Graphics with Processing

2022-05 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（折れ線か多角形）
  - 「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

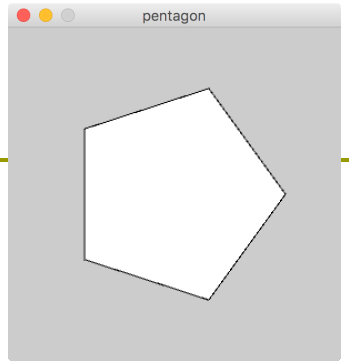
頂点の追加

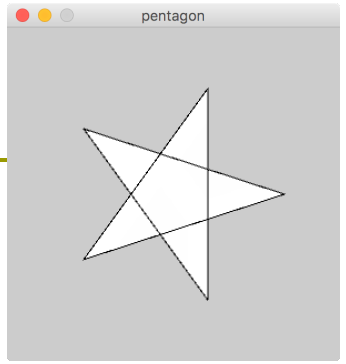
- vertex(x, y)
  - 図形に新しい頂点を追加する
- curveVertex(x, y)
  - 曲線を描く（4点以上連続で有効）
- bezierVertex(x1, y1, x2, y2, x3, y3)
  - ベジエ曲線をつなげて追加する

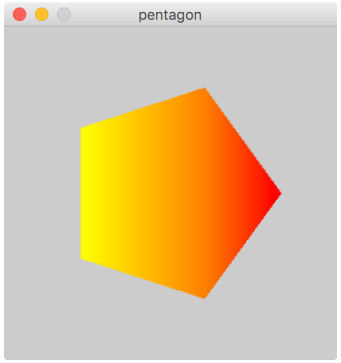
描画設定

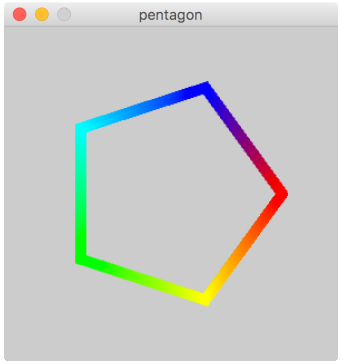
- strokeJoin(モード)
  - 頂点での接続形状を指定できる
  - MITER, BEVEL, ROUND
- 頂点ごとの着色
  - P2DまたはP3Dモードで可能
  - 最初に size(幅, 高さ, P2D);

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);  frameRate(30);
}

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);
}
```

P2Dモードを指定

いろいろ改造してみよう
1. 頂点ごとに色をつける
2. 星型にする
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 図形全体の移動

幾何変換（詳細は次回）

- translate(x_0, y_0)
  - 上下左右への平行移動
  - 以後すべての座標値に(x_0, y_0)を加えて描画する
- scale(α, β)
  - 拡大・縮小
  - 以後すべての座標値を横にα倍，縦にβ倍して描画する
- rotate(θ)
  - 回転
  - 以後すべての座標を原点中心にθラジアン回転して描画する
- pushMatrix()
- popMatrix()
  - 座標の変更部分の前後を囲む
  - popとpushは必ず対にして使う

基本的な書き方

```
pushMatrix();
translate(x移動量, y移動量);
rotate(回転角);
scale(x拡大率, y拡大率);
/* beginShape等での図形描画 */
popMatrix();
```

穴の空いた図形

- beginContour()
- endContour()
  - beginShape()～endShape()の中で使い，図形の内側に“穴”をあけることができる
  - 図形の外側の輪郭とは逆回転の向きに，vertexを指定していく

5.4 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()
  - ボタンは押されずに，マウスが動かされたとき
- 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

5.5 タイポグラフィ(文字表示)

文字列の表示

- text(文字列, x, y);
  - 文字列の表示
  - 前景色(fill)で描画される
  - 「¥n」を入れると改行する
- textSize(サイズ)
  - 文字サイズの設定
- textLeading(画素数)
  - 改行幅の設定
- tesxtAlign(横モード, 縦モード)
  - 基準点からのそろえ方を設定
  - 横モード: LEFT, CENTER, RIGHT
  - 縦モード: TOP, BOTTOM, CENTER, BASELINE

フォントの指定

- PFont型
  - フォントを表すクラス
  - 例) PFont font1
- loadFont("ファイル名")
  - [ツール]→[フォント作成...]で作っておいたProcessing用のフォントデータを読み込む
- createFont("フォント名", ポイント数)
  - 実行環境(OS,Java)が提供するフォントから動的にフォントデータを生成する
- textFont(フォント)
- textFont(フォント, サイズ)
  - 表示フォントの設定

5.6 サンプルプログラム

