

Graphics with Processing



2020-02 基本図形と曲線

<http://vilab.org>

塩澤秀和

2.1 変数と制御構造 (Javaと同じ)

データ型

- int, float, byte
 - 数値
 - 実数はfloatが標準
- boolean
 - 真偽値(Yes/No)
 - 定数: true(真), false(偽)
- char
 - 文字(漢字も可)
 - char ch = 'あ'
- String
 - 文字列
 - String str = "あいうえお"
- 文字列の連結
 - +演算子で文字列を連結できる

制御構造

- if-else
- switch-case
 - 条件分岐
- for, while
 - ループ(繰り返し)
 - do-whileは(一応)ない
- break
 - ループ中断

演算子

- 関係演算子
 - == != < > <= >=
- 論理演算子
 - || (OR) && (AND) ! (NOT) 2

2.2 基本図形

図形描画関数

- point, line, rect
 - 点, 直線, 長方形 (既出)
- triangle(x1, y1, x2, y2, x3, y3)
 - 3点を結ぶ三角形
- quad(x1, y1, x2, y2, x3, y3, x4, y4)
 - 4点を結ぶ四角形
- ellipse(x, y, 幅, 高さ)
 - 楕円(円)
- arc(x, y, 幅, 高さ, 開始角, 終了角)
 - 弧 (角度はラジアン)
 - π として定数PIが使える

描画色

- stroke(色)
 - 線(境界線)の色を設定
 - noStroke()で境界線なし
- strokeWeight(太さ)
 - 線の太さを設定
- fill(色)
 - 塗りつぶしの色を設定
 - noFill()で塗りつぶしなし

座標指定モード

- rectMode(モード)
- ellipseMode(モード)
 - 左上を指定: CORNER
 - 中心を指定: CENTER

2.3 曲線の表現形式

曲線の数式表現 (p.72)

□ 陽関数形式

- $y = f(x)$ 型

- 例
$$y = \sqrt{r^2 - x^2}$$

□ 陰関数形式

- $f(x, y) = 0$ 型

- 例
$$x^2 + y^2 - r^2 = 0$$

□ パラメータ形式

- $x = f(t), y = g(t)$ 型

- パラメータ=媒介変数

- 例
$$\begin{cases} x = r \cos(t) \\ y = r \sin(t) \end{cases}$$

パラメトリック曲線 (p.76)

□ パラメータ形式による曲線

- 少ない変数で滑らかな曲線
- 曲線を点列に分解するのが容易

□ 点間を補間する曲線

- Ferguson曲線
- Catmull-Rom曲線

□ 制御点(アンカー点)による曲線

- Bezier曲線
- Bスプライン曲線
- CGモデリングで広く用いられる

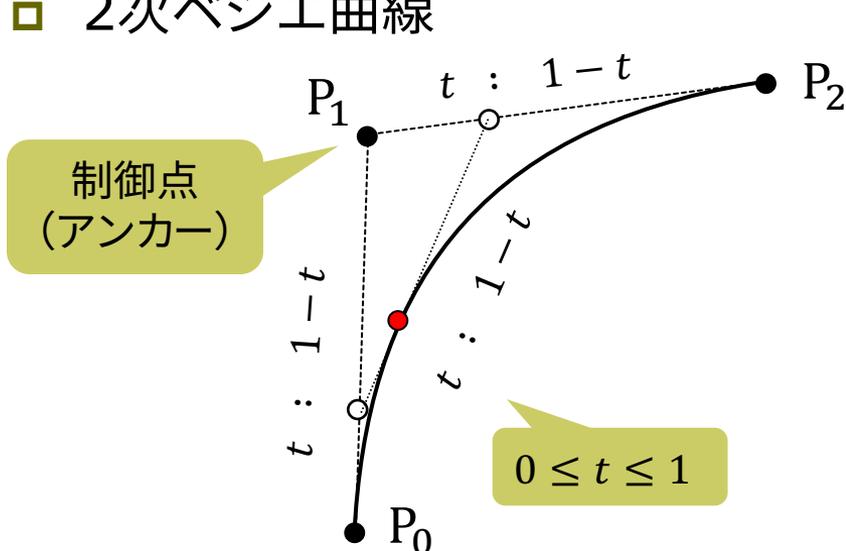
□ 重み付き制御点による曲線

- 有理Bezier曲線
- NURBS曲線(Non-Uniform Rational B-Spline)

