# p5.js 入門

## メディアアートを体験しよう

2021 年度　情報デザインコース 2 年 「情報テクノロジー」教材

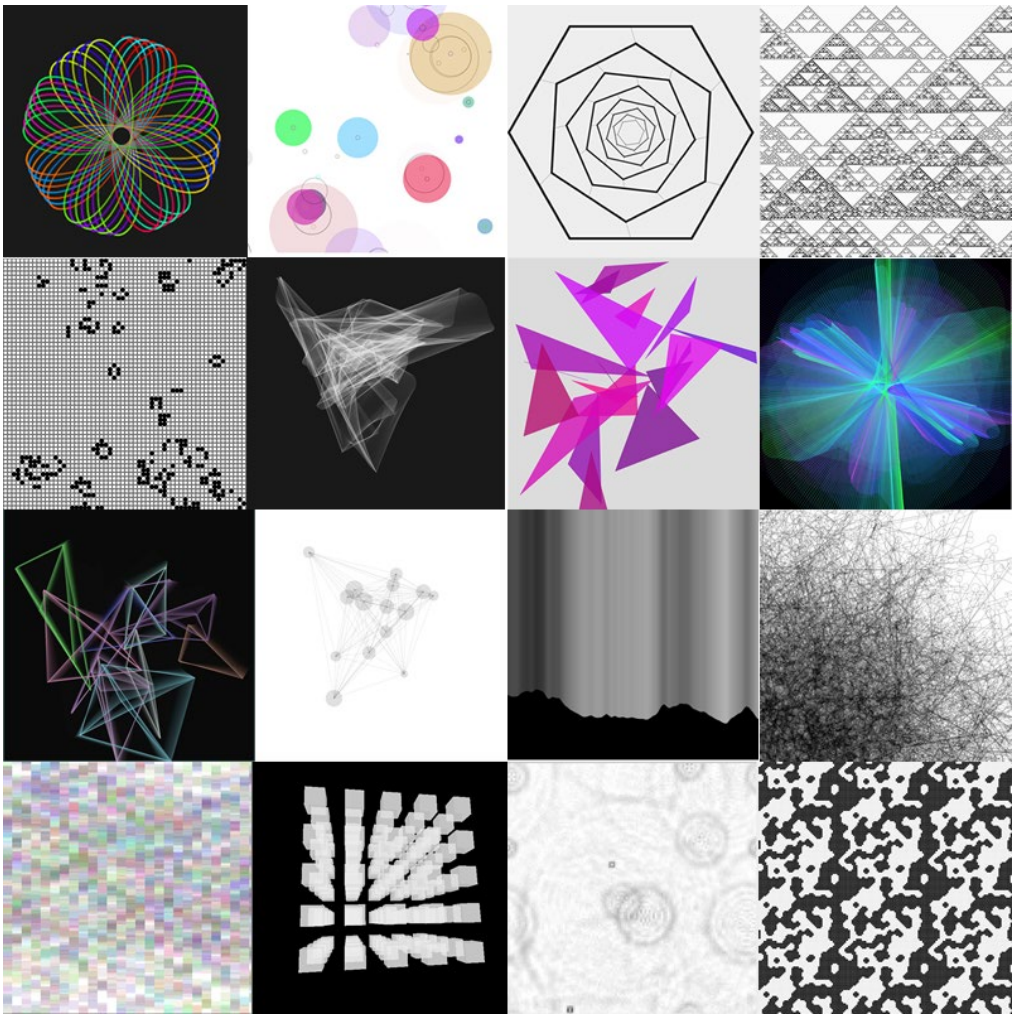
# 目次

- この教材のコードは、下記サイトを参考に作成しています。
  - 「メディアアート・プログラミング　I　2021」https://yoppa.org/geidai-media1-21

# I　この授業でやること

## 1　プログラミングとアート

- こういった絵を見たことありますか？これって、何で作られている？



https://infosmith.biz/blog/it/p5js-generativeart より
（カラーで見ることをおすすめします！）

- プログラミングって、専門家（エンジニアなど）がやるもの？
  - そうではありません！
  - 「他に専門を持ったプログラマー」が世の中にはたくさんいます。
  - 近年、アーティストも、プログラムを行うことで世に作品を送り出している人もいます。

## 2　p5.js とは？

- 「ぴーふぁいぶ どっと じぇいえす」と呼びます
- 「Javascript」というプログラミング言語のライブラリです
  - ライブラリ：いろいろな機能を詰め込んだ便利ツールと思ってください。p5.js というライブラリを用いることで、簡単に作品を作ることができるようになっています。

