスティックのX軸ピン
int yPin = A1; // ジョイスティックのY軸ピン

int ballX, ballY;
int goalX, goalY;
int ballSize = 20;
int goalSize = 40;
int windowWidth = 800;
int windowHeight = 600;

void setup() {
  size(windowWidth, windowHeight);
  arduino = new Arduino(this, Arduino.list()[0], 57600); // Arduinoに接続
```

時には人間が教えてあげる（F君）

A0 cannot be resolved to a variable

そのエラーは、`A0`という名前がProcessingのスケッチでは定義されていないために発生しています。Arduino IDEでは`A0`のようにアナログピンを表現できますが、Processingではそのような定数は自動的には提供されていません。

そこで、Arduinoのアナログピンを表すために数値を使用する必要があります。たとえば、`A0`は`0`に対応し、`A1`は`1`に対応します。コードを修正することで、この問題を解決できます。

修正済みのProcessingコード

java　コードをコピーする

```
import processing.serial.*;
import cc.arduino.*;
import org.firmata.*;

Arduino arduino;
int xPin = 0; // ジョイスティックのX軸ピン（A0は0に置き換え）
int yPin = 1; // ジョイスティックのY軸ピン（A1は1に置き換え）
```

IllegalStateException: size() cannot be used here, see https://processing.org/reference/size_.html

このエラーは、`size()`関数が`setup()`関数の外で呼び出されていることが原因です。Processingでは、`size()`は必ず`setup()`の中で定義する必要があります。

解決方法

`size()`関数を`setup()`の中に移動させてください。コードを以下のように修正します。

processingのバージョンは何系を想定して作成していますか？
3系以降ではsize関数のパラメータに変数を使うとエラーが出ます。
もし、3系より前を想定して作成している場合はバージョン4.3を想定してプログラムを作成して下さい。
そして、size関数のパラメータに変数を使う場合は以下のようにsettings関数の中でsize関数を使ってください。