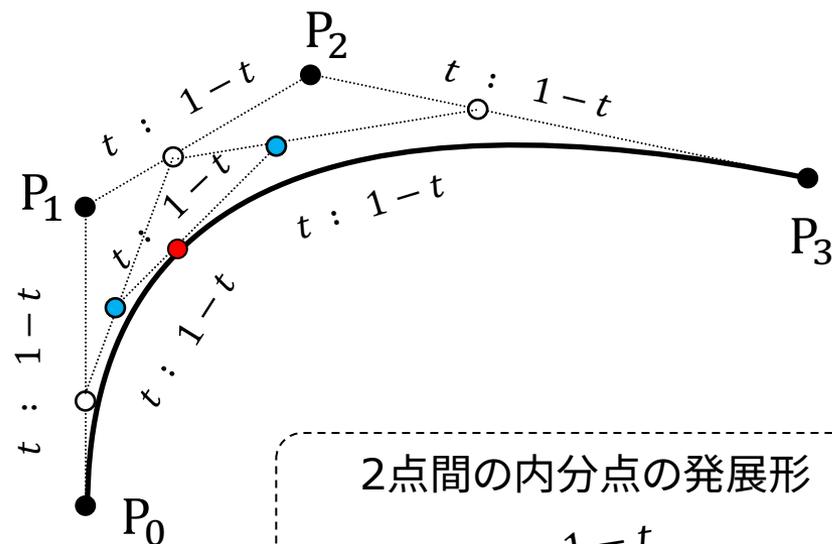
2.4* 制御点による曲線

ベジエ曲線 (p.77)

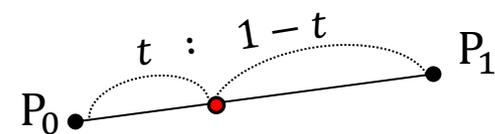
□ 2次ベジエ曲線



□ 3次ベジエ曲線



2点間の内分点の発展形



$$P = (1-t)P_0 + tP_1$$

$t = 0.5$ なら中点

□ 制御点による曲線の典型的な形

$$P(t) = b_0(t)P_0 + b_1(t)P_1 + \dots + b_n(t)P_n$$

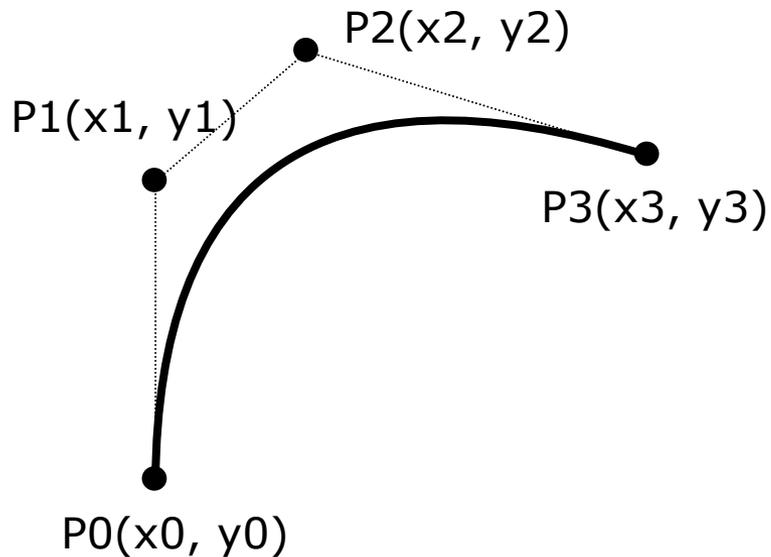
$$b_0(t) + b_1(t) + \dots + b_n(t) = 1 \quad (0 \leq t \leq 1)$$

