

# Graphics with Processing

2019-05 複雑な図形の描画

<http://vilab.org>

塩澤秀和

# 5.1 頂点列による図形描画

## 複雑な図形描画

### □ beginShape(図形)

- 頂点列モードの開始
- 「図形」を省略: 全頂点を順に線でつなぐ(折れ線或多角形)
- 「図形」を指定(以下の定数):  
POINTS, LINES,  
TRIANGLES,  
TRIANGLE\_FAN,  
TRIANGLE\_STRIP,  
QUADS, QUAD\_STRIP
- 頂点を順に組みにして、図形を複数連続的に描画する

### □ endShape()

- 頂点列モードの終了
- endShape(CLOSE): 起点と終点を線で結んで閉じる

## 頂点の追加

### □ vertex(x, y)

- 図形に新しい頂点を追加する

### □ curveVertex(x, y)

- 曲線を描く頂点を追加する

### □ bezierVertex(x1, y1, x2, y2, x3, y3)

- ベジエ曲線をつなげて追加する

## 描画設定

### □ strokeJoin(モード)

- 頂点での接続形状を指定できる
- MITER, BEVEL, ROUND

### □ 頂点ごとの着色

- P2DまたはP3Dモードで可能
- 最初に size(幅, 高さ, P2D);

## 5.2 多角形の描画例

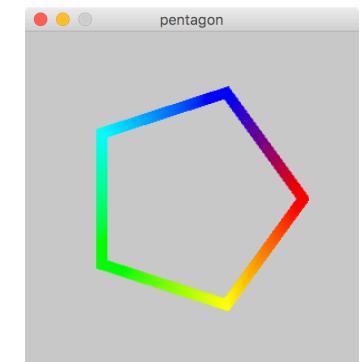
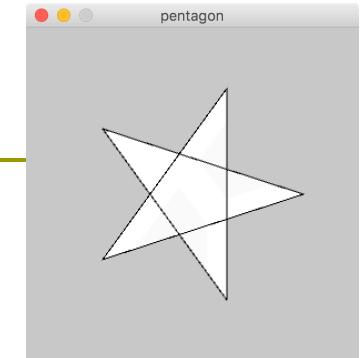
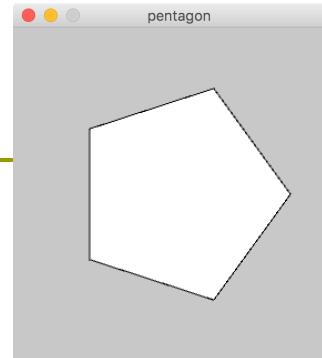
```
final float a = 2 * PI / 5;
```

```
final float r = 200;
```

```
void setup() {
    size(600, 600, P2D); noLoop();
}
```

P2Dモードを指定

```
void draw() {
    float x0 = width/2, y0 = height/2;
    beginShape();
    vertex(x0 + r * cos(0*a), y0 + r * sin(0*a));
    vertex(x0 + r * cos(1*a), y0 + r * sin(1*a));
    vertex(x0 + r * cos(2*a), y0 + r * sin(2*a));
    vertex(x0 + r * cos(3*a), y0 + r * sin(3*a));
    vertex(x0 + r * cos(4*a), y0 + r * sin(4*a));
    endShape(CLOSE);
}
```



- いろいろ改造してみよう
1. 星型にする
  2. 頂点ごとに色をつける
  3. forループを使う
  4. 回転させる
  5. 正六角形やそれ以上

# 5.3 画像の変形

## 2Dテクスチャマッピング

### □ texture(画像)

- 画像: PImage型(第3回参照)
- 図形に“貼り付ける”画像の設定
- beginShape()～endShape() のなかで指定する
- P2Dモードで利用可能

