後ろ5列は着席禁止
3人掛けの中央は着席禁止

プログラミング入門
Processingプログラミング
第9回

九州産業大学　理工学部情報科学科
神屋郁子
(pp@is.kyusan-u.ac.jp)

時限	クラス
水1	機械（クラス3）
水2	機械（クラス1）
水4	電気（B1、B2）

今後の予定

第9回：複数の図形(2)
　　繰り返しと座標変換
第10回：画像の表示と音の再生

配列というやや難しい内容を扱う。
後期以降に詳しく説明するのでよくわからなくても心配しないように。

プログラミング入門では、配列の概要とどんなものが作れるのかイメージをつかむことができればOK！

第11回：応用課題プログラムの開発
第12回：Wordの基本操作と　応用課題プログラムについて
　　Wordを用いたレポート作成
第13回：Excelの基本操作と
　　Excel VBAによるプログラミング
第14,15回：　Javaプログラミング
定期試験

第9回の内容

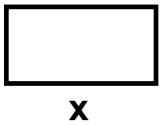
- 座標変換
  - 並行移動、回転、拡大
- 配列
  - 配列の基礎
  - データに従って図形を描く

復習）変数

- データを格納する入れ物
- 変化するデータを扱うのに使う
- processingで用意されている変数
  - mouseX, mouseY, width, height
- 自分で変数を作ることもできる（変数宣言）

型　名前;

例）**int x;**　整数を扱う名前がxという変数を作る

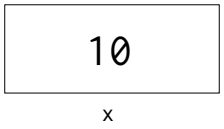


復習）値を格納する（代入）

- 代入の構文

  変数の名前 = データ;

- データは具体的な数値であったり式であったり
  - 例 `x = 10;`



復習）条件分岐

- もし〇〇なら××する(条件によって行動を変える)

```
if(条件){
    処理
}
```

条件がtrue(正しい)なら文を実行

- 例：もしx座標が右端(360)より大ならば、xに0を代入

```
if (x > 360) {
      x = 0;
}
```

↑閉じ括弧までが一つのまとまり

開き括弧と
閉じ括弧の間に
条件が成り立つとき
の処理を書く

復習）繰り返しfor文

- for文の書き方

```
for(初期化;条件;更新){
    処理
}
```

初期化：ループ変数の初期化
条件　：ループを続けるかどうか
更新　：ループ変数の更新

復習）0から9までの値を表示

- 0 1 2 3 4 5 6 7 8 9 と値を増やして表示する

初期化
変数宣言と代入
変数iを作り0に

条件
iが9以下の間
(i<10 でもよい)

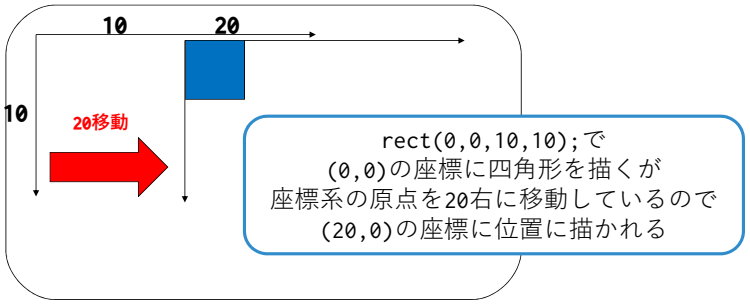
更新
iを1ずつ増やす
(i=i+1と同じ)

```
for(int i = 0; i <= 9; i++){
    print(i+",");
}
```

0から9まで1ずつ値を増やして繰り返し表示

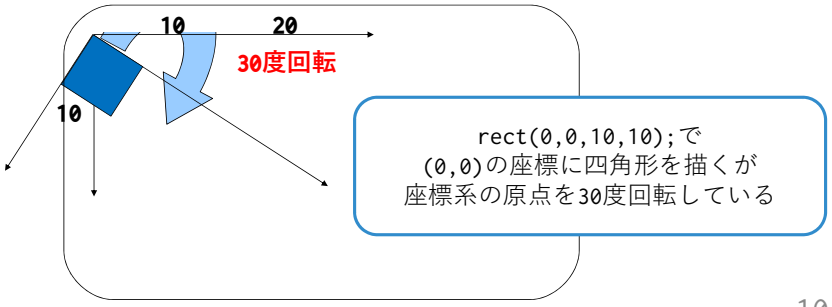
並行移動

```
translate(20,0);
rect(0,0,10,10);
```



