Design Programming

Spring 2018 - 4

2018.5.9 Keio University, SFC

char型について

型名	型の種類	例
int	整数	255, 100
float	小数	3.14, 0.
char	文字	'A'
boolean	真偽値	true, false
color	カラー	RGB値

char型について

```
char a = 'a';
char b = "b";
char c = 'abc';
char d = z;
```

- シングルクォートで囲む.
- １文字まで代入可能

Stringについて

```
String s = "test";
```

- ダブルクォートで囲む.
- 複数の文字を代入可能
- StringのSは大文字（Stringはオブジェクト）

Stringを扱うときの演算子

```
String a = "test";
String b = "test";

if (a == b) {
    println("aとbは同じです");
}
```

メモリの同じ位置に格納されているか

比較してしまう

文字列が等価である条件として

等価演算子は使えない

Stringを扱うときの演算子

```
String a = "test";
String b = "test";

if (a.equals(b)) {
    println("aとbは同じです");
}
```

equals()メソッドを使うことで内容を
比較し，等価ならtrueを返す

ドット演算子を使用することで
Stringオブジェクトの
・フィールド（オブジェクト内の変数）
・メソッド（オブジェクト内の関数）
などを呼び出せる

Stringオブジェクトのメソッド

メソッド	意味	例，形式
charAt(x)	文字列のうちx番目の文字	s.charAt(2);
equals(a)	文字列aと等価の場合true	s.equals(t);
length()	文字列の長さ（文字数）	s.length();
substring(x, y)	文字列のうちx番目からy番目までの文字列	s.substring(1, 4);
toLowerCase()	文字列を小文字にする	s.toLowerCase();
toUpperCase()	文字列を大文字にする	s.toUpperCase();

text関数

```
fill(0);
text("test", 0, 40);
```

(0, 40)に「test」を書く


```
textSize(40);
fill(0);
text("test", 0, 40);
```

テキストのサイズを40に変更


```
textSize(40);
fill(0);
text("test1", 0, 40);
fill(255);
text("test2", 0, 80);
```

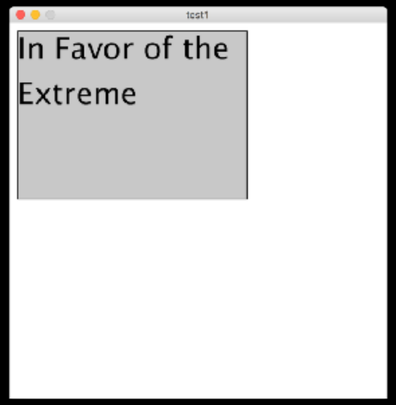
塗り色をそれぞれ設定

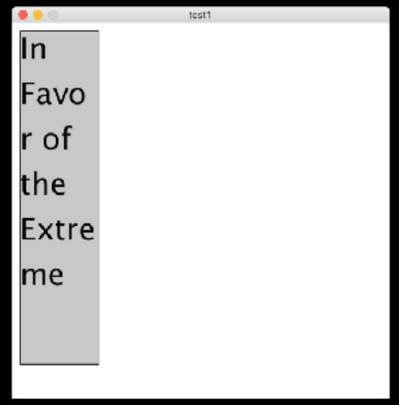
text関数

```
String s = "In Favor of the Extreme";

void setup() {
    size(500, 500);
    textSize(40);
}

void draw() {
    background(255);
    fill(200);
    rect(10, 10, mouseX, mouseY);
    fill(0);
    text(s, 10, 10, mouseX, mouseY);
}
```