### □ vertex(x, y, u, v)

- 図形に頂点(x, y)を追加し,その頂点に画像内の座標(u,v)を対応づけて貼り付ける
- (u, v)は画像の端でなくてもよい

### □ textureMode(モード)

- uv座標の指定モード
- NORMAL: 0.0～1.0(正規化)
- IMAGE: 画像内のピクセル座標

```
PImage pic;
```

```
void setup() {
```

```
size(800, 800, P2D);
```

```
pic = loadImage("sharaku.jpg");
```

```
}
```

```
void draw() {
```

```
background(0);
```

```
int w = pic.width;
```

```
int h = pic.height;
```

```
// noStroke();
```

```
beginShape(QUAD_STRIP);
```

```
texture(pic);
```

```
for (float f = 0; f < 1.1; f += 0.1) {
```

```
float x = mouseX * sin(f * PI);
```

```
float y = f * h;
```

```
vertex(x, y, 0, y);
```

```
vertex(x + w, y, w, y);
```

```
}
```

```
endShape();
```

```
}
```

P2Dモード  
で利用可能

# 5.4 図形の回転・拡大(予習)

基本的な書きかた

```
pushMatrix();
translate(x, y);
rotate(a);
/* (x,y)からの相対位置で描画 */
popMatrix();
```

簡単な意味

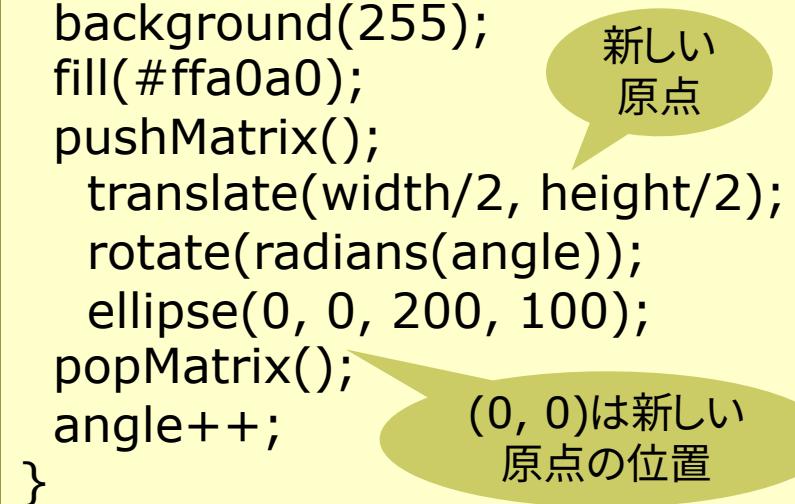
- pushMatrix()～popMatrix()
  - 座標の変更部分を囲む
- translate(x, y)
  - 座標原点(回転・拡大の中心)を(x, y)に移動する
- rotate(a)
  - 原点を中心に,aラジアン回転
- scale(s), scale(sx, sy)
  - 原点を中心に,拡大または縮小

図形の回転の例

```
int angle = 0;

void setup() {
  size(400, 400);
  rectMode(CENTER);
}

void draw() {
  background(255);
  fill(#ffa0a0);
  pushMatrix();
  translate(width/2, height/2);
  rotate(radians(angle));
  ellipse(0, 0, 200, 100);
  popMatrix();
  angle++;
}
```



# 5.5 タイポグラフィ(文字表示)

---

```
// 描画用フォントの変数(PFont型)
PFont font1, font2;

void setup() {
    size(300, 300);

    // Processing用のフォントを作る方法
    // (Tools → Create Font... でファイルを
    // 作っておけば,どんな環境でも利用可能)
    // ※ 日本語の場合ファイルが大きくなる
    font1 = loadFont("Impact-48.vlw");

    // JavaまたはOSのフォントを使う方法
    // ※ 日本語の場合はこちらがおすすめ
    font2 = createFont("メイリオ", 48);

    // 座標指定モード(通常はMODEL)
    textMode(MODEL);
}
```