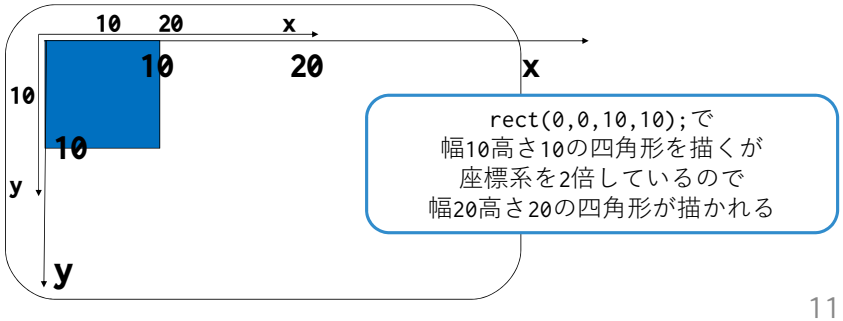
回転

```
rotate(radians(30));
rect(0,0,10,10);
```



拡大・縮小

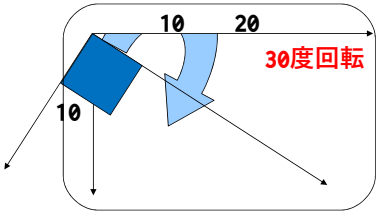
```
scale(2);
rect(0,0,10,10);
```

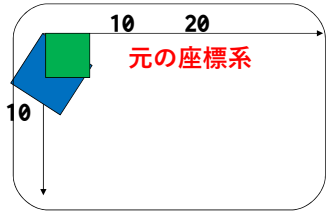


pushMatrix/popMatrix

```
pushMatrix();
rotate(radians(30));
rect(0,0,10,10);
popMatrix();
rect(0,0,5,5);
```

pushMatrixで座標系を保存
回転させても
popMatrixで座標系を戻せる





注意

- 今回も基本的に画面の変化のないプログラム
  - setup、drawの形ではない
  - 座標変換、for、配列の理解に集中しよう
- 余裕がある人はsetup、drawの構造にして色や動きを派手に！

作業

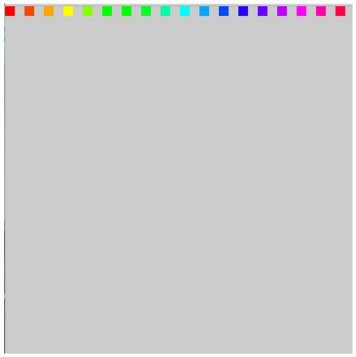
1. 座標変換
   1. 並行移動
   2. 回転
   3. 拡大・縮小
   4. 座標の保存・復元
2. 配列
   1. 配列の基礎
   2. 配列を使用した描画
   3. 二次元配列を利用した描画


```
size(360, 360);
colorMode(HSB, 359, 99, 99);
noStroke();

for (int i = 0; i< 360; i+=20) {
  fill(i, 99, 99);
  rect(0, 0, 10, 10);
  translate(20, 0);
}
```

rectの座標は(0,0)で変数を含んでいないところに注目

演習1

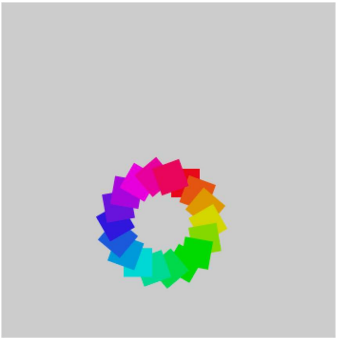
並行移動 translate




```
size(360,360);
colorMode(HSB,359,99,99);
noStroke();

translate(180,180);
for(int i = 0; i < 360; i+=20){
     fill(i,99,99);
     rect(0,0,30,30);
     rotate(radians(20));
     translate(20,0);
}
```

