コンピュータ基礎演習

第6回

理工学部 情報科学科 隅田 康明

sumida@ip.kyusan-u.ac.jp

レポート締め切りについて

- レポートの締め切り次回講義日の16時までに変更
  - 授業時間ではなく、
    締め切りに合わせて作業する人が大半となっているため
    - 対面授業での移動などもあるので、
      講義日中とするのもどうかと・・・
  - 授業時間中に前回のレポートの質問が出来るように

- ただし、なるべくその日のうちにレポートを提出しましょう
  - あくまで、質問しやすくするための措置
    - 1週遅れでいいと考えていると、いい成績は取れなくなる
      - 場合によっては単位も危なくなってくる
  - 第10回の締め切りは短くする予定
    - 第11回からは、制作課題を作成するため
      - 制作課題は40点、これを出すまでは他のレポートに取り組む余裕はなくなる

大学の自習室等の利用について

- 制限付きで自習室などが利用可能
  - 総合情報基盤センター等のPCを利用可能に
    - https://www.cnc.kyusan-u.ac.jp/aboutus/003560.html
    - 大学のPCにはProcessing、PowerPointがインストール済み（全部ではないですが）
- 通信環境の問題で、授業の受講が困難な場合も、自習室などで遠隔授業を受けることができます
  - https://www.kyusan-u.ac.jp/news/coronavirus20200519-2/
- 自宅で授業を受けられない場合には検討しましょう

  ※大学で作業することを推奨しているわけではありません

講義用HPについて

- 講義資料や、講義に必要な情報は、
  この授業用のホームページに掲載しています
  - K'sLifeの通知にも添付していますが、HPの方がアクセスはしやすいはずです
    - K'sLifeに接続しにくい状況が続いています
  - 念の為、K'sLifeにもアップロードはしますが、
    基本的には講義HPを見るようにしましょう
- 下記のHPから、講義資料、動画のURL、
  Zoomの招待リンクなどを確認できます。

http://www.is.kyusan-u.ac.jp/~sumida/class/pckiso/

  - だんだんMoodleに移行してきます
    - Moodleに行けば全部ある！の方が楽ですよね

遠隔授業期間中の質問

- **困ったら早めに質問・相談！！**
  - 学生側から質問されないと、
    誰が困っているのか、何が分からないのか、分かりません
- メール：やり取りに時間はかかるが一番確実
- Zoom：授業時間中限定
  - 時間は限られるが、作業中の画面を見ながら教えられるので、問題を短時間で解決出来る可能性が高い
- Line OpenChat：授業時間中限定
  - 文字だけのやり取りに限定（画像アップロードは禁止）
  - 質問内容が他の学生にも分かるので注意すること
    - プログラムの全文貼り付け等は厳禁！

授業についての質問メールについて

[授業名(曜日時限)]についての質問	件名
～先生	誰宛か
[授業名(曜日時限)]を受講しています、 20AA999の九産太郎です。	何者か
（質問内容） --	用件
20AA999 九産太郎 九州産業大学　芸術学部　○○学科 1年	署名

2日返信がなければもう一度送って下さい（なるべく見落としたくない）

各回のレポートの確認（必ず確認しておくこと）

提出ステータス

提出ステータス	評定のために提出済み
評定ステータス	未評定
終了日時	2020年 05月 25日(月曜日) 00:00
残り時間	4 日 2 時間
最終更新日時	2020年 05月 20日(水曜日) 21:19

