

計算機実習 III

第2回： 静止画の作成2

担当： 佐藤

2018年4月17日(火)

前回課題解説

※ チャレンジ問題は解説しない。解答例をCoursePowerで確認せよ

2018年4月17日(火)

課題 1

```
size(800, 600);
rect(250, 50, 100, 500);
rect(150, 150, 500, 100);
rect(450, 50, 100, 500);
rect(150, 350, 500, 100);
rect(250, 250, 100, 300); // overwrite
```

課題2

```
float diam = 500; // diameter

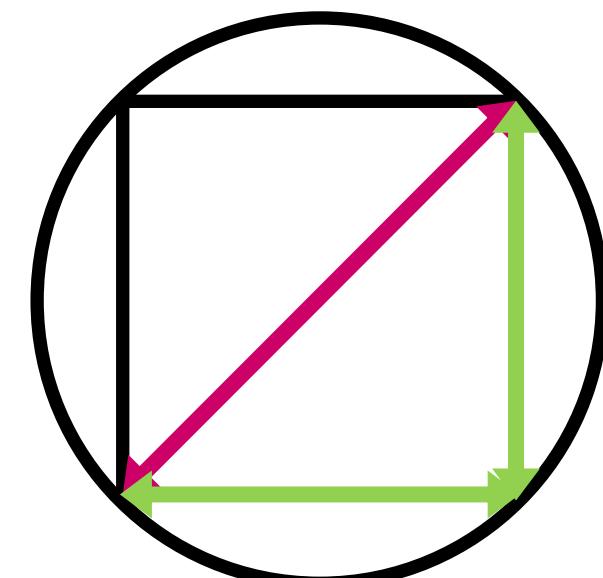
void setup() {
    size(600, 600);
    rectMode(CENTER);
    for (int i = 0; i < 10; i++) {
        drawShape(width / 2, height / 2);
    }
}

void drawShape(float x, float y) {
    ellipse(x, y, diam, diam);
    diam /= sqrt(2);
    rect(x, y, diam, diam);
}
```

四角形の描画の基準点を変更

反復処理は繰り返し文で行う

定型処理は関数にまとめる



課題3

```
int n = 5; // number of sides of a polygon
int iAngle = 360 / n; // interior angle
float r = 250; // radius

size(600, 600);
beginShape(TRIANGLE_FAN);
vertex(width / 2, height / 2);
for (float i = 0; i < n + 1; i++) {
    float theta = radians(i * iAngle);
    float x = width / 2 + r * cos(theta);
    float y = height / 2 + r * sin(theta);
    vertex(x, y);
}
endShape();
```

図形内部の線
はオプション
で描ける

課題4

```
size(800, 600);
for (float i = 0; i < 360; i += 5) {
    float theta = radians(i);
    float x = map(cos(theta), -1, 1, 0, width);
    float y = map(sin(theta), -1, 1, 0, height);
    line(width >> 1, y, x, height >> 1);
}
```

三角関数の値の範囲
をウィンドウサイズ
に合わせて変更

図形の性質

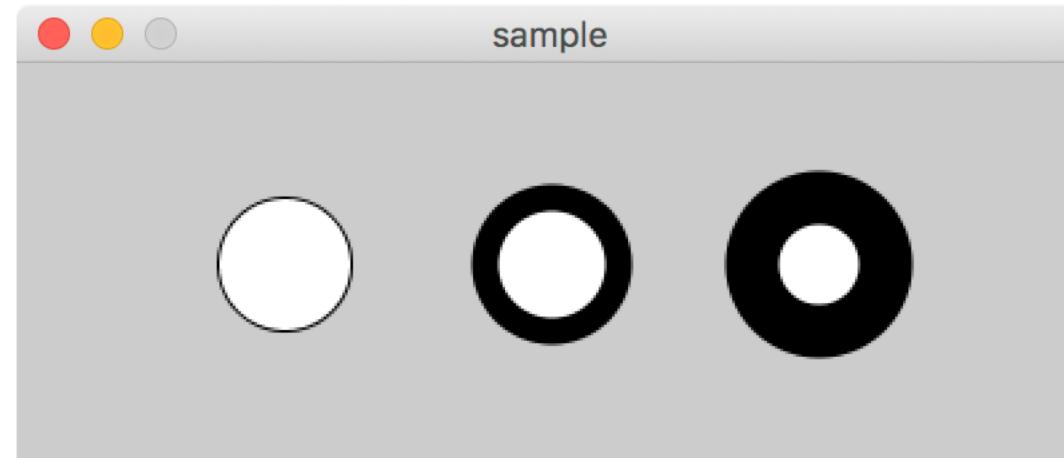
2018年4月17日(火)

strokeWeight()