## 3　この授業の目標

- 「表現技法の一つ」として、プログラミングという手段があるということを体験する
- プログラミングでどのような表現ができるのかを、「p5.js」を通して体験し、作品を制作する

## 4　評価の方法

- 評価対象（変更の可能性があります）：
  - 授業中の演習課題・・・平常点としてカウント
  - 「まとめの課題」3 つ（予定）・・・課題点としてカウント

# II　はじめの準備とはじめの一歩

## 1　はじめの準備

- 「BitArrow」を使用します。
  URL：https://bitarrow.eplang.jp/bitarrow/

| ※ここで、ログイン方法の説明（クラス名やユーザ名など）の説明をしていました |
|---|

## 2　はじめの一歩（試しに書いてみよう）

- 下記を「firstStep」に入力し、ブラウザを更新しよう。

```
【firstStep】※下線部を追加
function setup(){
  createCanvas(800,800);
  background(255,255,255);
}
function draw(){
  fill(0,0,0,70);
  ellipse(mouseX,mouseY,random(100));
}
```

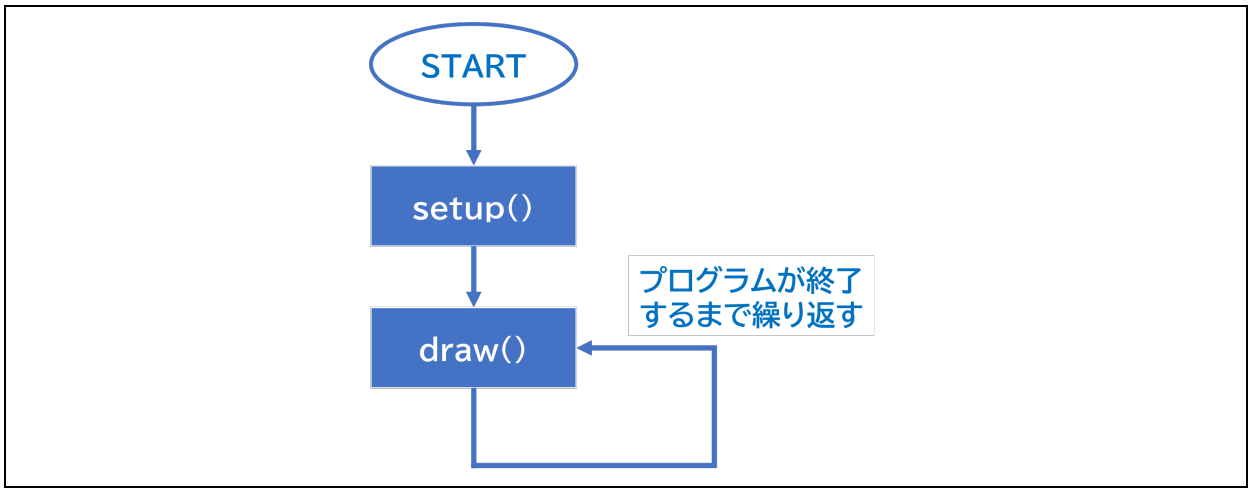
## 3　プログラムの書き方

- 半角文字で書く（全角はダメ）
- 大文字と小文字は区別される
- 文末にはセミコロン“;” を入れる
- 「中かっこ」を対応させること！
  - おなじ括弧に囲まれている部分がひとつのブロック・いろいろな括弧が入れ子構造になっている
- ブロック中で、上から順番にプログラムが実行される

# III　基本知識

## 1　function setup()と function draw()

- アニメーションを実現する仕組み
- パラパラ漫画のイメージ
- 両者の関係




## 1-1　function setup()とは？

- いわゆる「初期設定」
  - ➢ プログラムの起動時に一度だけ実行される
  - ➢ アニメーションの「前準備」をここで行う

【setup()の中で行われることの多い処理　※他にも様々な処理が書けます】
- createCanvas：画面のサイズを設定
- background：背景色の指定

- setup()の中を書き換えよう

【firstStep】 ※下線部を変更しよう

```
function setup(){
  createCanvas(800,800);
  background(214,233,202);
}
function draw(){
  fill(0,0,0,70);
  ellipse(mouseX,mouseY,random(100));
}
```