text関数の４つめ，５つめのパラメータで

テキストボックスの幅と高さを指定する

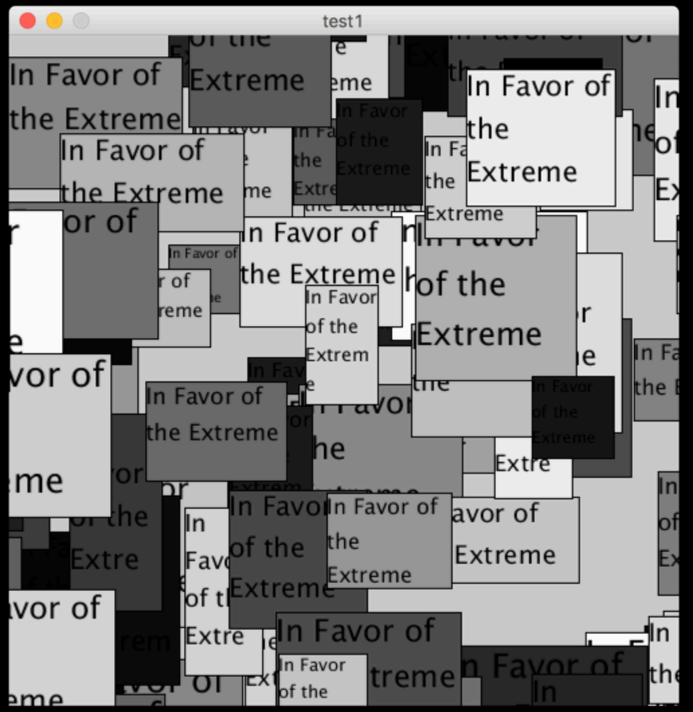




演習1

さきほどの例を利用して以下の条件で右のような描きなさい.

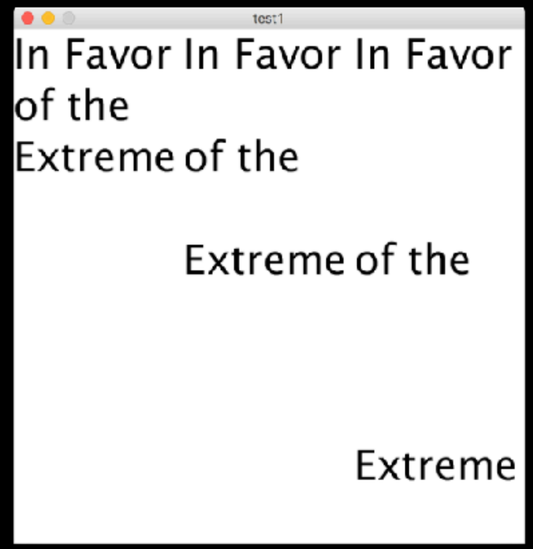
- for文を使用してテキストボックスを100回描く
- テキストの座標はx: -150 — width, y: -150 — heightの乱数
- 幅高さはw: 50 — 150, h: 50 — 150の乱数
- 色は0 — 255の乱数
- テキストサイズは（幅 + 高さ） / 10



テキスト属性

```
String s = "In Favor of the Extreme";
size(500, 500);
textSize(40);
background(255);
fill(0);
textLeading(50);
text(s, 0, 0, width / 3, height);
textLeading(100);
text(s, width / 3, 0, width / 3, height);
textLeading(200);
text(s, width * 2 / 3, 0, width / 3, height);
```

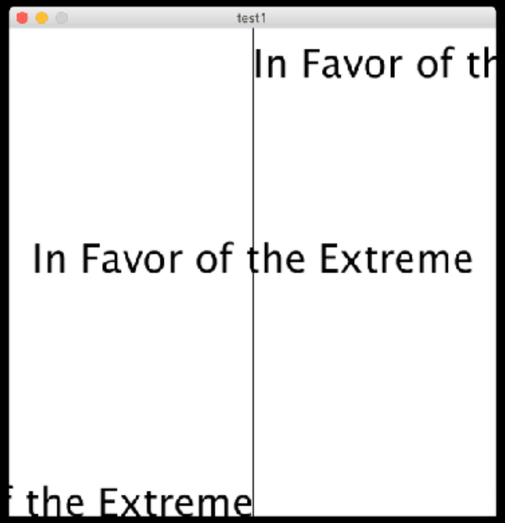
textLeading関数で行間隔を調整する



テキスト属性

```
String s = "In Favor of the Extreme";
size(500, 500);
textSize(40);
background(255);
fill(0);
textAlign(LEFT);
text(s, width / 2, 50);
textAlign(CENTER);
text(s, width / 2, height / 2);
textAlign(RIGHT);
text(s, width / 2, height);
line(width / 2, 0, width / 2, height);
```