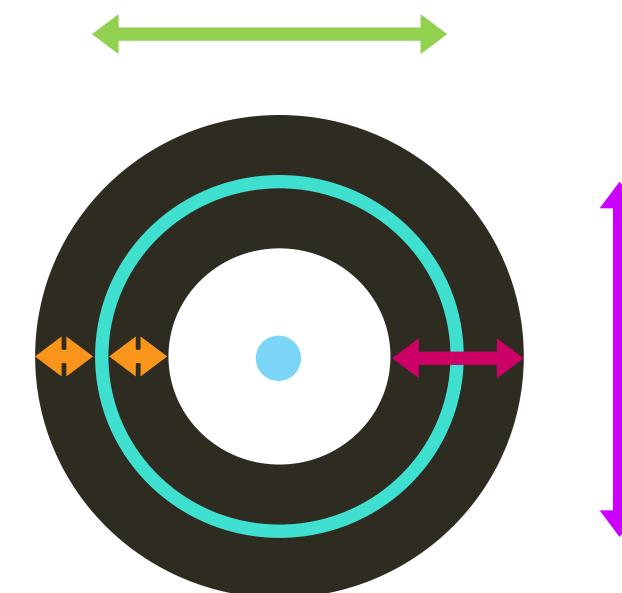
- **strokeWeight()**は、線の太さを設定する関数
 - 引数：線の太さ(ピクセル数)

例

```
size(400, 150);
ellipse(100, 75, 50, 50);
strokeWeight(10);
ellipse(200, 75, 50, 50);
strokeWeight(20);
ellipse(300, 75, 50, 50);
```



- 太さは線の中心から両側に半分ずつ反映される
 - `strokeWeight(10)` → 各側の太さ == 5
 - `strokeWeight(20)` → 各側の太さ == 10



strokeJoin(), strokeCap()

- **strokeJoin()**は、線のつなぎ方(角の形)を指定する関数

- 引数

- MITER(**デフォルト**): 直角
- ROUND: 円形
- BEVEL: 斜め

- **strokeCap()**は、線の両端の形状を指定する関数

- 引数

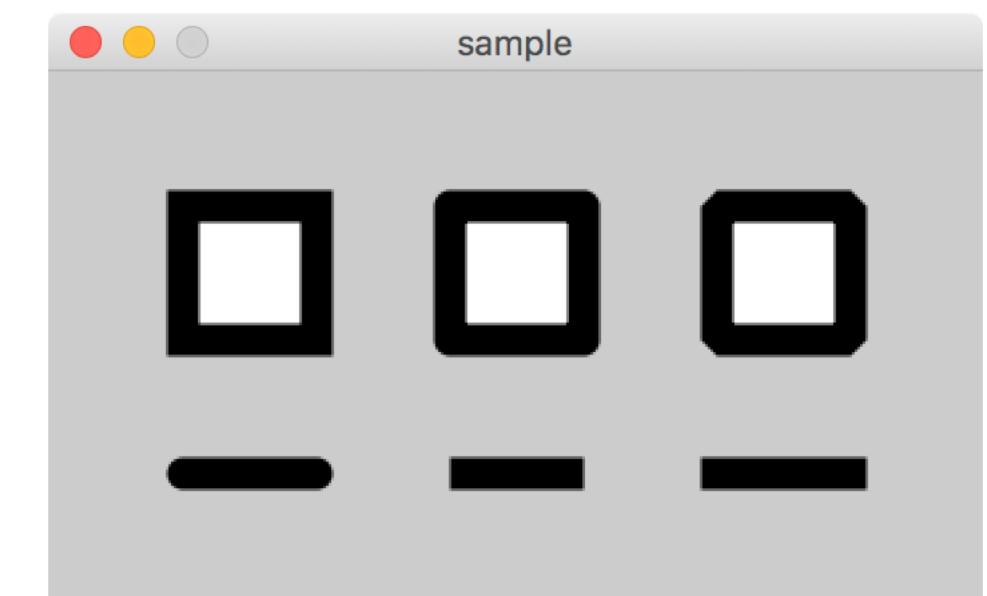
- ROUND(**デフォルト**): 円形
- SQUARE: 直角
- PROJECT: 直角(両端を太さの半分伸ばす)