背景色はどう変わった？

## 1-2　function draw()とは？

- 「描画」を行うパート
- function setup()で設定した環境内で、プログラムが終了するまでくりかえし実行される
- ループの中で図形の場所や色、形を操作してアニメーションにする
  - ここでは、fill(～)で、塗りつぶしの色を決め、ellipse()でその色の円を描画している。これを何回も繰り返している

- draw()を書き換えよう

```
【firstStep】 ※下線部を変更しよう
function setup(){
  createCanvas(800,800);
  background(214,233,202);
}
function draw(){
  fill(random(255),random(255), random(255),70);
  ellipse(mouseX,mouseY,random(100));
}
```

どんな変化が起きただろう？

- さらに書き換えよう

```
【firstStep】 ※下線部を変更しよう
function setup(){
  createCanvas(800,800);
  background(214,233,202);
}
function draw(){
  fill(random(255),random(255), random(255),70);
  ellipse(random(width),random(height),random(100));
}
```

## 2　関数

- 決められた処理を実行してくれる便利な機能のこと
  - 例：Excel だと、SUM 関数や AVERAGE 関数→合計や平均を計算し、答えを返してくれる
- p5.js にも、ビジュアルプログラミングのための関数がたくさん準備されています
  - これまでに書いたコードの関数の一部の紹介（詳細は後述）

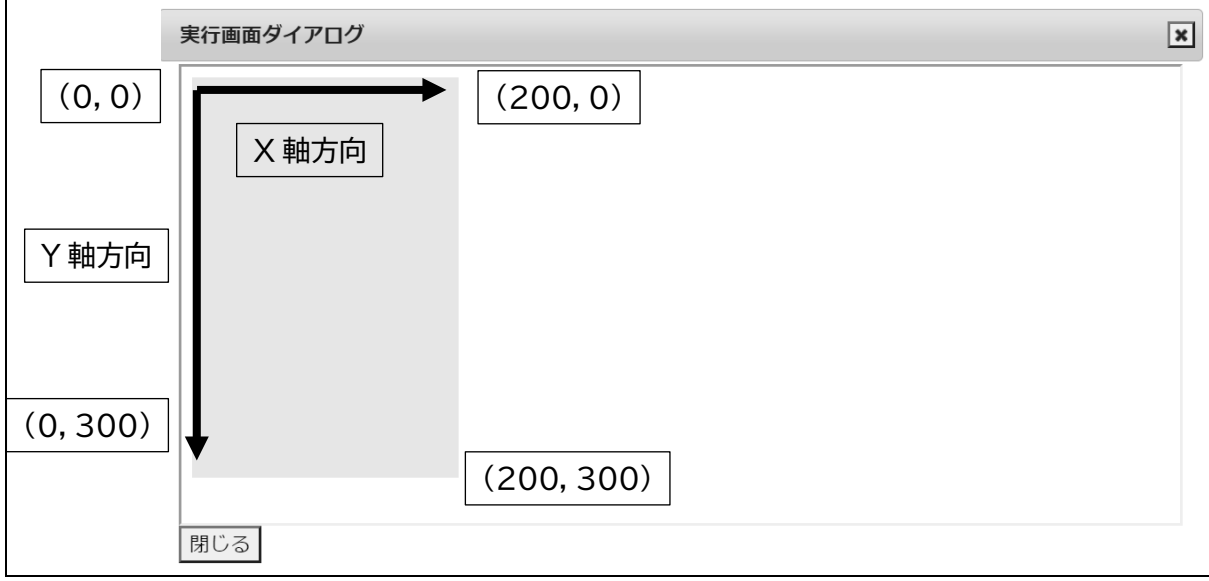
| | |
|---|---|
| createCanvas 関数 | 指定された描画領域を用意する |
| background 関数 | 指定された色で描画領域の背景を塗りつぶす |
| random 関数 | ランダムに生成された数値を返す |

- 関数は、「関数名」と「引数」で構成される
  - 基本的な書き方は、関数名(引数 1, 引数 2, 引数 3...);
  - 引数の数は関数によって異なる
- p5.js の関数一覧は、こちらへ
  - 「P5.js 日本語リファレンス」：
    https://qiita.com/bit0101/items/91818244dc26c767a0fe
  - 他にも様々なサイトがあるので、調べてみてください。