```
void settings() {
 size(windowWidth, windowHeight);
 smooth();
}
```

最後に、settings関数はsetup関数の前に書きます。

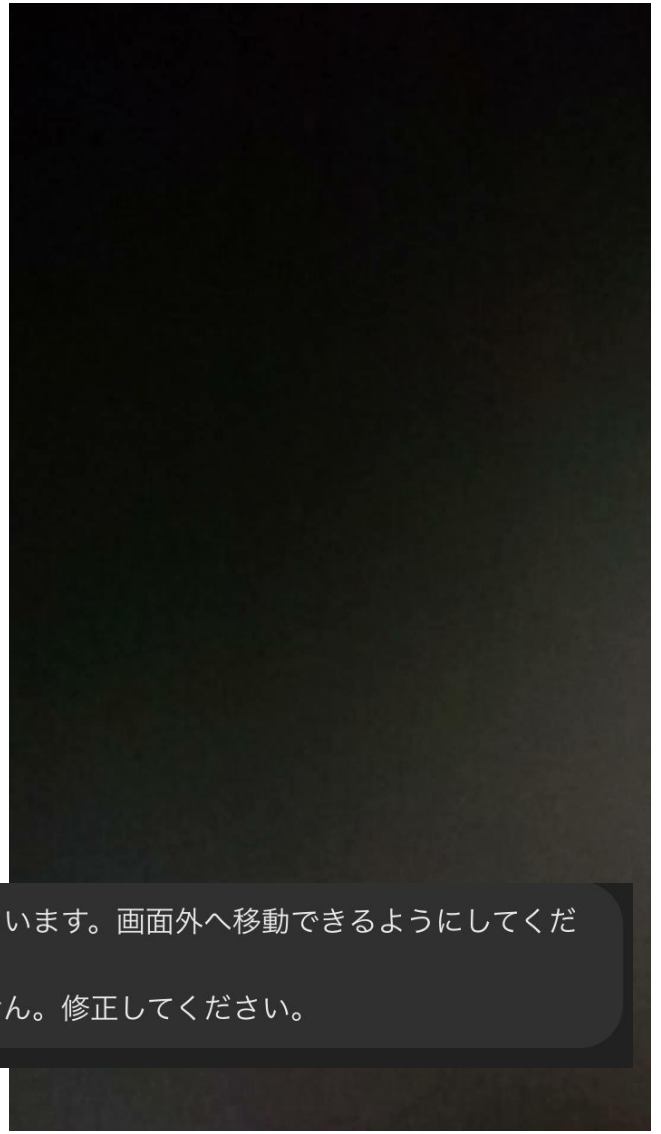
確かに、Processing 3系以降では`size()`関数で変数を使用する場合に、`settings()`関数を用いる必要があります。バージョン4.3でもこの方法は推奨されており、`size()`と`smooth()`は`settings()`内で定義するようにしてください。

丁寧な修正（F君）

ゲーム内容の変更をします。
ユーザーが動かす円(自機)にほかの円(敵)が当たるとゲームオーバーです。
時間制限を設けます。
1分以上時機が敵に当たらなければゲームクリアです。
敵は画面外から自機に向かって移動します。(表示された時点での時機の座標に向かって移動します。)
同時に画面上に登場できる敵は5個までです。

また、文字化けしてしまっているので改善してください。

敵が画面外から画面内、画面外の順番で移動するようにしてください。
敵が最後に画面外へ移動したら敵を削除して新たな敵を生成するようにしてください。
以上をループさせてください。
また、joystickの挙動がおかしいので修正してください。
joystickと時機の移動を同期させられる様に左右上下反転させてください。



敵が中央で止まってしまいます。画面外へ移動できるようにしてください。
文字化けが直っていません。修正してください。

改善を重ねていく（G君）

Arduinoとprocessingで、ボリュームを使用して、firmataを活用したいのですが、いくつか案をください

4番の3Dオブジェクトの回転が気になります。

発展アイデア

- **複数のポテンショメータを使用:**
  - X軸とY軸を別々のポテンショメータで制御する。
- **オブジェクトの種類を変更:**
  - 立方体ではなく球体や円柱などを描画。
- **ビジュアルエフェクト:**
  - 回転角度に応じてオブジェクトの色やサイズを変化させる。
- **複数オブジェクト:**
  - 回転する複数のオブジェクトを描画。

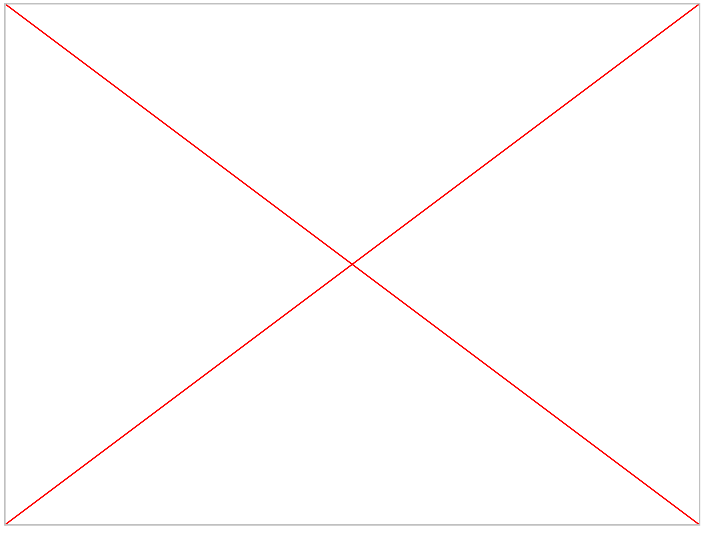
これをベースにカスタマイズしてみてください！疑問があればお知らせください。

ビジュアルエフェクトをやってみたいです

追加のカスタマイズ案

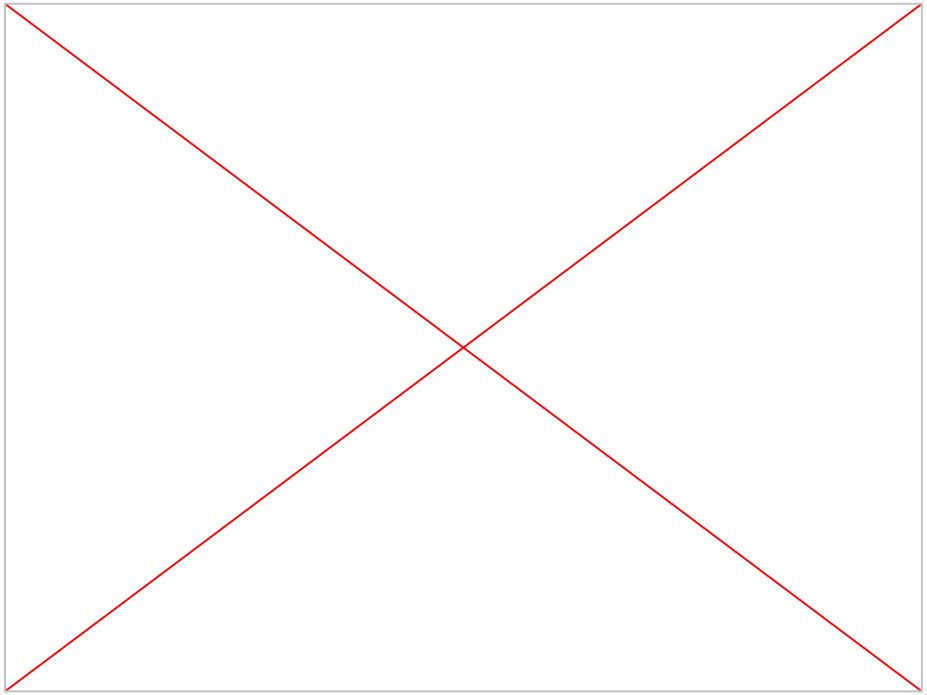
- **透明度の変化:** `fill()` の最後のパラメータを変化させて透明度を加える（例: `fill(red, green, blue, alpha)`）。
- **複数オブジェクトの描画:** 例えば、複数の小さなボックスを同心円状に配置し、それぞれ異なる回転や色を持たせる。
- **カメラの動き:** `camera()` 関数を使い、視点を動かすことでさらに立体感を演出。

では、先ほどの提案にあった、カメラの動きを追加したいです。



一発でうまくいく（H君）

Aruduinoでfirmataを使ってprocessingで動くプログラムを作成して欲しいです。
可変抵抗期を動かすと画面左側にある丸が上下に動く。
正方形が右から左に流れるので、それにプレイヤーが当たるとゲームオーバーになる。



GPTがチュータ（H君）

ProcessingとArduinoを使ってアナログの世界とサイバーをつなぐ作品を一つつくる。今、ProcessingとArduinoにFirmataというライブラリを挿入した。これを使って作品を作る。なお、Arduinoのポート番号は"COM49"とする。

ProcessingとArduinoを使って、アナログの世界（センサーなど）とサイバー（デジタル表示など）をつなぐ作品を作る際に、Firmataライブラリを使うことで簡単に通信が可能になります。ここでは、例として以下の作品を作ります：

作品の概要

- Arduinoで光センサー（フォトレジスタ）からアナログ値を取得。
- Processingでその値をもとに円の大きさや色を変化させる。

必要なもの

1. Arduino
2. フォトレジスタ（光センサー）
3. 抵抗（10kΩなど）
4. ブレッドボードとジャンパー線

手順

1. ハードウェア接続

- フォトレジスタを抵抗と直列に接続し、分圧回路を作成します。
- Arduinoの `A0` ピンに分圧回路の出力を接続。
- 回路図：

diff　　コードをコピーする