例

```
size(350, 200);
strokeWeight(12);
rect(50, 50, 50, 50);
strokeJoin(ROUND);
rect(150, 50, 50, 50);

(右へ続く)
```

```
strokeJoin(BEVEL);
rect(250, 50, 50, 50);
line(50, 150, 100, 150);
strokeCap(SQUARE);
line(150, 150, 200, 150);
strokeCap(PROJECT);
line(250, 150, 300, 150);
```



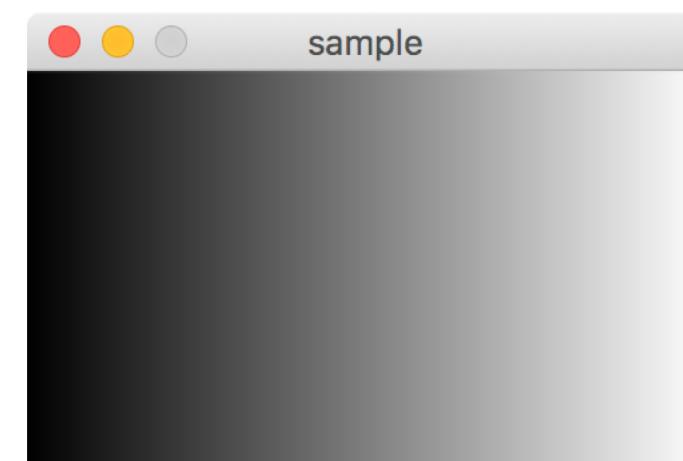
背景色，線色，図形色

background(), stroke(), fill()

- **background()**は，背景色を設定する関数
 - 省略可→自動的に「background(204);」が実行される
- **stroke()**は，線色(図形の枠線も含む)を設定する関数
- **fill()**は，図形色(図形の中身の色)を設定する関数
 - これら3つの関数は，引数に0~255の整数を与えると256段階のグレイスケールで色が設定される
 - **0(黒)↔255(白)**

例

```
size(255, 150);
for (int i = 0; i < 255; i++) {
    stroke(i);
    line(i, 0, i, 150);
}
```



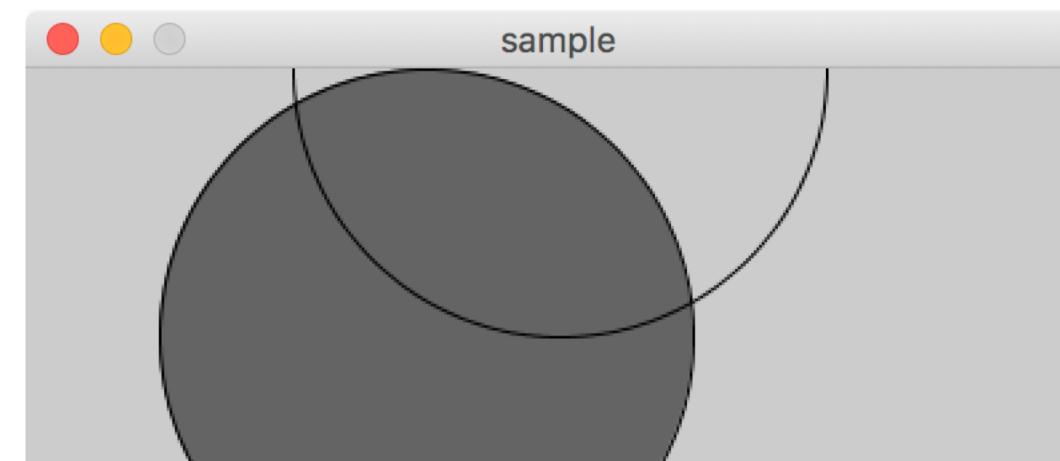
noFill(), noStroke()

- **noFill()**は、図形色を無効(図形内部が透明)にする関数
 - fill(...)を実行すると再び図形色が有効になる
- **noStroke()**は、線色を無効(線が透明)にする関数
 - stroke(...)を実行すると再び線色が有効になる
- noFill()とnoStroke()を両方実行 → **描画内容が見えなくなる！**

例

```
size(400, 150);
fill(100);
ellipse(150, 100, 200, 200);
noFill();
ellipse(200, 0, 200, 200);
noStroke();
ellipse(250, 100, 200, 200);
```