## 3　座標系

- 左上が原点（0, 0）
- 右に行くほど x 座標の値が増える
- 下に行くほど y 座標の値が増える

【createCanvas(200,300)の正体とは？】
左上が原点



# IV　かたちと色の表示

使うプロジェクト：shape_and_color

## 1　図形の表示

### 1-1　本項で紹介する図形描画関数一覧

| | |
|---|---|
| point 関数 | 点を描画する |
| line 関数 | 線を描画する |
| triangle 関数 | 三角形を描画する |
| rect 関数 | 長方形を描画する |
| quad 関数 | 四角形を描画する |
| ellipse 関数 | 円（正円・楕円）を描画する |

### 1-2　point 関数：点を描く

- 書き方：point(＜X 座標＞,＜Y 座標＞);

```
【shape】※下線部を追加
function setup() {
  createCanvas(800, 600);
  background(220);
}
function draw() {
  point(100, 200);
}
```

X 座標 100, Y 座標 120 の位置に点を描く

### 1-3　line 関数：直線を描く

- 書き方：line(＜X 座標始点＞,＜Y 座標始点＞,＜X 座標終点＞,＜Y 座標終点＞);

```
【shape】※下線部を追加
function setup() {
  createCanvas(800, 600);
  background(220);
}
function draw() {
  point(100, 200);
  line(80, 40, 700, 500);
}
```

始点(80,40)、終点(700,500)を結ぶ線ができる

## 1-4　rect 関数:長方形を描く

- 書き方:rect(<X 座標>,<Y 座標>,<長方形の幅>,<長方形の高さ>);

```
【shape】 ※下線部を追加
function setup() {
  createCanvas(800, 600);
  background(220);
}
function draw() {
  point(100, 200);
  line(80, 40, 700, 500);
  rect(200, 300, 400, 200);
}
```

(200,300)を左上の頂点とし、幅400、高さ 200 の四角形ができる

## 1-5　quad 関数:四角形を描く

- 書き方:

quad(<X座標 1>,<Y座標 1>, <X座標 2>,<Y座標 2>, <X座標 3>,<Y座標 3>, <X座標 4>,<Y座標 4>);

```
【shape】　※下線部を追加
function setup() {
  createCanvas(800, 600);
  background(220);
}
function draw() {
  point(100, 200);
  line(80, 40, 700, 500);
  rect(200, 300, 400, 200);
  quad(100,5, 180, 100, 170, 180, 30, 50);
}
```

左上(100,5)・右上(180,100)
右下(178,180)・左下(30,50)
を結ぶ四角形が作られる

## 1-6　ellipse 関数：円、楕円を描く

- 書き方：ellipse(＜X座標＞,＜Y座標＞,＜楕円の幅＞,＜楕円の高さ＞);

```
【shape】
function setup() {
  createCanvas(800, 600);
  background(220);
}
function draw() {
  point(100, 200);
  line(80, 40, 700, 500);
  rect(200, 300, 400, 200);
  quad(100,5, 180, 100, 170, 180, 30, 50);
  ellipse(500, 300, 300, 200);
}
```

（500,300）を中心とし、
幅 300、高さ 200 の円が作られる

## 2　色や輪郭(描画属性)の表示

### 2-1　描画属性の代表的な関数一覧

- 描画属性:描画を行う場合に必要な情報のこと(→線の幅・線の色・塗りつぶしの色など)

| | |
|---|---|
| background 関数 | 背景色を指定する |
| stroke 関数 | 線の色を指定する |
| strokeWeight 関数 | 線の幅を指定する |
| noStroke 関数 | 輪郭線の描画をオフにする |
| fill 関数 | 塗りつぶしの色を指定する |
| noFill 関数 | 塗りつぶしの色をオフにする |

### 2-2　background 関数:背景色をつける

- 書き方:background(<R の値>, <G の値>, <B の値>); など

【color】※下線部を変更

```
function setup() {
  createCanvas(800, 600);
  background(0,0,128);
}
function draw() {
  point(100, 200);
  line(80, 40, 700, 500);
  rect(200, 300, 400, 200);
  quad(100,5, 180, 100, 170, 180, 30, 50);
  ellipse(500, 300, 300, 200);
}
```

- 色を指定する関数の引数の書き方は、いくつかのパターンがある

【3つの引数をとる場合:RGB カラー】

例:background(255, 255, 0);

↑ ↑ ↑

R G B

赤・緑・青の強さをそれぞれ 0～255 の 256 段階で表記する

【4つの引数をとる場合:透明度の設定】

例:background(255, 255, 0, 128);

↑ ↑ ↑ ↑

R G B 透明度

R,G,B のほか、透明度を 0～255 で表記できる。

0 に近いほど透明になる。255 にすると不透明になる。

【1つの引数をとる場合:グレースケールまたは色名の指定など】

例:background(128);

