Graphics with Processing

2024-02 基本図形と曲線

https://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
  - Javaに符号なし整数はない
- boolean
  - 真偽値（Yes/No）
  - 定数：true（真）, false（偽）
- char
  - 文字（UTF-16）
  - char ch = 'あ'
- String
  - 文字列
  - String str = "あいうえお"
  - 文字列は+演算子で連結できる

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 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)
- quad(x1, y1, x2, y2, x3, y3, x4, y4)
  - 三角形と四角形
- circle(x, y, 直径)
- ellipse(x, y, 幅, 高さ)
  - 円と楕円(半径モードも可能)
- arc(x, y, 幅, 高さ, 開始角, 終了角)
  - 弧(角度の単位はラジアン)
  - 円周率として定数PIが使える

描画色

- stroke(), strokeWeight()
  - 線の色と太さ(既出)
- noStroke()
  - 境界線を描画しない
- fill(色)
  - 塗りつぶしの色を設定
  - noFill()で塗りつぶしなし

図形の基準位置

- rectMode(モード)
- ellipseMode(モード)
  - 左上座標で指定: CORNER
  - 中心座標で指定: CENTER
  - 中心座標と半径: RADIUS

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
  - 並列処理や衝突判定には向く
  - 例) $x^2 + y^2 - r^2 = 0$
- パラメータ(媒介変数)形式
  - $x = f(t),\ y = g(t)$ 型
  - 変数tの変化による軌跡
  - 例) $\begin{cases} x = r\cos(t) \\ y = r\sin(t) \end{cases}$

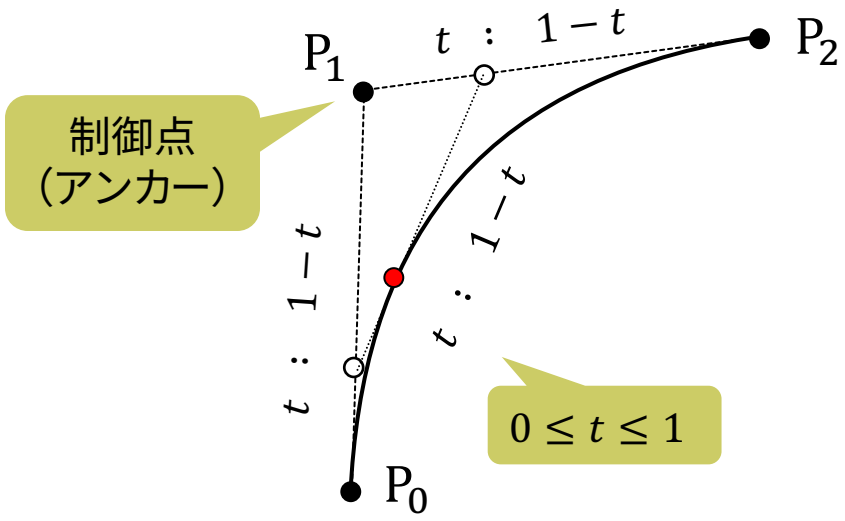
パラメトリック曲線(p.76)

- パラメータ形式による曲線
  - 滑らかで複雑な曲線を描ける
  - 曲線を点列に分解して扱いやすい
- 通過点による曲線(補間曲線)
  - 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)

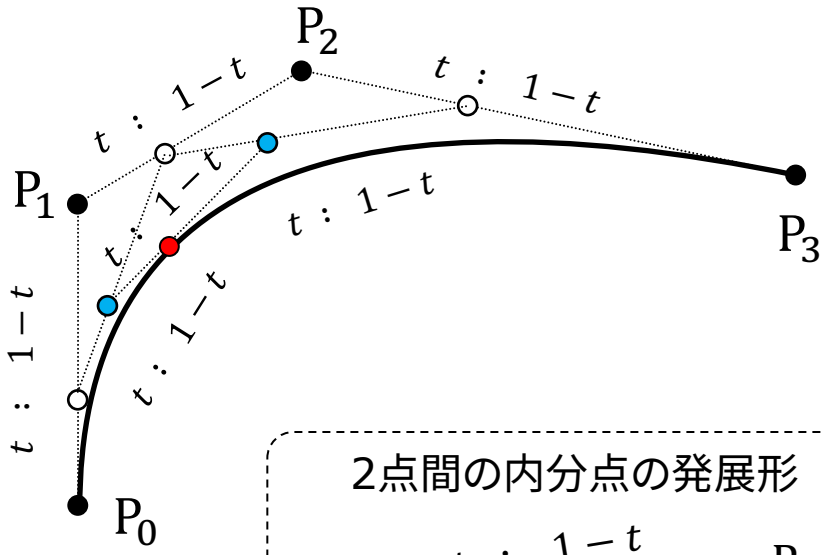
2.4* 制御点による曲線

ベジエ曲線(p.77)