見えない！

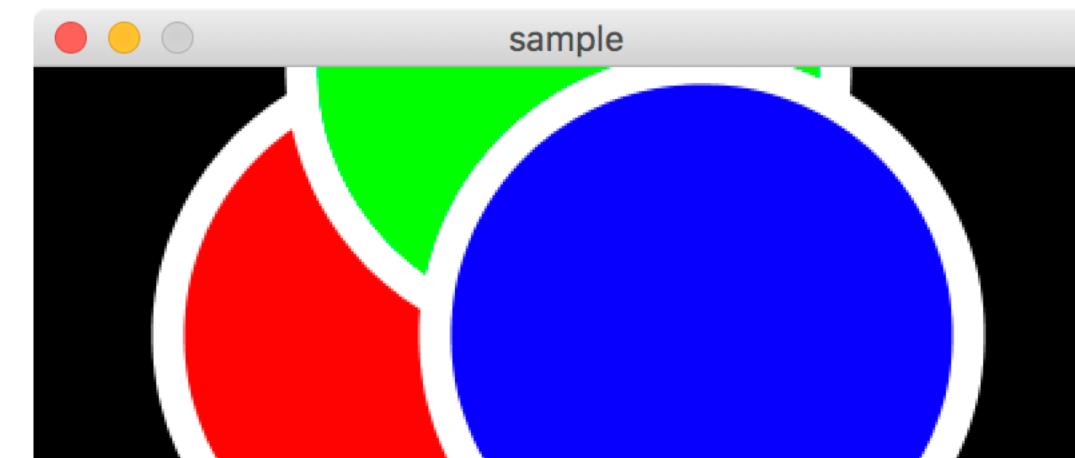


RGB

- Processingのデフォルトの色モデルはRGB
- background(), stroke(), fill()の引数に赤，緑，青の光の三原色に対応する3つの値(0~255)をこの順に設定することで，グレイスケール以外の色を設定できる

例

```
size(400, 150);
background(0) ; // black
strokeWeight(12);
stroke(255); // white
fill(255, 0, 0); // red
ellipse(150, 100, 200, 200);
fill(0, 255, 0); // green
ellipse(200, 0, 200, 200);
fill(0, 0, 255); // blue
ellipse(250, 100, 200, 200) ;
```



HSB

- Processingでは、RGBとは別の色モデルHSBも扱える
 - ・「**colorMode(HSB);**」でデフォルトのRGBからHSBへ切り替える
- HSBは色相(Hue), 彩度(Saturation), 明度(Brightness)に対応する3つの値(**0~255**)をこの順に設定することで、グレイスケール以外の色を設定できる
 - ・原色は彩度と明度がいずれも255(100%)
 - ・色相は赤を0とする角度[deg]で表現される
 - RGBと異なり、1つの引数(色相)だけで色を変化させられる

例

```
size(255, 150);
colorMode(HSB);
for (int i = 0; i < 255; i++) {
  stroke(i, 255, 255);
  line(i, 0, i, 150);
}
```



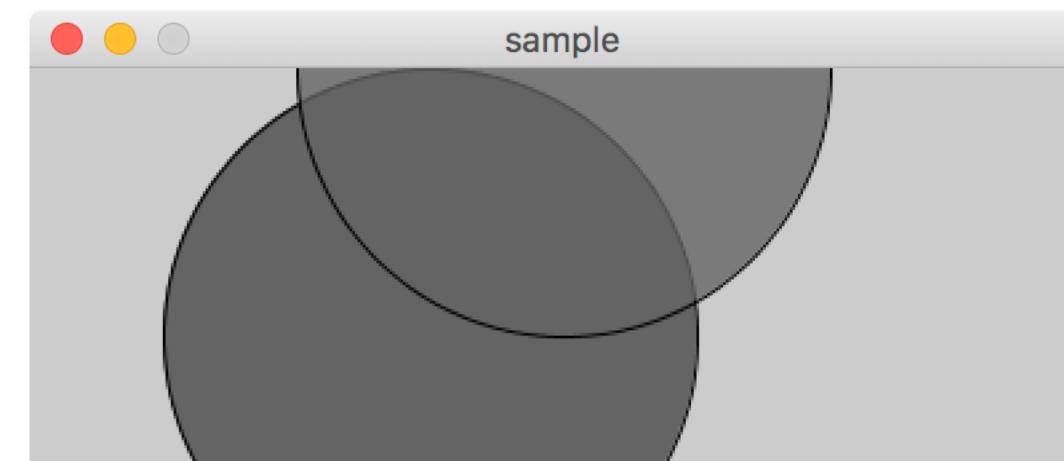
アルファ値

- fill()やstroke()にグレースケールならば2つ目，RGBまたはHSBならば4つ目の引数を加えることで**透明度**を設定可能。透明度の値を**アルファ値(0~255)**という
 - 0(透明)↔255(不透明)