グレースケール値の設定。0 から 255 まで設定できる。

0 が真っ黒、255 が真っ白になる。

例:background(“red”);

ダブルクオーテーションで色名を表記することもできる。

## 2-3 線の描画属性

### 2-3-1 stroke 関数:線に色をつける

- 書き方:stroke(<R の値>, <G の値>, <B の値>); など

【color】※下線部を変更

```
function setup() {
  createCanvas(800, 600);
  background(255,255,0);
}
function draw() {
  stroke(255, 255, 31);
  point(100, 200);
  line(80, 40, 700, 500);
  rect(200, 300, 400, 200);
  quad(100,5, 180, 100, 170, 180, 30, 50);
  ellipse(500, 300, 300, 200);
}
```

- 図形により色を変える場合は、色を変えたい図形を描画する直前に関数を追加すればよい

```
【color】 ※下線部を追加
function setup() {
  createCanvas(800, 600);
  background(255,255,0);
}
function draw() {
  stroke(255, 255, 31);
  point(100, 200);
  line(80, 40, 700, 500);
  stroke(255, 0, 128);
  rect(200, 300, 400, 200);
  quad(100,5, 180, 100, 170, 180, 30, 50);
  ellipse(500, 300, 300, 200);
}
```

### 2-3-2 strokeWeight 関数:線の幅を指定する

- 線の幅を指定したい場合は、strokeWeight 関数を使う
- 書き方:strokeWeight(<線幅>);

```
【color】 ※下線部を追加
function setup() {
  createCanvas(800, 600);
  background(255,255,0);
}
function draw() {
  stroke(255, 255, 31);
  strokeWeight(5);
  point(100, 200);
  line(80, 40, 700, 500);
  stroke(255, 0, 128);
  strokeWeight(20);
  rect(200, 300, 400, 200);
  quad(100,5, 180, 100, 170, 180, 30, 50);
  ellipse(500, 300, 300, 200);
}
```

### 2-3-3 noStroke 関数：線を消す

- 線を消したい場合は、noStroke 関数を使う
- 書き方：noStroke();

```
【color】 ※下線部を追加
function setup() {
    createCanvas(800, 600);
    background(0,0,128);
}
function draw() {
    stroke(255, 255, 31);
    strokeWeight(5);
    point(100, 200);
    line(80, 40, 700, 500);
    stroke(255, 0, 128);
    strokeWeight(20);
    rect(200, 300, 400, 200);
    quad(100,5, 180, 100, 170, 180, 30, 50);
    noStroke()
    ellipse(500, 300, 300, 200);
}
```

## 2-4　塗りの色をつける

### 2-4-1 fill 関数

- 書き方:fill(＜R の値＞，＜G の値＞，＜B の値＞); など

```
【color】※下線部を追加
function setup() {
    createCanvas(800, 600);
    background(0,0,128);
}
function draw() {
    stroke(255, 255, 31);
    strokeWeight(5);
    point(100, 200);
    line(80, 40, 700, 500);
    stroke(255, 0, 128);
    strokeWeight(20);
    fill(31, 127, 255);
    rect(200, 300, 400, 200);
    fill(200, 0, 0);
    quad(100,5, 180, 100, 170, 180, 30, 50);
    noStroke();
    fill(31, 127, 255);
    ellipse(500, 300, 300, 200);
}
```

## 2-4-2　noFill 関数

- 書き方：noFill();

```
【color】 ※打ち消し線部をコメントアウト、下線部を追加
function setup() {
    createCanvas(800, 600);
    background(0,0,128);
}
function draw() {
    stroke(255, 255, 31);
    strokeWeight(5);
    point(100, 200);
    line(80, 40, 700, 500);
    stroke(255, 0, 128);
    strokeWeight(20);
    fill(31, 127, 255);
    rect(200, 300, 400, 200);
    ~~fill(200, 0, 0);~~
    noFill();
    quad(100,5, 180, 100, 170, 180, 30, 50);
    noStroke();
    fill(31, 127, 255);
    ellipse(500, 300, 300, 200);
}
```

## 2-5　色の塗り方の規則

- 色や線塗りつぶしの設定は、それ以降すべての描画に使われる
- 色を変えるには改めて別の色を設定する命令を入れる

## 3　まとめ課題：「かたちと色を使った制作」

- 使用するファイル：matome
- かたちと色を使い、自由に制作を行ってください
- 1～3 行目のコメント行に、出席番号・名前・タイトルを入力してください
- キャンバスの大きさは、(800,800)とします