こうすると、 b の値によって各点に近づく ($b_0 = 1$ のとき P_0 , $b_n = 1$ のとき P_n)

2.5* Bezier曲線

ベジエ曲線の描画

- $\text{bezier}(x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3)$



- 単純な数式で自然な曲線
- CGでは通常3次(以上)が利用される

ベジエ曲線の数式表現

- 2次ベジエ曲線(3点)

$$P(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = (1-t)^2 \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + 2t(1-t) \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + t^2 \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$$

- 3次ベジエ曲線(4点)

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

各係数は、 $(1-t) + t$ を n 乗したときの形

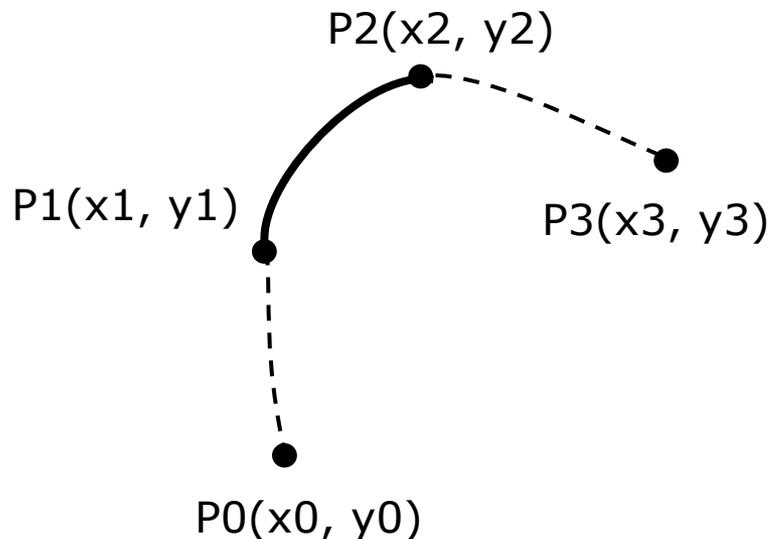
サンプルプログラム

- File → Examples
 - → Basics → Form → Bezier

2.6 補間曲線

補完曲線の描画

- $\text{curve}(x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3)$



- 4点を滑らかに補間し,真ん中の2点を結ぶ曲線を描く
- Catmull-Romスプライン曲線

Catmull-Rom曲線の数式表現

- 3次のスプライン曲線

$$P(t) = \frac{1}{2} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

- 各座標の計算

$$x(t) = \frac{1}{2} \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= \frac{1}{2} \left[(-x_0 + 3x_1 - 3x_2 + x_3)t^3 + (2x_0 - 5x_1 + 4x_2 - x_3)t^2 + (-x_0 + x_2)t + 2x_1 \right]$$

2.7 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(y, x)
 - 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

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

2.8 演習課題

課題

- 基本図形を繰り返し描画することによって、複雑な幾何学模様を描くプログラムを作成せよ
 - 条件: プログラム全体のなかで、1カ所はbezierを使うこと
 - for文で図形を繰り返すとよい (2重ループを使うのも面白い)
 - 座標の値を乱数(random)で生成するのも面白い

- 注意
 - 学年・クラス・出席番号・氏名をプログラムの最初に書くこと
 - 例: // 2-3-10 鈴木太郎
 - 面白い作品は授業中に紹介してボーナス点!

参考プログラム

```
void setup() {
  size(600, 400);
  // 基本的にはアニメーション不要
  noLoop();
}

void draw() {
  background(240, 240, 255);
  // 図形を繰り返して模様を描く
  for (float a = 0; a < 2 * PI;
       a += PI / 64) {
    float x = 300 * cos(a) + 300;
    float y = 200 * sin(a) + 200;
    stroke(0, 0, 150);
    line(300, 200, x, y);
  }
}
```