

Graphics with Processing



2014-03 アニメーションと画像

<http://vilab.org>

塩澤秀和

3.1 アニメーション

アニメーション(p.162)

- パラパラマンガのように
 - draw()の中で毎回形や位置をずらして描画する
 - 各図形の形や位置(座標等)はグローバル変数で保持する
- 例
 - サンプルのSetupDrawより

```
int y = 100; // 図形の座標位置
...
void draw() {
  background(0); // 毎回消去
  ...
  y = y - 1; // 毎回位置変更
  ...
  line(0, y, width, y); // 描画
}
```

関連関数

- frameRate(回数)
 - 毎秒の描画(draw)回数を設定
 - 停止・再開: noLoop(), loop()
- redraw()
 - 強制的に再描画させる(おもにアニメーションでないときに使う)
- millis()
 - プログラム開始からのミリ秒

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

3.2 配列とシステム変数

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

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) {...`

3.3 自作関数と組み込み関数

自作関数(メソッド)

- JavaやCと同様

```

戻り値の型 関数名(引数並び) {
    処理手順
    ...
    return 戻り値;
}

```

数学関数

- sqrt(値)
 - 平方根($\sqrt{\quad}$)
- pow(x, y)
 - xのy乗
- dist(x1, y1, x2, y2)
 - 2点間の距離
- constrain(式, 最小, 最大)
 - 式の値を範囲内に収める

三角関数

- sin(角度), cos(角度), ...
- atan2(x, y)
 - x軸とベクトル(x, y)の成す角
- radians(deg), degrees(rad)
 - 度 \leftrightarrow ラジアンの変換関数

時刻関数

- year(), month(), day()
- hour(), minute(), second()

乱数関数

- randomSeed(種)
 - 乱数の準備
 - 種は関数 millis() などを使う
- random(最小値, 最大値)
 - 乱数の発生(float型)

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...でデータ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 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

3.6 画像によるアニメーション

```
// キャラクタの画像をdataフォルダに用意  
// hone0.png, hone1.png, hone2.png
```

```
PImage [] sprites = new PImage[3];
```

```
int x, y;
```

```
int dots = 128;
```

```
String name = "hone";
```

```
void setup() {
```

```
  size(400, 400);
```

```
  frameRate(30);
```

```
  for (int i = 0; i < 3; i++) {
```

```
    sprites[i] =
```

```
      loadImage(name + i + ".png");
```

```
  }
```

```
  imageMode(CENTER);
```

```
  randomSeed(millis());
```

```
  x = (int) random(0, width);
```

```
  y = (int) random(0, height);
```

```
}
```

```
void draw() {
```

```
  background(128, 0, 0);
```

```
  int f = (frameCount / 6) % 4;
```

```
  if (f == 3) f = 1;
```

```
  if (x < mouseX) x++;
```

```
  else if (x > mouseX) x--;
```

```
  if (y < mouseY) y++;
```

```
  else if (y > mouseY) y--;
```

```
  noSmooth();
```

```
  image(sprites[f], x, y, dots, dots);
```

```
}
```

3.7 図形データの表示

図形データ

- 画像の形式
 - 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 → LoadDisplayShapeSVG 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...でデータフォルダにコピーしておく
- shpae(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

- サンプルBounceを参考にして、4つ(以上)のボールがはね返るアニメーションを作成しなさい
 - サンプル Examples → Topics → Motion → Bounce
 - **条件1**: ボールの座標や方向は配列に格納すること
 - **条件2**: ボールの最初の座標は乱数で決めること
 - (条件ではないが) 各ボールの大きさや速さを変えると面白い
- 提出
 - ×切: 次回講義開始時
 - 提出ページ <http://www2.vilab.org/upload/cg-upload.html>

ヒント

- 条件1(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(setup()の中で)

```
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]+size/2,
            ypos[i]+size/2, size, size);
}
```