# V　乱数を利用した描画

使用するプロジェクト：random

## 1　random 関数

- ある範囲の数値をランダムに生成させたいときは、ramdom 関数が使える。
- 例えば、
  - 表示色をランダムに変えたい
  - 表示させる座標をランダムに変えたい　など
- 書き方：random(<数値>);

```
【ramdom 初期画面】
function setup() {
  createCanvas(800, 600);
  background(255);
}
function draw() {
  strokeWeight(5);
  fill(255,0,0);
  ellipse(50,50,50,50);
}
```

- 円を増殖させる描画

```
【ramdom】※下線部を追加
function setup() {
  createCanvas(800, 600);
  background(255);
}
function draw() {
  strokeWeight(5);
  fill(255,0,0);
  ellipse(random(800),random(600),50,50);
}
```

- 円の色もランダムに表示させる描画

```
【ramdom】※下線部を変更
function setup() {
  createCanvas(800, 600);
  background(255);
}
function draw() {
  strokeWeight(5);
  fill(random(255),random(255),random(255));
  ellipse(random(800),random(600),50,50);
}
```

- background 関数を draw 内に表記するとどうなるか？

```
【ramdom】※打ち消し線部をコメントアウト、下線部を追加
function setup() {
  createCanvas(800, 600);
  background(255);
}
function draw() {
  background(255);
  strokeWeight(5);
  fill(random(255),random(255),random(255))
  ellipse(random(800),random(600),50,50);
}
```

## 2　まとめ課題：「ランダム関数を利用した制作」

- ファイル名：matome
- ランダム関数を用いた制作を行ってください。かたち・色など、これまでに得た知識、新しく調べたもの、様々に利用してください
- 1～3 行目のコメント行に、出席番号・名前・タイトルを入力してください
- キャンバスの大きさは、(800,800)とします

# VI　マウスを使ったインタラクション：システム変数を利用した描画

使うプロジェクト：system_variable

## 1　変数とは？

- コンピュータに一時的に保存しておく箱のようなモノ
- システム変数（p5.js が用意してくれているもの）と自分で作れる変数がある
- 本章では、システム関数を扱って描画をします

## 2　システム変数（System Variables）を使った描画

### 2-1　マウスの位置を取得するシステム変数

- 円の中心座標（位置）をマウスで変更できるようにしてみたい!

```
【mouse】※下線部を変更
function setup() {
  createCanvas(400, 400);
}
function draw() {
  background(31);
  noStroke();
  fill(63, 127, 255);
  circle(mouseX, 200, 20);
}
```

- p5.js では、マウスの現在の位置を取得する機能が搭載されている

| | |
|---|---|
| mouseX | マウスの x 座標の位置 |
| mouseY | マウスの y 座標の位置 |

- システム変数とは？
  - システムによって決められている変化する値を格納する場所

```
【mouse】※下線部を変更
function setup() {
  createCanvas(400, 400);
}
function draw() {
  background(31);
  noStroke();
  fill(63, 127, 255);
  circle(mouseX, mouseY, 20);
}
```

- お絵かきソフトを作ってみよう

```
【mouse】※下線部を追加、打ち消し線の部分をコメントアウト
function setup() {
  createCanvas(400, 400);
  background(31);
}
function draw() {
  ~~background(31);~~
  noStroke();
  fill(63, 127, 255);
  circle(mouseX, mouseY, 20);
}
```

画面の表示が変わる！
なぜこうなっているか考えてみよう

- さらに濃淡を調整

```
【mouse】※下線部を追加
function setup() {
  createCanvas(400, 400);
  background(31);
}
function draw() {
  noStroke();
  fill(63, 127, 255,50);
  circle(mouseX, mouseY, 20);
}
```

## 2-2　その他のシステム変数

- 代表的なものとしては、下記のものがある

| | |
|---|---|
| width | 画面(キャンバス) の幅 |
| height | 画面(キャンバス) の高さ |
| windowWidth | 現在開いているウィンドウの幅 |
| windowHeight | 現在開いているウィンドウの高さ |