例

```
size(400, 150);
fill(100, 255);
ellipse(150, 100, 200, 200);
fill(100, 200);
ellipse(200, 0, 200, 200);
fill(100, 0);
stroke(100, 0);
ellipse(250, 100, 200, 200);
```

見えない！



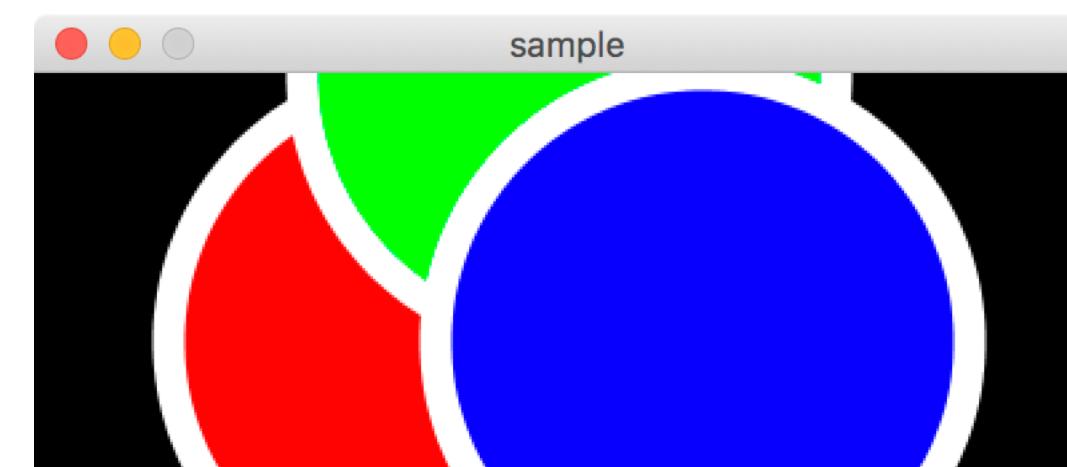
color型

- Processingでは、色データを格納するデータ型としてcolor型が用意されている

例

```
color[] c = { color(255, 0, 0),  
              color(0, 255, 0),  
              color(0, 0, 255) };  
  
size(400, 150);  
background(0) ; // black  
strokeWeight(12);  
stroke(255); // white  
fill(c[0]); // red  
ellipse(150, 100, 200, 200);  
fill(c[1]); // green  
ellipse(200, 0, 200, 200);  
fill(c[2]); // blue  
ellipse(250, 100, 200, 200);
```

色データは、color()の引数に色の値を入れることで作成



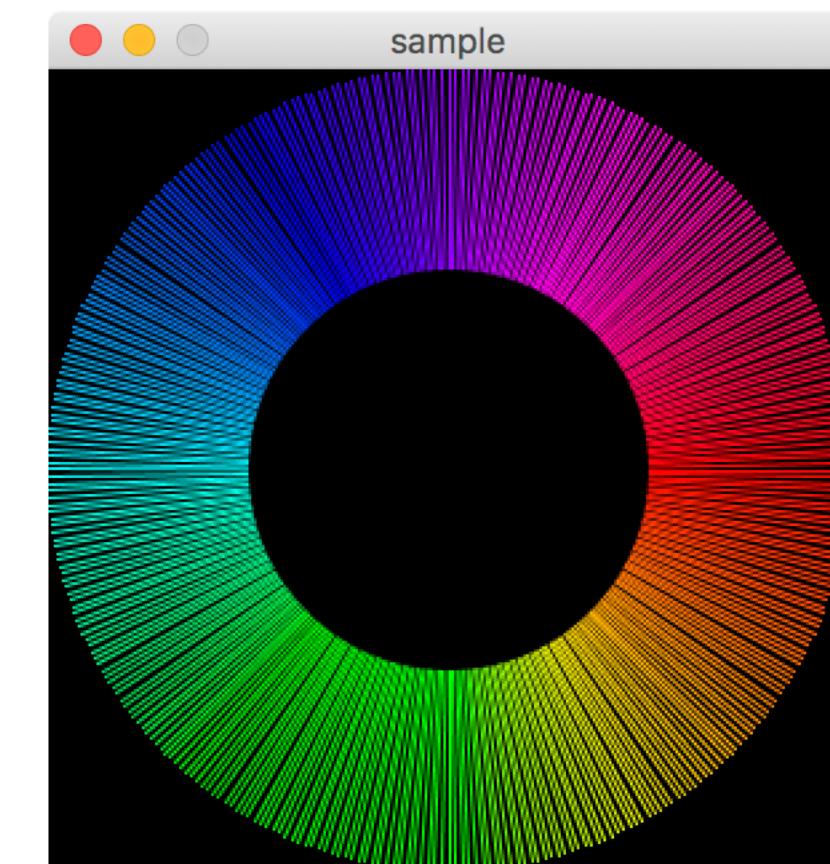
colorMode()