演習2

回転 rotate



演習3

拡大・縮小　scale

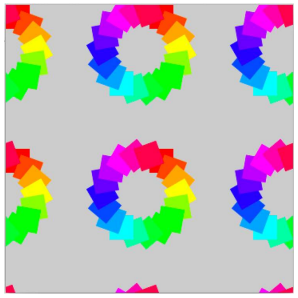
```
size(360,360);
colorMode(HSB,359,99,99);
noStroke();

translate(180,180);
for(int i = 0; i < 360; i+=20){
     fill(i,99,99);
     rect(0,0,30,30);
     rotate(radians(20));
     translate(20,0);
     scale(1.1);
}
```


```
size(360,360);
colorMode(HSB,359,99,99);
noStroke();
for(int y = 0; y <= 360; y+=180){
    for(int x = 0; x<= 360; x+=180){
        pushMatrix();
        translate(x,y);
        for(int i = 0; i< 360; i+=20){
            fill(i,99,99);
            rect(0,0,30,30);
            rotate(radians(20));
            translate(20,0);
        }
        popMatrix();
    }
}
```

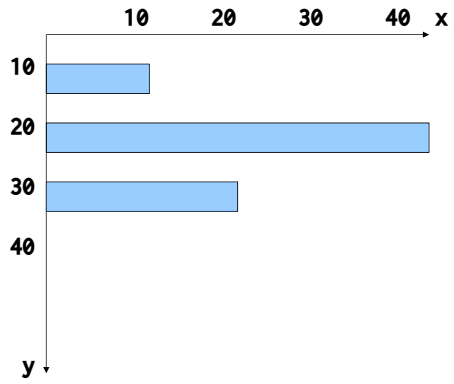
繰り返しの最初で座標を保存しておくと、いくら変更しても元の座標系に戻せる。

演習4

座標の保存・復元
pushMatrix, popMatrix



データに対応した棒グラフ




```
int d1 = 11;
int d2 = 43;
int d3 = 21;
```
データ

```
rect(0,10,d1,5);
rect(0,20,d2,5);
rect(0,30,d3,5);
```


配列を使ってデータをまとめる

```
int d1 = 11;  int d2 = 43;  int d3 = 21;
```

型 [] 配列の名前={データ1,データ2,...};

1つ目、2つ目、3つ目

```
int [] d={11, 43, 21};
```

配列d

11	43	21
d[0]	d[1]	d[2]

```
rect(0, 10, d[0], 5);
rect(0, 20, d[1], 5);
rect(0, 30, d[2], 5);
```

もっとまとまらない？

for文でもっと簡潔に

```
rect(0, 10, d[ 0 ], 5);
rect(0, 20, d[ 1 ], 5);
rect(0, 30, d[ 2 ], 5);
```

```
int [] d = {11,43,21};
```

配列の長さ d.length ここでは3

```
for(int i = 0; i  < d.length; i++) {
    rect(0, ( i+1)*10 , d[ i ], 5);
}
```


配列

- 同一の型のデータを複数格納できる特別な変数
- 配列の宣言と初期化
  - 型 [] 配列の変数の名前 = {データ1, データ2, データ3, …};
- 配列の個々のデータ（要素）へのアクセス
  - 配列の変数の名前[番号] 番号のことをインデックスまたは添え字と呼ぶ。先頭は0番目。
  - 配列の要素数は配列の変数の名前.length で取得できる


```
int [] a = { 1, 7, 3, 2 };

println(a[0]);
println(a[1]);
println(a[2]);
println(a[3]);
```

演習5

配列

型 [] 名前 = {データ1,データ2,…,データn};
整数を扱う名前がaの配列を作り、
データで初期化

名前[番号] の要素の値を表示
名前がaの配列の先頭(0番目)の値を表示

1	7	3	2


```
for(int i = 0; i < a.length; i++){
    println(a[i]);
}
```

演習5

繰り返し(for文)を使って表示を簡潔に

1	7	3	2

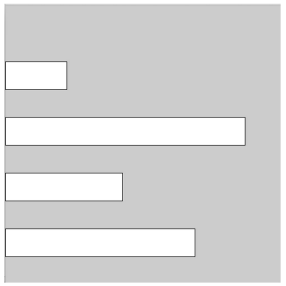
演習6

配列を使ってデータに対応した大きさの複数の長方形を棒グラフ風に描く

```
size(500,500);
int [] data = {110,430,210,340};

for(int i = 0; i< data.length; i++){
     rect(0, (i+1)*100, data[i], 50);
}
```