【mouse】※下線部を変更

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(31);
}
function draw() {
  noStroke();
  fill(63, 127, 255,50);
  circle(mouseX, mouseY, 20);
}
```

## 2-3　システム変数を組み合わせた描画の工夫

- システム変数width を用いる

【mouse】※下線部を追加

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(31);
}
function draw() {
  noStroke();
  fill(63, 127, 255,50);
  circle(mouseX, mouseY, 20);
  fill(255, 127, 63, 50);
  circle(width-mouseX, mouseY, 20);
}
```

width-mouseX
何を意味しているだろう？

- システム変数 height も用いる

```
【mouse】※下線部を追加
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(31);
}
function draw() {
  noStroke();
  fill(63, 127, 255,50);
  circle(mouseX, mouseY, 20);
  fill(255, 127, 63, 50);
  circle(width-mouseX, mouseY, 20);
  fill(127, 63, 255, 100);
  circle(mouseX, height-mouseY, 10);
  fill(255, 63, 127, 100);
  circle(width-mouseX, height-mouseY, 10);
}
```

## 3　まとめ課題:「マウスの動きを用いた制作」

- マウスの位置を取得するシステム変数（mouseX, mouseY）を使用し、自由に制作をしてください。
  - 面白いものを考えてください！
  - 図形や位置、色、大きさなどを変化させたり、ランダム関数も使って OK です
  - それ以外の技術も活用して OK！
  - 1～3 行目のコメント行に、出席番号・名前・タイトルを入力してください
- 描画領域はフルスクリーンにします。
  - createCanvas(windowWidth,windowHeight);

# VII 変数を使った描画

使うプロジェクト：variable

## 1　変数とは？（再掲）

- コンピュータに一時的に保存しておく箱のようなモノ
- システム変数（p5.js が用意してくれているもの）と自分で作れる変数がある
- 本章では、自分で作れる変数を使って描画をします

## 2　変数を用いた描画

### 2-1　定義の仕方と代入

- 初期状態を実行してから、下記の通りに修正してみよう

```
【variable】※下線部を追加
let circleX;

function setup(){
  createCanvas(windowWidth, windowHeight);
  circleX=100;
}

function draw(){
  background(63);
  noStroke();
  fill(63, 127, 255);
  circle(circleX, 100, 40);
}
```

右辺の数値を変えて
実行してみよう！

- 何が起きているか？

```
let circleX;

function setup(){
  createCanvas(windowWidth, windowHeight);
  circleX=100;
}

function draw(){
  background(63);
  noStroke();
  fill(63, 127, 255);
  circle(circleX, 100, 40);
}
```

変数 circleX を定義
（→「circleX という箱を準備してください！」）

変数 circleX に、100 を代入
（→「変数 circleX の中身には、100 を入れてください！」）

円の中心の x 座標は、変数 circleX の値を使う

- 変数を使用するときに気をつけてほしいこと
  - 最初に宣言してから使う！

【宣言の仕方】
let 変数名;

  - 変数の名前の付け方にはルールがある
    - ✧ 半角英数字・アンダースコア(_)・ドル記号($)を使用可 ex)×circle？ ○circle_X
    - ✧ 先頭に数字は使えない ex)×1circle ○circle1
    - ✧ 予約語(プログラムであらかじめ使い勝手が決まっている語)は使えない
      ex)let function など

## 2-2　変数を使ってアニメーションを作る

- 円を動かすには，どうしたらよい？

```
【variable】※下線部を変更・追加
let circleX;

function setup(){
  createCanvas(windowWidth, windowHeight);
  circleX=0;
}

function draw(){
  background(63);
  noStroke();
  fill(63, 127, 255);
  circle(circleX, 100, 40);
  circleX=circleX+1;
}
```

circleX の中身を、
今ある circleX の値に 1 を足したものにする

→うまくいくと、円が横に動くはず！どうしてこうなったか考えてみよう！

- Y 座標も動かしてみる

```
【variable】※下線部を追加
let circleX;
let circleY;

function setup(){
  createCanvas(windowWidth, windowHeight);
  circleX=0;
  circleY=0;
}
function draw(){
  background(63);
  noStroke();
  fill(63, 127, 255);
  circle(circleX, circleY, 40);
  circleX=circleX+1;
  circleY=circleY+1;
}
```

実行させる前に、どんな動きになるか考えてみよう！

- 大きさも変えてみる

```
【variable】※下線部を追加
let circleX;
let circleY;
let radius;

function setup(){
  createCanvas(windowWidth, windowHeight);
  circleX=0;
  circleY=0;
  radius=1;
}
function draw(){
  background(63);
  noStroke();
  fill(63, 127, 255);
  circle(circleX, circleY, radius);
  circleX=circleX+1;
  circleY=circleY+1;
  radius=radius+1;
}
```