- **colorMode()**は， RGB↔HSBの切り替えを行うだけでなく， 値の範囲を変更する機能も持つ
 - 値の範囲を変更すると**再度変更しない限りそのまま**

例

```
float radius = 150;  
  
size(300, 300);  
float cX = width / 2;  
float cY = height / 2;  
background(0);  
colorMode(HSB, 360, 100, 100);  
for (int i = 0; i < 360; i++) {  
    float x = cX + radius * cos(radians(i));  
    float y = cY + radius * sin(radians(i));  
    stroke(i, 100, 100);  
    line(cX, cY, x, y);  
}  
noStroke();  
fill(0);  
ellipse(cX, cY, radius, radius);
```

色相を0～360，彩度と明度を0～100の範囲に変更



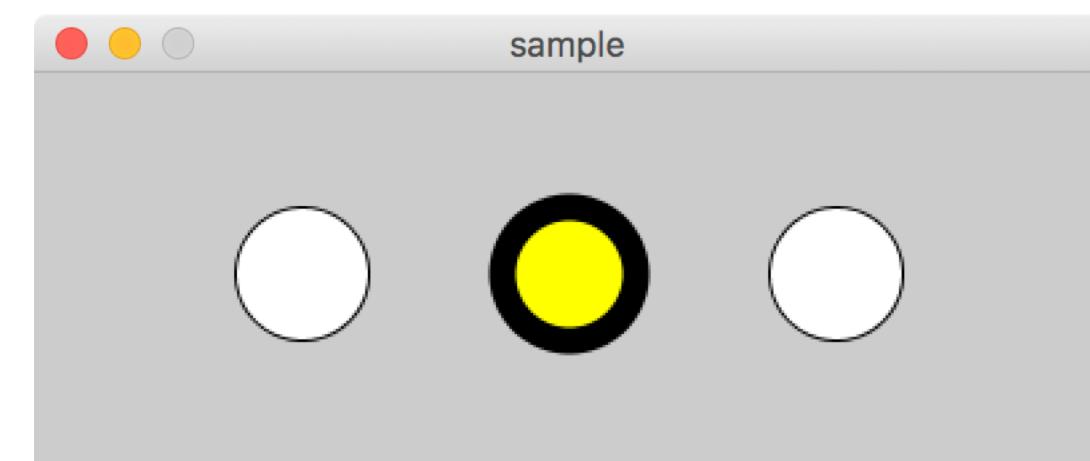
pushStyle(), popStyle()

- 「**pushStyle();**」と「**popStyle();**」の間で設定されたスタイルは、この間に限定される
 - 対象となるスタイル
 - `fill()`, `stroke()`, `strokeWeight()`, `strokeCap()`, `strokeJoin()`, `rectMode()`, `ellipseMode()`, `colorMode()`, etc.

例

```
size(400, 150);
ellipse(100, 75, 50, 50);
pushStyle();
strokeWeight(10);
fill(#FFFF00); // fill(0xFFFF00);
ellipse(200, 75, 50, 50);
popStyle();
ellipse(300, 75, 50, 50);
```

「`fill(255, 255, 0);`」
のように引数を3つ用
いすとも引数1つで色
を設定する方法もある
(リファレンス参照)



文字



text()

● text() は、文字を描く関数

- text("...", x, y)
 - ""で囲まれた部分の文字列を(x, y)の位置に表示
- text(s, x, y)
 - sに格納されているString型の文字列を(x, y)の位置に表示
- text(s, x1, y1, x2, y2)
 - (x1, y1)を左上端とする幅x2, 高さy2のテキストボックス内にsを表示