```
+5V ---- フォトレジスタ ---- A0 ---- 抵抗 ---- GND
```

フォトレジスタは２つ使うということですか。

いいえ、フォトレジスタは**1つ**だけ使用します！抵抗と組み合わせて、分圧回路を作るだけです。フォトレジスタと抵抗を直列に接続し、その中間点からアナログ入力（A0）に電圧を送る構造です。

フォトレジスタと抵抗の役割

- **フォトレジスタ**
  光の強さによって抵抗値が変わります。光が強いと抵抗が小さくなり、暗いと抵抗が大きくなります。
- **抵抗**
  電圧の変化を分かりやすくするために、フォトレジスタと一緒に分圧回路を作ります。この抵抗は固定値（例：10kΩ）で構いません。

接続の具体例

1. 必要なパーツ

- フォトレジスタ：1つ
- 抵抗（10kΩ）：1つ

GPTがチューター（H君）

よくある質問

1. フォトレジスタを2つ使うのか？

→ いいえ、1つだけで十分です。抵抗がもう1つの部品に見えるため混乱するかもしれませんが、抵抗はフォトレジスタと一緒に動作する補助部品です。

2. 抵抗の値は10kΩ固定ですか？

→ 基本的には10kΩが一般的ですが、用途によっては1kΩや100kΩなどでも試してOKです。光センサーの感度が変化します。

3. GNDと5Vを間違えるとどうなる？

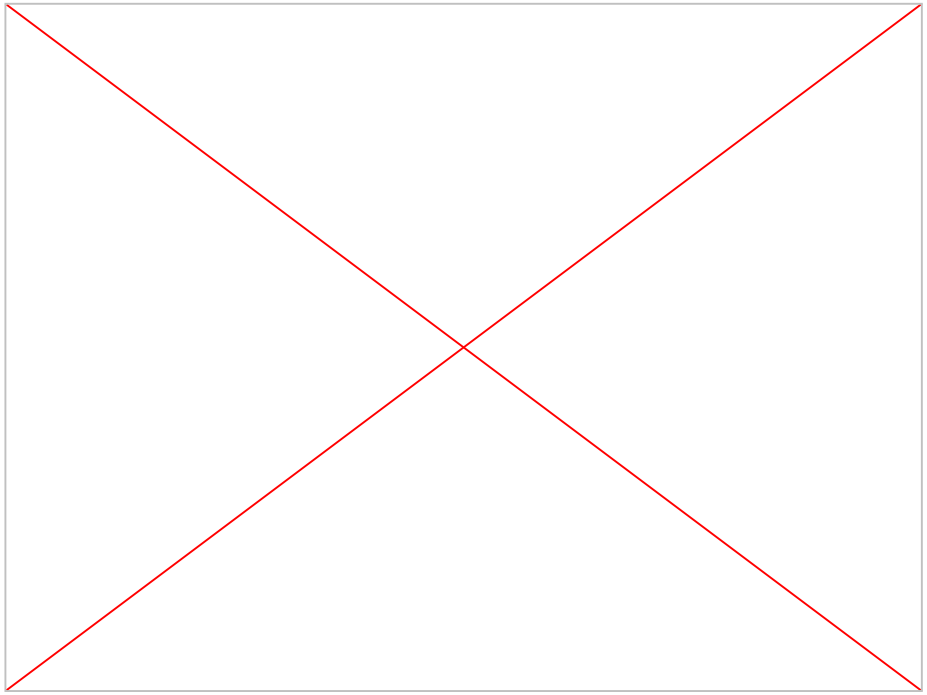
→ 正しく動作しない、もしくはセンサーが故障する可能性があるため、必ず確認してください。

この接続で問題がないはずです！もし不明点があれば、どこがわからないか教えてください 😊

フォトレジスタは列の向きに刺すのか？

フォトレジスタはブレッドボードの**列**（縦方向）に挿してください！

ブレッドボードでは、**同じ列に挿した端子同士は電気的に接続される**というルールがあります。フ



APIで間違える

Arduino UNO R3とProcessingでファーマタとボリュームスティックを使って360度カメラを作成し、回路とプログラムを表示してください。

Arduino UNO R3とProcessingを使って、ボリュームスティック（アナログジョイスティック）を利用した360度カメラコントローラーを作成する例を紹介します。このプロジェクトでは、ファーマタ（Firmata）プロトコルを用いてArduinoとProcessingを連携させ、ジョイスティックの入力でカメラを制御します。

以下に、**回路図**と**プログラムコード**を示します。

必要な部品

1. Arduino UNO R3
2. ボリュームスティック（アナログジョイスティック）
3. サーボモーター（カメラの回転用）
4. ブレッドボードとジャンパーワイヤー
5. 360度カメラ（シミュレーション用なら実際のカメラは不要）

java　　コードをコピーする

