

Graphics with Processing



2025-03 アニメーションと画像

<https://vilab.org>

塩澤秀和

3.1 システム変数と配列

グローバル変数

- `setup()`, `draw()`などの関数の外側で変数を定義すると...
 - すべての関数から参照できる
 - 関数を抜けても値が保持される

システム変数

- 自動設定されるグローバル変数
- `width`, `height`
 - ウィンドウのサイズ
- `mouseX`, `mouseY`
 - マウスのX座標とY座標
- `mousePressed`
 - 現在ボタンが押されているか?
 - 例: `if (mousePressed) {...`

配列の作成

- 初期値のある配列の作成
 - `int [] a = { 1, 2, 3, 4, 5 };`
⇒ `a[0]=1`から `a[4]=5`まで
- 空の配列の作成
 - `int [] a = new int [10];`
⇒ `a[0]~a[9]`を 0で初期化

配列の使用

- 配列の添字
 - 添字(番号)は 0~(要素数-1)
 - **【注意】** `new int [10]` で作成した配列に `a[10]` は存在しない!
- 配列の要素数
 - `a.length` で取得できる

3.2* アニメーション

アニメーション(p.202)

□ アニメーションの原理

- 静止画像の連続表示で動きを見せる映像(仮現運動)
- コマ撮り、パラパラマンガなど、静止画像の連続で動きを見せる

□ フレームレート=毎秒コマ数

- 映画 24fps、TV放送 30fps
- CG/ゲーム 30fps以上推奨(60fpsで視覚的にほぼ充分)

アニメーションプログラミング

□ Processingでは

- draw()の中で毎回少しずつ図形を移動・変形して描画する
- または、あらかじめ用意した静止画像を切り替えて表示する
- 基本的には、毎回backgroundで全部消して、新しく描画し直す
- 変化させる座標等は、グローバル変数で保持する必要がある



少しずつ図形を移動・変形して表示する

3.3* アニメーションの実現

関連関数

- `frameRate(回数)`
 - 毎秒の描画(draw)回数を設定
- `millis()`
 - プログラム開始からのミリ秒
- `noise(x)`
 - `x`にともなって連続的に変化するでたらめな値(0~1)を生成
 - 乱数のように不連続にならない

システム変数

- `frameCount`
 - `draw()`が呼ばれた回数
- `frameRate`
 - 実際の現在の毎秒コマ数

```
float a; // 角度
```

変化させる値は
グローバル変数

```
void setup() {  
  size(600, 600);  
  frameRate(30); // 30fpsに設定  
}
```

```
void draw() {  
  //background(255);
```

```
  // 毎フレーム、座標等を計算
```

```
  float x = 200 * cos(3*a) + 300;
```

```
  float y = 200 * sin(2*a) + 300;
```

```
  fill(255 * noise(a), 120, 150);
```

```
  circle(x, y, 100);
```

滑らかな色の変化

```
  // 少しずつ動かす(1度= $\pi/180$ rad)
```

```
  a += 1 * PI/180;
```

```
}
```

3.4 画像データの表示

画像データ(ラスター画像)

- 画像ファイルの利用
 - サンプル→Basics→Image→LoadDisplayImage など
 - 対応形式: jpg gif png tga
- PImage型
 - 画像を扱うには、PImage型のグローバル変数を用意しておく
例) PImage img;
- loadImage("ファイル名")
 - 画像データの読み込み
 - 通常、setup()で1回だけ行う
img = loadImage("a.jpg");
 - ファイルは、事前にメニューの[スケッチ]→[ファイルを追加]でdataフォルダにコピーしておく

画像表示

- image(画像, x, y)
 - 画像の描画
- image(画像, x, y, 幅, 高さ)
 - サイズを変更して画像を描画
- imageMode(モード)
 - rectMode/ellipseModeと同様

画像の部分表示

- copy(画像, x_{画像}, y_{画像}, w_{画像}, h_{画像}, x, y, w, h)
 - 画像内の指定領域だけを描画
- blend(画像, x_{画像}, y_{画像}, w_{画像}, h_{画像}, x, y, w, h, 混色演算)
 - 画像を指定した方法で重ね塗り