例

```
String s = "Pride goes before destruction.";
size(200, 150);
fill(0);
rect(10, 10, 80, 130);
fill(255);
text(s, 10, 10, 80, 130);
fill(0);
rect(100, 0, 100, 150);
fill(204);
rect(110, 10, 80, 130);
fill(0);
text(s, 110, 10, 80, 130);
```

- 文字色は fill() で設定
 - デフォルト: 白



textSize()

- **textSize()**は、テキスト(※)の大きさを指定する関数

- 引数：テキストの大きさ(ピクセル数)

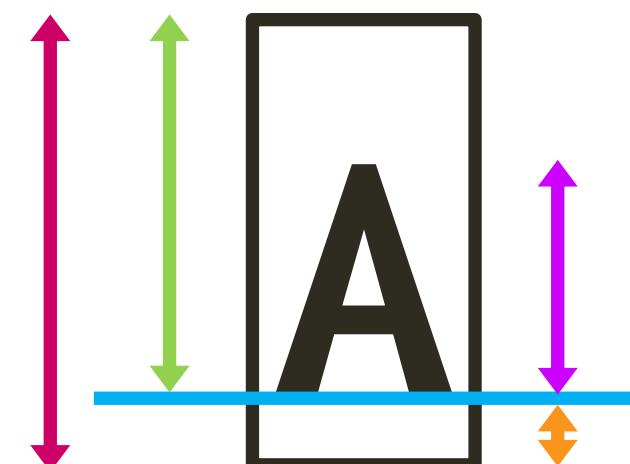
例

```
float base = 10;
float scalar = 0.61;           フォントごとに異なる

size(200, 200);
fill(0);
rect(10, 10, 180, 180);
stroke(255);
fill(255);
for (int i = 1; i <= 9; i += 2) {
  textSize(10 * i);
  float textHeight = (textAscent() + textDescent()) * scalar;
  base += textHeight;
  text("AGU", 10, base);
  line(10, base, 190, base) ;
}
```

- フォントの高さ == **textAscent()** + **textDescent()**
 - **textAscent()**: ベースラインから現在のフォントの上端までの長さ
 - **textDescent()**: ベースラインから現在のフォントの下端までの長さ

テキストの高さ



textAlign()

- **textAlign()**は、テキストのそろえ方を指定する関数

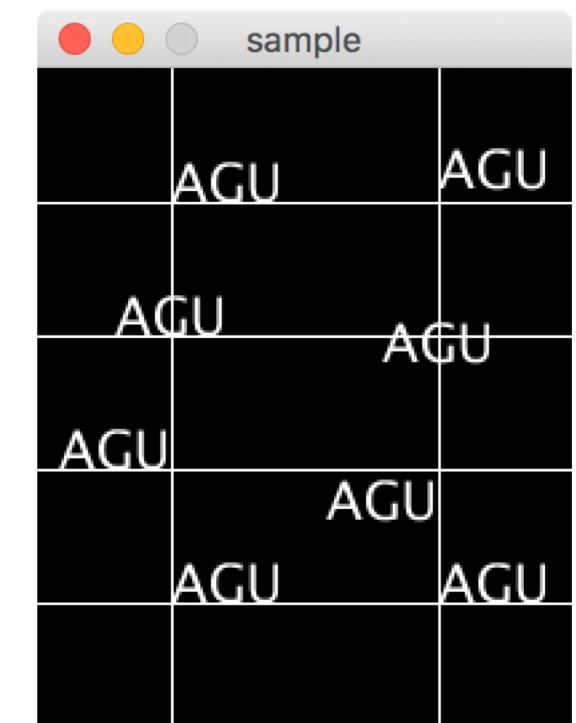
- **textAlign(h)**
- **textAlign(h, v)**
 - h: 水平方向(RIGHT, CENTER, LEFT(デフォルト))
 - v: 垂直方向(BOTTOM, CENTER, TOP, BASELINE(デフォルト))

例

```
void setup() {  
    size(200, 250);  
    background(0);  
    textSize(20);  
    stroke(255);  
    drawLines(); 省略  
    fill(255);  
    drawLeftChars();  
    drawRightChars();  
}
```

```
void drawLeftChars() {  
    float x = 50;  
    text("AGU", x, 50);  
    textAlign(CENTER);  
    text("AGU", x, 100);  
    textAlign(RIGHT);  
    text("AGU", x, 150);  
    textAlign(LEFT);  
    text("AGU", x, 200);  
}
```

```
void drawRightChars() {  
    float x = 150;  
    textAlign(LEFT, BOTTOM);  
    text("AGU", x, 50);  
    textAlign(CENTER, CENTER);  
    text("AGU", x, 100);  
    textAlign(RIGHT, TOP);  
    text("AGU", x, 150);  
    textAlign(LEFT, BASELINE);  
    text("AGU", x, 200);  
}
```

