

Graphics with Processing



2020-03 アニメーションと画像

<http://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`
 - 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

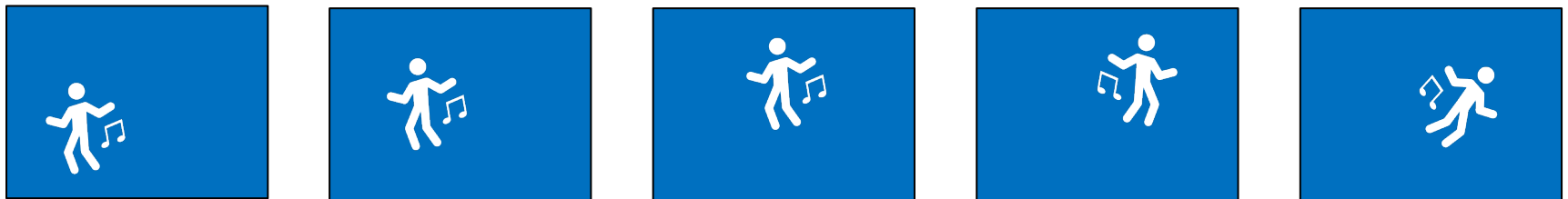
3.2* アニメーション

アニメーション(p.202)

- アニメーションの原理
 - 人間の視覚の「仮現運動」(残像効果)を利用した映像
 - コマ撮り,パラパラマンガなど,静止画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(視覚的には60fpsで十分)

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`の変化にともなって、滑らかに変化する値(0~1)を生成する
 - 色の変化などに使うと面白い

システム変数

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

```
// 変化する値はグローバル変数にする
float a; // 角度
```

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

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

```
  float x = 100 * cos(a) + 200;
  float y = 100 * sin(a) + 200;
  // fill(200 * noise(a/2) + 55);
  circle(x, y, 50); // 円の描画
```

```
  // 2度ずつ動く(1度=π/180ラジアン)
  a += 2 * 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
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 Sketch → Add File...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)
 - 指定した方法で画像を重ね塗り

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
// http://vilab.org/cg2020/
// 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];
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);

    // 6フレームごとに画像を変える
    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
 - データとその操作をセットにして、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 Sketch → Add File...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

- 下記サンプルを参考に、4つ以上のボールがはね回る(動き回る)プログラムを作成しなさい
 - [サンプル]→[Topics]→[Motion]→[Bounce]
 - **条件1:** ボールや画像の位置や方向は配列に格納する
 - **条件2:** それらの最初の配置(座標等)は乱数で決める
 - 発展例: 画像を使う(背景色を透過にするとよい) / 速度, 色, 大きさ等を個別に設定する
- 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]

ヒント

- 条件1の例(グローバル変数で)


```
float [] xpos = new float[4];
float [] ypos = new float[4];
int [] xdirection = { 1, 1, 1, 1 };
int [] ydirection = { 1, 1, 1, 1 };
```
- 条件2の例(setup()の中で)

(補足: RADIUSは中心&半径モード)

```
randomSeed(millis());
for (int i = 0; i < 4; i++) {
    xpos[i] = random(0, width);
    ypos[i] = random(0, height);
}
```
- ボールの描画(draw()の中で)


```
for (int i = 0; i < 4; i++) {
    ellipse(xpos[i], ypos[i], rad, rad);
}
```