3.5 画像によるアニメーション

```
// キャラクタ画像をdata内に用意
// https://vilab.org/cg2025/
// skel0123.zipを展開し、各画像を
// [スケッチ]→[ファイルを追加]
```

```
PImage [] sprites
    = new PImage[4]; // 画像4枚
int dots = 128;
```

```
void setup() {
    size(400, 400);
    frameRate(30);
```

```
    noSmooth(); // ドット感を維持
    imageMode(CENTER);
```

```
// 画像ファイル名を生成して読み込む
for (int i = 0; i < 4; i++) {
    String fn = "skel" + i + ".png";
    sprites[i] = loadImage(fn);
}
}
```

```
void draw() {
    background(128, 0, 0);
```

```
// 4枚の画像を6フレームごとに変える
int f = (frameCount / 6) % 4;
```

```
    image(sprites[f],
        mouseX, mouseY, dots, dots);
}
```

3.6 オブジェクト指向基礎

オブジェクト指向

□ オブジェクトとは

- データとその関連操作をまとめて、使いやすくしたもの
- 例) PImage img

オブジェクト指向用語

□ 「クラス」:オブジェクトの型

- 例) PImage

□ 「インスタンス」:オブジェクト変数

- 例) img

□ 「フィールド」:オブジェクトの属性

- 例) img.height

□ 「メソッド」:オブジェクトの操作

- 例) img.resize(64, 64)

PImage型の例

□ フィールド

- img.width, img.height
 - 画像のサイズ(横・縦の幅)
- img.pixels[]
 - 画像データのピクセル配列(次回)

□ メソッド(一部)

- img.save("ファイル名")
 - 画像にファイル名をつけて保存
- img.get(x, y, 幅, 高さ)
 - 画像の一部を画像として取り出す
- img.resize(幅, 高さ)
 - 画像のサイズを変更する
- img.loadPixels(), img.updatePixels()
 - ピクセル処理のためのメソッド

3.7 図形データの表示

図形データ

- 画像の形式(p.16)
 - ラスター画像: ピクセル(ドット)の集合として画像を表現
 - ⇒ 高速に処理できる
 - ベクター画像: 座標値と数式で決まる図形で画像を表現
 - ⇒ 拡大変形しても滑らか
 - 図形(ベクター画像)の利用
 - サンプル→Basics→Shape→LoadDisplaySVG など
 - 対応形式: SVG
 - Inkscape等で作成できる
 - PShape型
 - SVG図形を扱うための型
- PShape shape;

図形表示

- loadShape("ファイル名")
 - SVGデータの読み込み
 - 通常、setup()で1回だけ行う

```
sh = loadShape("a.svg")
```

- ファイルは、事前にメニューの[スケッチ]→[ファイルを追加]でdataフォルダにコピーしておく
- shape(図形, x, y)
- shape(図形, x, y, 幅, 高さ)
 - 図形の描画
- shapeMode(モード)
 - imageModeと同様
- その他の操作
 - PShapeのメソッドで拡大、回転、図形の合成などの編集ができる

3.8 演習課題

課題

- 右のプログラムを理解してから、位置や速度の変数を**配列**に変更して20個以上の図形が動き回るアニメーションを作成しなさい
 - 例) float [] px = new ...
 - forで配列を扱う方法を再確認!
 - 図形・動き方・色などは変更してもよい(画像を使ってもよい)
 - 図形同士の反射には、物理学の知識が必要なので注意!
 - 【注意】色を乱数で**チカチカさせない**こと(noiseを使ってみよ)
- 提出について
 - 画像を含む場合はZIP化する
 - [ツール]→[スケッチをアーカイブ]

```
float px, py; // 図形の位置
float vx, vy; // 図形の数度
float r = 20;
```

```
void setup() {
  size(800, 600);
  px = random(r, width - r);
  py = random(r, height - r);
  vx = random(-5.0, 5.0);
  vy = random(-5.0, 5.0);
}
```

```
void draw() {
  background(0, 200, 255);

  px += vx;
  if (px < r || px > width - r)
    vx *= -1;
  py += vy;
  if (py < r || py > height - r)
    vy *= -1;

  rectMode(CENTER);
  rect(px, py, 2*r, 2*r);
}
```