- 色も変えてみる

```
【variable】※下線部を追加・変更
let circleX;
let circleY;
let radius;
let red;

function setup(){
  createCanvas(windowWidth, windowHeight);
  circleX=0;
  circleY=0;
  radius=1;
  red=0;
}
```

```
function draw(){
  background(63);
  noStroke();
  fill(red, 0, 0);
  circle(circleX, circleY, radius);
  circleX=circleX+1;
  circleY=circleY+1;
  radius=radius+1;
  red=red+1;
}
```

- こうするとどうなるだろう？

```
【variable】※下線部を追加・打ち消し線をコメントアウト
let circleX;
let circleY;
let radius;
let red;

function setup(){
  createCanvas(windowWidth, windowHeight);
  circleX=0;
  circleY=0;
  radius=1;
  red=0;
  background(255);
}
function draw(){
  ~~background(63);~~
  noStroke();
  fill(red, 0, 0);
  circle(circleX, circleY, radius);
  circleX=circleX+1;
  circleY=circleY+1;
  radius=radius+1;
  red=red+1;
}
```

背景色の初期化(→ここでは、白色)

背景の再描画をしないと、どのように動いて見えるだろう？

## 3　まとめの制作

- ファイル名：matome
- 自分で変数を定義し、それを使って自由に制作をしてください。
  - 図形や位置、色、大きさなどを変化させたり、これまで学んだ知識を組み合わせても OK です！
  - 1～3 行目のコメント行に、出席番号・名前・タイトルを入力してください
- 描画領域はフルスクリーンにします。
  - createCanvas(windowWidth,windowHeight);

# VIII 文字と書式の表示

使うプロジェクト：text

## 1　文字の表示

### 1-1　text 関数

- text 関数を使うと、文字の描画ができる
- 書き方：text(<文字列>,<X 座標>,<Y 座標>);

```
【text】※下線部を追加
function setup(){
  createCanvas(600,425);
  background(255,255,255);
}
function draw(){
  text('The quick brown fox jumped over the lazy dog.', 30, 100);
}
```

- 文字内に「¥n」を入力すると、その部分で改行される

```
【text】※下線部を追加
function setup(){
  createCanvas(600,425);
   background(255,255,255);
}
function draw(){
  text('The quick brown fox¥njumped over the lazy dog.', 30, 100);
}
```

## 2　書式の設定

文字に対する書式の設定のタイミングは、図形の描画属性を設定する方法と同じ。文字の設定の前に、フォントサイズや色などの設定をすれば良いです。

### 2-1　サイズの設定：textSize 関数

- 書き方：textSize(＜文字サイズ＞);

```
【text】※下線部を追加
function setup(){
  createCanvas(600,425);
  background(255,255,255);
}
function draw(){
  textSize(22);
  text('The quick brown fox¥njumped over the lazy dog.', 30, 100);
}
```

### 2-2　色の設定：fill 関数

- 書き方：fill(＜R の値＞，＜G の値＞，＜B の値＞);

```
【text】※下線部を追加
function setup(){
  createCanvas(600,425);
  background(255,255,255)
}
function draw(){
  textSize(22);
  fill(65,117,122);
  text('The quick brown fox¥njumped over the lazy dog.', 30, 100);
}
```

## 2-3　行間の設定：textLeading 関数

- 書き方：textLeading(＜行間の値＞)

```
【text】 ※下線部を追加
function setup(){
  createCanvas(600,425);
  background(255,255,255)
}
function draw(){
  textSize(22);
  fill(65,117,122);
  textLeading(10);
  text('The quick brown fox¥njumped over the lazy dog.', 30, 100);
}
```

## 3　フォントの設定

- 書き方：textFont(＜フォント名＞);

```
【コードを書いてみよう】 ※下線部を追加
function setup(){
  createCanvas(600,425);
  background(255,255,255)
}
function draw(){
  textSize(22);
  fill(65,117,122);
  textLeading(10);
  textFont("Times New Roman");
  text('The quick brown fox¥njumped over the lazy dog.', 30, 100);
}
```

## 4　ここまでのまとめ：文字を使った制作

- ファイル名：matome
- 文字（何らかの意味のある文字列または無意味な綴りでも可能）を使って自由に制作をしてください
- テキスト内容やフォントや色を駆使して表現してください
- これまで使った知識（図形・ランダム・システム変数・自分で定義した変数）も自由に組み合わせてください
- 描画領域はフルスクリーンにします。
  - createCanvas(windowWidth,windowHeight);