ループ変数iを使ってy座標を毎回下にずらす

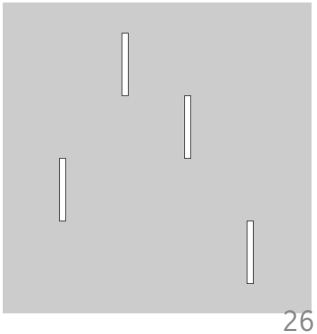
配列dataの要素の値の横幅



演習7

向き（回転）のデータ用の配列と回転と描画を追加

```
size(500,500);
rectMode(CENTER);
int [] x = {100,200,400,300};
int [] y = {300,100,400,200};

for(int i = 0; i< x.length; i++){
     pushMatrix();
     translate(x[i],y[i]);
     rect(0, 0, 10, 100);
     popMatrix();
}
```

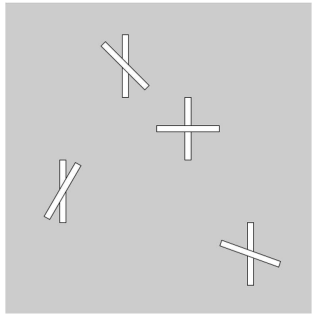


演習8

2次元配列を使ってデータに対応した複数の図形を描く

```
size(500,500);
rectMode(CENTER);
int [] x = {100,200,400,300};
int [] y = {300,100,400,200};
int [] r = {30,-45,110,90};

for(int i = 0; i< x.length; i++){
     pushMatrix();
     translate(x[i],y[i]);
     rect(0, 0, 10, 100);
     rorate(radians(r[i]));
     rect(0,0,10,100);
     popMatrix();
}
```

回転前の図形が不要な場合はここを削除



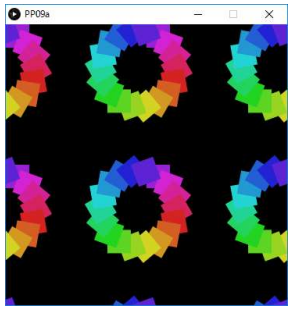
プログラムの提出（演習点）

- 3,4,6,7,8のどれかをベースにアレンジしてみよう！
  - 他の人のプログラムをコピーするのはNGですよ！
  - 周りの人とは違う改良をしよう！
- K’s Lifeのレポート機能から
- 締め切りは6日後の23:59
  - それ以降も受け取るが**減点**する
  - もっと改良したくてもこの時間に一度提出しよう
  - （削除して提出しなおし可能）
- 評価はK’s Life上ではしない

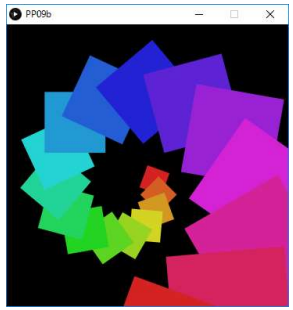
提出する内容

- 以下をコメントとして入力
  - 今回の内容の概要（どのような図形をどう並べたか）
  - 工夫した点
  - 質問（何かあれば）・感想（簡単だった・難しかったなど）
  - 入らない場合は３つ目の提出ファイルとして追加してもよい
- 提出ファイルとして以下の２つ
  - プログラム(*.pde)
  - 実行画像(PNGまたはJPG)
- 提出ファイルの名称にはそれぞれ学籍番号を入力

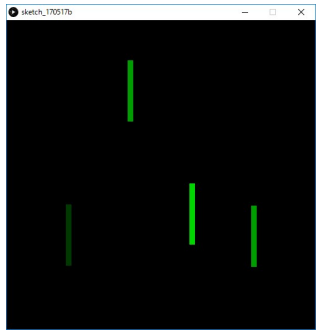

```
int cc = 0;
void setup() {
  size(360, 360);
  colorMode(HSB, 360);
  noStroke();
  frameRate(3);
}

void draw() {
  background(0);
  cc+=20;
  for (int y = 0; y <= 360; y += 180) {
    for (int x = 0; x <= 360; x += 180) {
      pushMatrix();
      translate(x, y);
      for (int i = 0; i < 360; i+=20) {
        fill((i+cc)%360, 300, 300);
        rect(0, 0, 30, 30);
        rotate(radians(20));
        translate(20, 0);
      }
      popMatrix();
    }
  }
}
```