```
import processing.serial.*;
import cc.arduino.*;

Arduino arduino;
int servoPin = 9; // サーボモーターの接続ピン
int joystickX = 0; // ジョイスティックのX軸（A0）
int joystickY = 1; // ジョイスティックのY軸（A1）

void setup() {
  size(400, 400); // ウィンドウサイズ
  println(Arduino.list()); // 利用可能なシリアルポートを表示
  arduino = new Arduino(this, Arduino.list()[0], 57600); // シリアルポート接続

  // ピンモード設定
  arduino.pinMode(joystickX, Arduino.INPUT);
  arduino.pinMode(joystickY, Arduino.INPUT);
  arduino.pinMode(servoPin, Arduino.SERVO);
}
```

No library found for cc.arduino
Libraries must be installed in a folder named 'libraries' inside the sketchbook folder (see the Preferences window).
The package “cc” does not exist. You might be missing a library.
というエラーが出ました。改善をお願いします

このエラーは、ProcessingにFirmataライブラリがインストールされていない場合に発生します。以下の手順で問題を解決してください。

同様に菅野さん

2回目のコード

java　　コードをコピーする

```java
import processing.serial.*;
import org.firmata.*;

Firmata arduino;
int servoPin = 9; // サーボモーターの接続ピン
int joystickX = 0; // ジョイスティックのX軸（A0）
int joystickY = 1; // ジョイスティックのY軸（A1）

void setup() {
  size(400, 400); // ウィンドウサイズ
  println(Firmata.list()); // 利用可能なシリアルポートを表示
  arduino = new Firmata(this, Firmata.list()[0]); // シリアルポート接続

  // サーボモーターの設定
  arduino.servoConfig(servoPin, 0, 180);
}
```

パッケージが変わってる

混乱が始まる.

The function list() does not exist.

The constructor Firmata(sketch_241119a, String) is undefined

Firmataのver.2.5.9に合わせて作成しなおしてください

lesson learned

chatGPTは鏡

基礎は大事.

　　基礎がわかっているとかなりのことができる

チューターとしては優秀

　　モチベーションを高めるには

　　学術会議で生成AIに関する提言を取りまとめ中