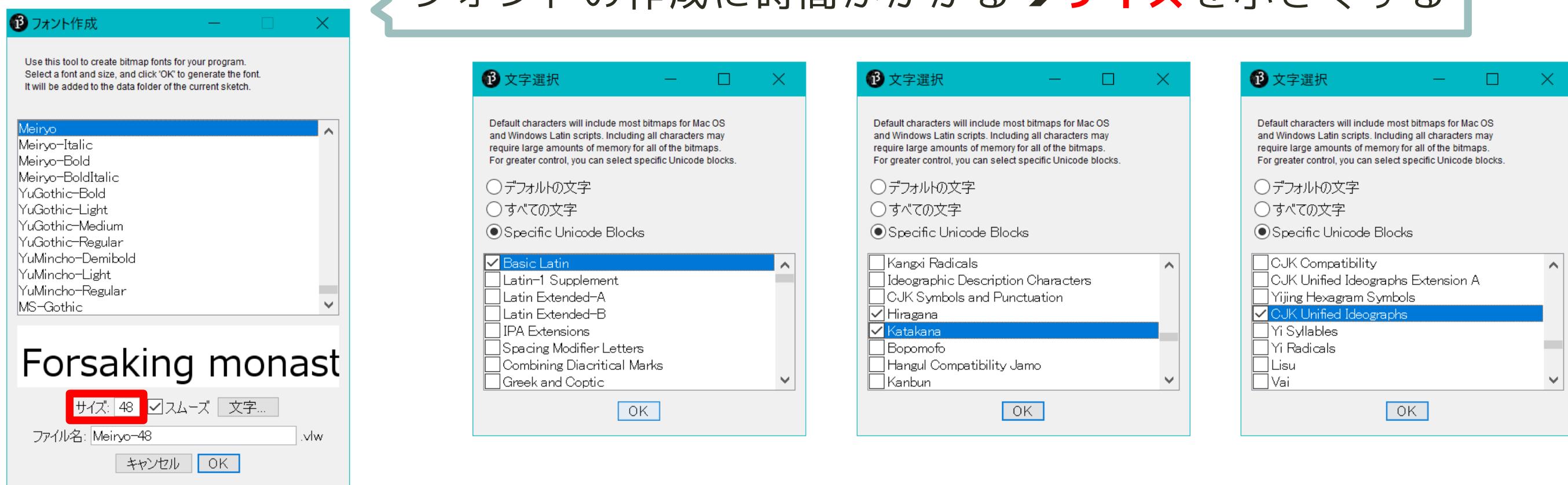


フォントの作成

● フォントの作成手順

- ① ツールバーの「ツール」→「フォント作成...」からフォント作成ダイアログを開く
- ② 作成したいフォントを選択
- ③ フォント作成ダイアログの「文字」から文字選択ダイアログを開く → 「Specific Unicode Blocks」を選択 → 以下の4つを選択 → 「OK」
 - Basic Latin, Hiragana, Katakana, CJK Unified Ideographs
- ④ フォント作成ダイアログの「OK」→ 作成中のスケッチのスケッチフォルダの直下に「data」という名前のフォルダが作成され、その中に作成したフォント(拡張子が「.vlw」のファイル)が入る

フォントの作成に時間がかかる → サイズを小さくする



フォントの設定

● フォントの設定手順

- ① フォントをdataフォルダに追加
- ② **PFont型**の変数を宣言
- ③ **loadFont()**の戻り値を変数に代入
 - 引数: フォント名
 - **setup()有→setup()の中で行う**
- ④ **textFont()** → フォントの設定完了
 - 引数: PFont型の変数

例

```
PFont font;

void setup() {
    size(200, 150);
    setStyle();
    text("AGU", width / 2, height / 2);
}
```

```
void setStyle() {
    stroke(0);
    fill(0);
    font = loadFont("Meiryo-24.vlw");
    textFont(font);
    textSize(90);
    textAlign(CENTER, CENTER);
```



作成したフォントサイズより大きなサイズで表示→ぼやける