提出すると「評定のために提出済み」になる

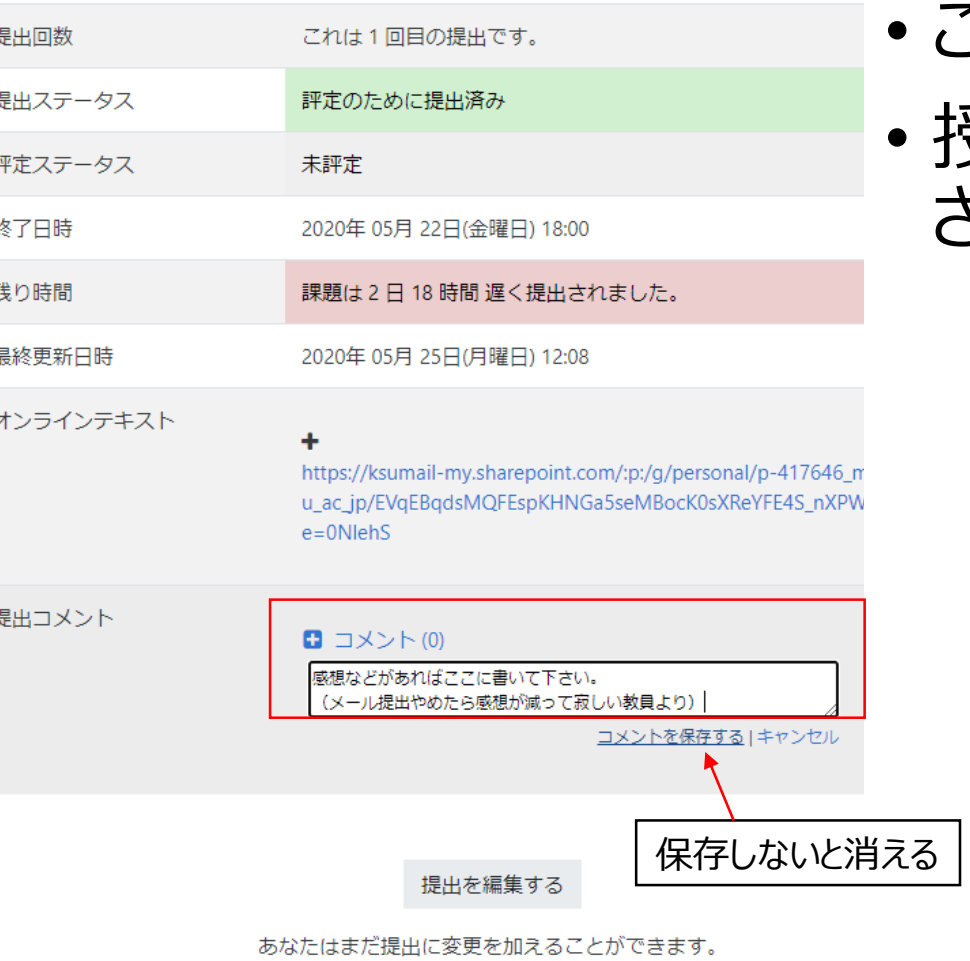
採点されると、「評定済み」になる

採点＝教員（私）が手動で点数を付ける

さらに下に、評点とフィードバックコメント（教員からのコメント）

毎回、何点だったかを確認しておこう（評点アップの交渉も遠慮せずにどうぞ）
（1回と2回はそのうち追加します(余裕が出来れば)）

Moodleのコメントについて



- こんな感じでコメント書けます。
- 授業の感想などはここに書いて下さい。
  - ただし、コメントへの返信は期待しないで下さい。
    - 返信欲しいことはメールで聞きましょう。
- 質問もメール等で！
  - レポートの採点はすぐに出来る訳ではありません（それなりに大変で時間もかかります）

そろそろ折り返しなので・・・

- 単位の取得とレポートについて
  - 難しい、で諦めないようにしましょう
    - 前も書いたけど、
      全員がしっかり分かってるわけではないです
    - 細かいこと考えるより前に、プログラムを書きましょう
      - そのうち分かってくればよい
  - 少し変えただけ、でも最低基準は満たす
    - ただし、小テストはしっかりやっておきましょう
  - 最後の課題をしっかり作ればいい成績も取れる
    - Sを取れるかはさすがに別になりますが

今回の授業内容

- ジェネラティブアート入門
  - ノイズとランダムを使って絵を描く
  - 同じものは二度と作れない
  - 生成される作品をテキトウに作ろう

レポート（PowerPoint）の注意

- スライドにアニメーションを付けないこと：採点が大変になるので
  - 最終回に提出するレポートは別
- 「プログラム」スライドの目的
  1. 採点のため：動かないプログラムは評価出来ない
  2. バックアップのため
     - どこに保存したか分からない　→　Moodleにあります
  3. プログラムの再利用
     - 前回までのプログラムをコピペで応用
- コピー＆ペーストでProcessingで動かせることが大事
  - Moodleからレポートをダウンロードし、プログラムをコピー＆ペーストで動かせればOK
  - Web版のOffice365でスライドを開くと、
    コピペで動かない場合があるので、一度ダウンロードして、
    アプリ版のPower Pointで開いてコピペすること