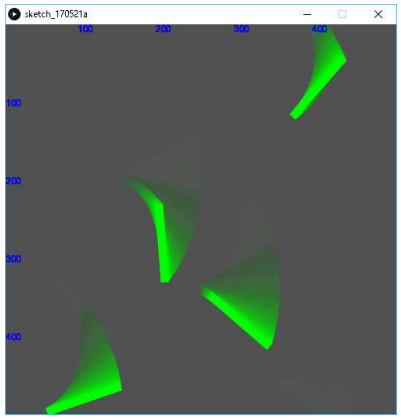

```
int n = 0;
void setup() {
  size(360, 360);
  colorMode(HSB, 360);
  noStroke();
  frameRate(30);
}

void draw() {
  background(0);
  n = (n+20)%1400;
  translate(180, 180);
  rotate(radians(n));
  for (int i = 0; i < n; i+=20) {
    fill(i%360, 300, 300);
    rect(0, 0, 30, 30);
    rotate(radians(25));
    translate(20, 0);
    scale(1.1);
  }
}
```



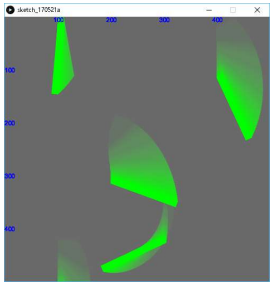

```
int [] x = { 100, 200, 300, 400 };
int [] y = { 200, 100, 300, 400 };

void setup() {
  size(500, 500);
  rectMode(CENTER);
}

void draw() {
  background(0);
  for (int i = 0; i < x.length; i++) {
    y[i] += random(0,21)-10;

    pushMatrix();
    fill(0,random(256),0);
    translate(x[i], y[i]);
    rect(0, 0, 10, 100);
    popMatrix();
  }
}
```



```
int [] x = { 100, 200, 300, 400 };
int [] y = { 300, 100, 200, 400 };
int [] r = { 30, -45, 90, 0 };


int f = 0;
int vf = 1;
void setup() {
  size(500, 500);
  noStroke();
}


void draw() {


  fade();
  fill(0, 0, 255);
  for (int tx = 100; tx < width; tx += 100) {
    text(""+tx, tx-8, 10);
  }
  for (int ty = 100; ty < height; ty += 100) {
    text(""+ty, 0, ty+5);
  }


  fill(0, 255, 0);
  for (int i = 0; i < x.length; i++) {
    rectMode(CENTER);
    r[i] = (r[i]+1)%360;
    //    y[i] = (y[i]+(int)random(3)-1)%500;
    y[i] = (y[i]+1)%500;
    pushMatrix();
    translate(x[i], y[i]);
    //rect(0, 0, 10, 100);
    rotate(radians(r[i]));
    rect(0, 0, 10, 100);
    popMatrix();
  }
}
```




```
int [] x = { 100, 200, 300, 400 };
int [] y = { 300, 100, 200, 400 };
int [] r = { 30, -45, 90, 0 };


int f = 0;
int vf = 1;
void setup() {
  size(500, 500);
  noStroke();
}


void draw() {


  fade();
  fill(0, 0, 255);
  for (int tx = 100; tx < width; tx += 100) {
    text(""+tx, tx-8, 10);
  }
  for (int ty = 100; ty < height; ty += 100) {
    text(""+ty, 0, ty+5);
  }


  fill(0, 255, 0);
  for (int i = 0; i < x.length; i++) {
    //rectMode(CENTER);
    r[i] = (r[i]+1)%360;
    //    y[i] = (y[i]+(int)random(3)-1)%500;
    y[i] = (y[i]+1)%500;
    pushMatrix();
    translate(x[i], y[i]);
    //rect(0, 0, 10, 100);
    rotate(radians(r[i]));
    scale(f/100.0);
    rect(0, 0, 10, 100);
    popMatrix();
  }
}


void fade() {
  rectMode(CORNER);
  fill(f, f, f, 10);
  f += vf;
  if (f < 0) vf = 1;
  if (f> 255) vf = -1;
  //f = (f + 1)% 256;
  rect(0, 0, width, height);
}
```



関係演算子

演算子	読み方	意味	利用例
<	小なり	～より小さい、～未満	a < b aはbより小さい
<=	小なりイコール	～以下	a <= b aはb以下
>	大なり	～より大きい	a > b aはbより大きい
>=	大なりイコール	～以上	a >= b aはb以上
==	イコールイコール	等しい	a == b aはbと等しい
!=	ノットイコール	等しくない	a != b aはbと等しくない

論理演算子

演算子	意味	利用例
&&	かつ	score >= 60 && score <= 100 scoreが60以上100以下 両方の条件が当てはまる
||	または	score < 0 || score > 100 scoreが0未満または100より大きい どちらかの条件が当てはまる
!	～でない	!(score == 100) scoreが100でない