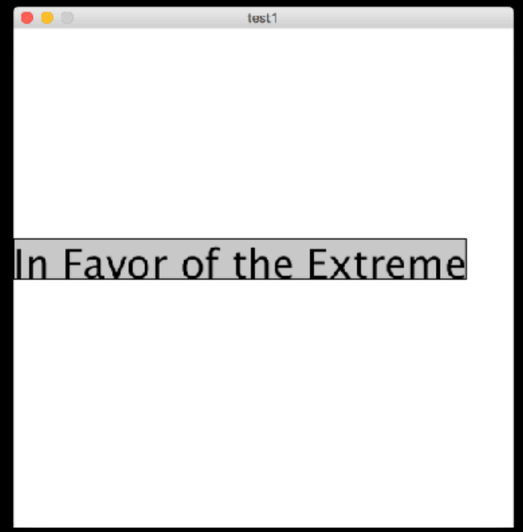
textAlign関数のカッコ内をLEFT, CENTER, RIGHTいずれかを書くことでテキストの行揃えを指定する



テキスト属性

```
String s = "In Favor of the Extreme";
size(500, 500);
textSize(40);
background(255);
fill(200);
rect(0, height / 2, textWidth(s), -40);
fill(0);
text(s, 0, height / 2);
```

textWidth関数でテキスト幅がわかる



ローカルフォントを調べる

```
String[] fontList = PFont.list();
printArray(fontList);
```

コンソールにローカルフォントが

全て表示される

```
[2044] "ZenGoNStd-Bold"
[2045] "ZenGoNStd-DeBold"
[2046] "ZenGoNStd-Heavy"
[2047] "ZenGoNStd-Light"
[2048] "ZenGoNStd-Medium"
[2049] "ZenGoNStd-Regular"
[2050] "ZenGoNStd-Ultra"
```

ローカルフォントを使う

```
PFont f;
f = createFont("futura", 40);
textFont(f);
fill(0);
background(255);
text("test", 0, 40);
```

PFontという独自のデータ型の変数を宣言

createFont関数を使ってフォントを変換

textFont関数を使って現在のフォントを指定

ローカルフォントを使う

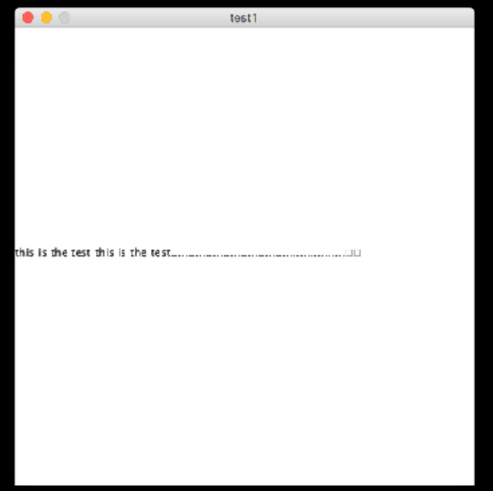
```
PFont f1, f2;
f1 = createFont("futura", 40);
f2 = createFont("osaka", 40);
fill(0);
background(255);
textFont(f1);
text("test", 0, 40);
textFont(f2);
text("test", 0, 80);
```

複数のローカルフォントもPFontを複数

宣言することで可能

インタラクティブなタイポグラフィ

```
String s = "";

void setup() {
    size(500, 500);
    fill(0);
}
void draw() {
    background(255);
    text(s, 0, height / 2);
}
void keyPressed() {
    if (key == BACKSPACE) {
        if (s.length() >= 1) {
            s = s.substring(0, s.length() - 1);
        }
    } else {
        s += key;
    }
}
```



課題

- 今日学習した「テキスト，フォント，タイポグラフィ」を使用してスケッチを１つアップロードしなさい.
  アップロード先は「03_text」とする.
  ※今日学習していない分野でもすでに自分が知っている技術は使用可能とする.

  〆切：5月15日24時まで

次回

- 画像を扱う
- 画像からの情報抽出