```
// 準備: Tahoma 48pt のフォントを作成
PFont font1, font2;
color tc = #0000ff; // RGB16進数形式

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

  // フォントの読み込みは必ずsetupで行う
  font1 = loadFont("Tahoma-48.vlw");
  font2 = createFont("Serif", 48);
}

void mousePressed() {
  switch (mouseButton) {
  case LEFT:   tc = #00ff00; break;
  case RIGHT: tc = #0000ff; break;
  }
}
```

コールバック関数の利用例

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

  fill(0);
  textFont(font1, 12);
  textAlign(LEFT, TOP);
  text(frameRate, 8, 8);

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

  fill(tc);
  textFont(font2, 48);
  textAlign(CENTER, CENTER);
  text("角度 " + frameCount, 0, 0);
  popMatrix();
}
```

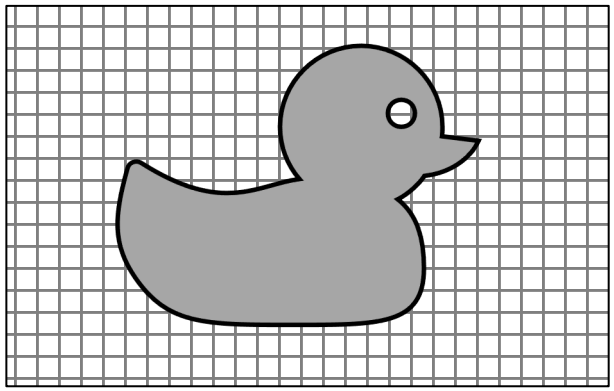
文字列の表示例

幾何変換による移動例

5.7 演習課題

課題

- beginShape()～endShape()を使って，有名なキャラクターやロゴマークのシルエット（影絵）を描き，それを動かすプログラムを作成しなさい
  - 発展例：各頂点に色をつける / マウスやキーの操作に反応する
  - 注意：curveVertexは4つ以上並べないと有効にならない



作り方の例

1. 方眼紙（またはペイントソフト）を用意して，キャラクターやロゴマークの輪郭線を描く
2. 輪郭線上で，間隔を開けて点を取り，それらの点を順につなぐと形ができるようにする
3. 方眼紙の目盛りから，それぞれの点のxy座標の値を調べる
4. プログラムでは，それらの座標を順に引数にして，vertex関数（やcurveVertex関数）を並べる
5. 図形を動かすには，表示処理の前に，translateやrotateを実行する方法（5.3）が手軽

5.8 参考:画像変形

2Dテクスチャマッピング

- texture(画像)
  - 図形の中に画像を貼る
  - 画像: PImage型(第3回参照)
  - 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(0, 0, P2D);
  pic = loadImage("sharaku.jpg");
  // ウィンドウサイズの変更
  surface.setResizable(true);
  surface.setSize(pic.width, pic.height);
  frameRate(30);
}

void draw() {
  textureMode(NORMAL);
  beginShape(TRIANGLE_FAN);
  texture(pic);
  vertex(mouseX, mouseY, 0.5, 0.5);
  vertex(0, 0, 0.0, 0.0);
  vertex(width, 0, 1.0, 0.0);
  vertex(width, height, 1.0, 1.0);
  vertex(0, height, 0.0, 1.0);
  vertex(0, 0, 0.0, 0.0);
  endShape();
}
```

頂点を共有する三角形

5.9 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dataフォルダにコピーしておく（[スケッチ]→[ファイルを追加]）

```
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）

5.10 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();
}
```