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

    // 文字同士のxy方向の位置あわせ
    textAlign(CENTER, BOTTOM);

    pushMatrix();
    translate(width/2, height/2);
    rotate(radians(frameCount));

    fill(128, 0, 0); // 文字の色
    textSize(32); // フォントとサイズ
    text("Processing", 0, 0); // 文字列と座標

    fill(0, 0, 128);
    textSize(48);
    text("角度 " + frameCount, 0, 100);
    popMatrix();
}
```

# 5.6 対話的入力処理

---

## システム変数

- mouseX, mouseY
- mousePressed
  - 既出
- pmouseX, pmouseY
  - 前フレームでのマウス位置
- mouseButton
  - 押されたマウスボタン
  - LEFT, RIGHT, CENTER
- keyPressed
  - キーが押されていればtrue
- key
  - 押された文字
- keyCode
  - 特殊キーのキーコード

## コールバック関数

- void mousePressed()
  - この関数があると、マウスボタンが押されたときに自動的に実行
- void mouseReleased()
  - 同様に、ボタンが離されたとき
- void mouseMoved()
  - マウスが動かされたとき(ただし、ボタンは押されていないとき)
- void mouseDragged()
  - マウスがドラッグされたとき
- void keyPressed()
  - キーが押されたとき
- void keyReleased()
  - キーが離されたとき

# 5.7 演習課題

## 課題

- マウスでクリックした座標の点を順に結ぶ“折れ線”(or 多角形)を描くプログラムを作成しなさい
  - **条件1:** beginShape, vertex, endShapeの組合せを使う
  - **条件2:** 線同士の接続形状は、とがった形(MITER)にする
  - **条件3:** マウスの入力判定には、mousePressed()関数を使う
  - **ヒント:** 5.2をforループにまとめたものを参考にするとよい
  - **発展例:** 頂点ごとに色をつける、ファイルから線画を読み込む、描いた図形を回転したり動かす
  
- 見やすいプログラムを提出せよ
  - Edit→Auto Format も試して

```

int npos = 0; // 点の数
int [] x = new int[100];
int [] y = new int[100];

void setup() {
  size(800, 600);
  noLoop();
}

void draw() {
  background(255);
  strokeWeight(8);

  for (int i = 0; i < npos; i++) {
    point(x[i], y[i]);
  }
}

void mousePressed() {
  x[npos] = mouseX;
  y[npos] = mouseY;
  npos++;
  redraw(); // 点が増えたら再描画
}

```

この行は消さない

ループの中はvertex 1つに

# 5.8 参考：ファイル入出力

## 簡易ファイル入出力

- **loadStrings("ファイル")**
  - ファイルから1行ごとに文字列として読み込み、配列をつくる
  - 画像と同様、ファイルは事前にデータフォルダにコピーしておく (Sketch → Add File...)

```
String [] lines =
loadStrings("data.txt");
for (int i = 0; i < lines.length;
     i++) {
    // lines[i]の処理
}
```

- **saveStrings("ファイル", 配列)**
  - ファイルに文字列を保存する
  - loadStringsの逆の処理

## 文字列処理

- **float(文字列), int(文字列)**
  - 文字列を数値に変換
- **str(数値)**
  - 数値を文字列に変換
- **hex(整数)**
  - 整数を16進文字列に変換
- **unhex(文字列)**
  - 16進文字列を数値に変換
- **trim(文字列)**
  - 文字列から前後の空白を除去
- **join(文字列配列)**
  - 文字列配列の要素を連結
- **split(文字列)**
  - 文字列を空白で分割(joinの逆)

# 5.9 参考：ファイル処理の例

---

```
// データファイルの形式:  
// -100～100の数値を1行に1ずつ入れる  
float[] data;  
  
void setup() {  
    size(400, 200); noLoop();  
    stroke(100); fill(255);  
    rectMode(CORNER);  
}  
  
void draw() {  
    background(200);  
    line(0, height/2, width, height/2);  
    if (data == null) return;  
    int w = width / data.length;  
    for (int i = 0; i < data.length; i++) {  
        rect(w * i + w/2, height/2,  
              w, -data[i]);  
    }  
}
```

```
void mouseClicked() {  
    // ファイル選択ダイアログを開く  
    selectInput("Open", "fileSelected");  
}  
  
// ファイル選択後の処理  
void fileSelected(File file) {  
    if (file == null)  
        println("File not found. ");  
    else  
        loadData(file.getAbsolutePath());  
}  
  
void loadData(String fname) {  
    String[] lines = loadStrings(fname);  
    data = new float[lines.length];  
    for (int i = 0; i < lines.length; i++)  
        data[i] = float(lines[i]);  
    redraw();  
}
```