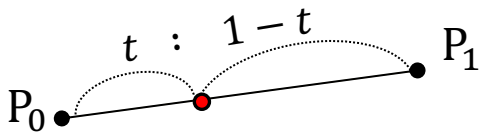
- 2次ベジエ曲線



- 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基本形

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

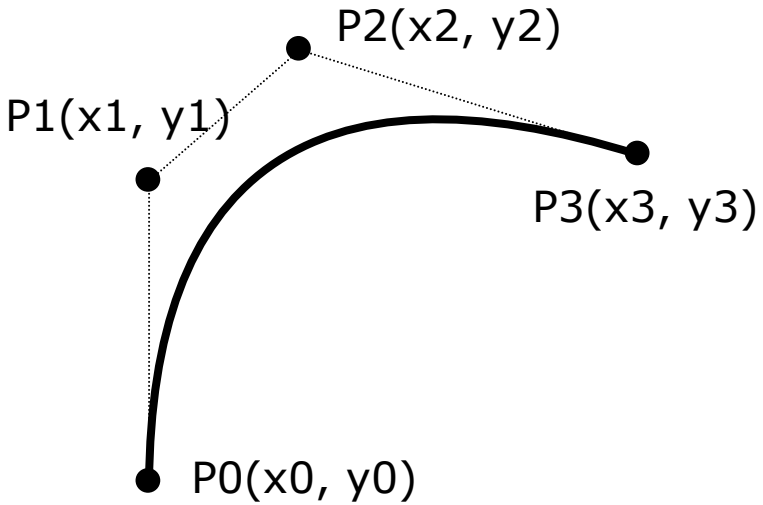
$$b_0(t) + b_1(t) + \cdots + b_n(t) = 1 \quad (0 \le t \le 1)$$

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(3次)の描画

- bezier(x0, y0, x1, y1, x2, y2, x3, y3)



- 単純な数式で自然な曲線
- [サンプル]→[Basics]→[Form]→[Bezier]
- noFill()で塗りつぶしなし
- 色々な形を描いてみよう

ベジエ曲線の数式表現

- 2次ベジエ曲線(3点)
  - 内分点の式を3回使って求める

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

- 3次ベジエ曲線(4点)
  - CGでは3次の曲線が一般的

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

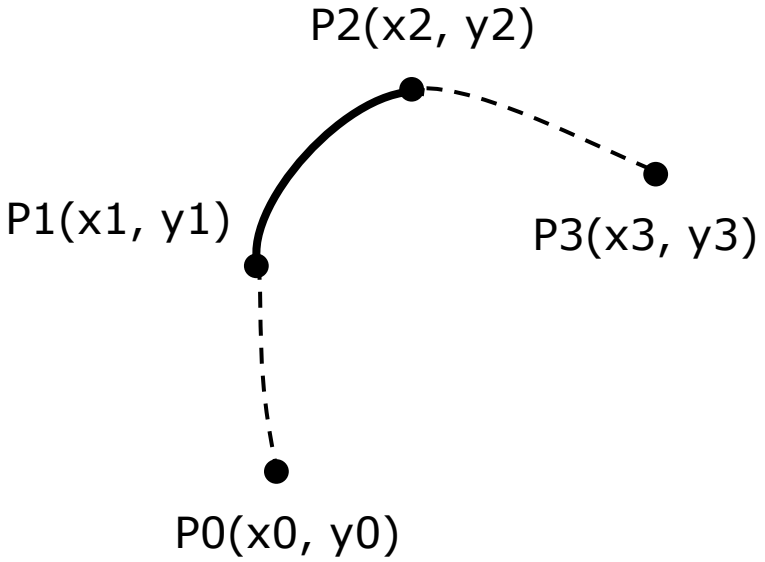
- n次ベジエ曲線(n+1点)
  - 係数 $b_k(t)$ は $(1-t)+t$ をn乗したときの各項になる
  - バーンスタイン基底関数という

$$b_k(t) = {}_n\mathrm{C}_k t^k (1-t)^{n-k}$$

2.6 補間曲線

補間曲線の描画

- curve(x0, y0, x1, y1,
  x2, y2, x3, y3)
  - 4点を滑らかに補間し、真ん中の2点を結ぶ曲線を描く
  - 座標を共有すると、前後の曲線が滑らかにつながる
  - Catmull-Romスプライン曲線



- Catmull-Rom曲線の数式
  - 3次のスプライン曲線 （$0 \le t \le 1$ で描画）

$$P(t) = \frac{1}{2}[(-t^3 + 2t^2 - t)P_0 + (3t^3 - 5t + 2)P_1 + (-3t^3 + 4t^2 + t)P_2 + (t^3 - t^2)P_3]$$

$$= \frac{1}{2}[(-P_0 + 3P_1 - 3P_2 + P_3)t^3 + (2P_0 - 5P_1 + 4P_2 - P_3)t^2 + (-P_0 + P_2)t + 2P_1]$$

t に値を代入すると $P(-1) = P_0,\ P(0) = P_1,\ P(1) = P_2,\ P(2) = P_3$ となる

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, ...) {
    処理手順
    ...
    return　戻り値;
}
```

数学関数

- sqrt(値)
  - 平方根 $\sqrt{x}$
- 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 ⇔ 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

- Bezier曲線を含む図形をfor文で繰り返し描画し、きれいな模様を描くプログラムを作成せよ
  - 条件：1つはfor文があり、その中に1つはbezierが含まれる
  - 工夫の例：座標や色を乱数や三角関数で生成する / 変化量を角度にして放射状の形を描く
- 提出について
  - [ファイル]メニュー→[設定]で日本語のフォントも選べる
  - プログラム冒頭にも氏名を記入
  - 画像は必要な部分だけ提出
  - 毎回、よくできた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);
  noLoop();  // アニメーションは不要
  randomSeed(millis());
}

void draw() {
  background(240, 240, 255);
  for (int x = 0; x < 600; x += 50) {
    for (int y = 0; y < 400; y += 50) {
      fill(random(255), random(255),
           random(255));
      noStroke();
      // forの中に1つはbezierを入れる
      quad(x, y, x + 50, y, x + 50,
           y + 25, x, y + 50);
    }
  }
}
```