プログラムスライドについて

- 動かないプログラムを送ってきても評価できません

- 動く状態になってから、プログラムをコピー、貼り付けましょう。

- **「全て選択」「Select All」** → コピー
  - マウスやタップで手動で選択しないように！
    - 何回も貼り付けをすると、文字が消える
  - PowerPointでプログラムを修正しないように！
    - 貼り付けた後は余計なことはしない

ジェネラティブアート

- プログラム等によって生成されるアート
  - 自然科学的システムを主体とした創造作品

- プログラムで何が描かれるか？どう動くか？
  - これまでの作品：決めたとおりに動く
  - ジェネラティブアート：どうなるか分からない
    - 毎回違うものを創造する：ランダム要素

- デジタルアートの一種
  - デジタルアート：PCを使って作られたアートのこと
    - デジタルアート自体は非常に幅が広い

ランダム要素を加える

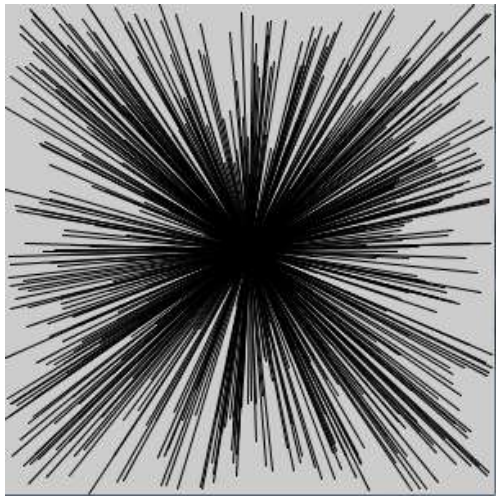
- プログラムにランダムな要素を加えると、
  作成にも予想できないような作品を作ることも出来る
  - 手軽にアートっぽくできる

- random(max);
  - 引数(括弧内の値)が１つ：0～maxまでの乱数を得る
- random(min,max);
  - 引数が2つ：min～maxまでの乱数を得る

ランダムに線を引くプログラム

- 始点(width/2,height/2)、
  終点ランダムな線を引くプログラムを作成せよ
- widthは実行画面の幅、
  heightは実行画面の高さを意味する変数

- 完成例

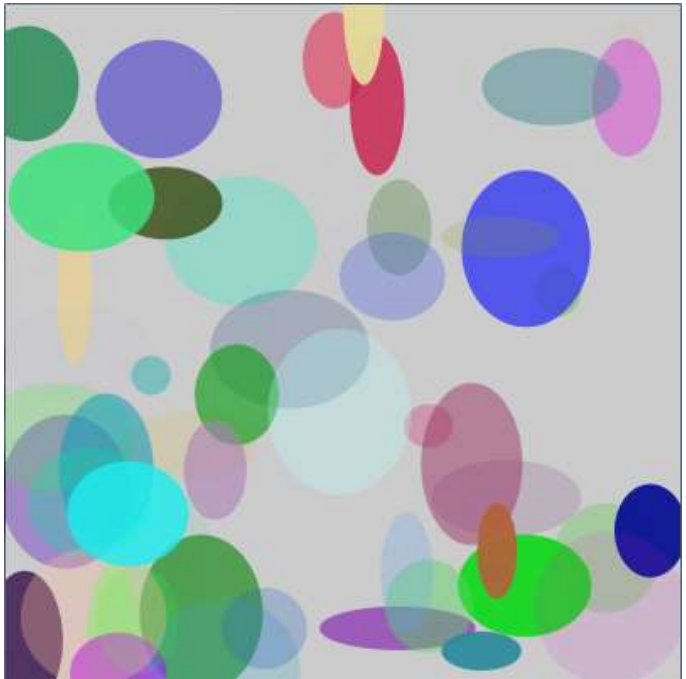


```
void setup(){
  size(300,300);
}

void draw(){
  line(width/2,height/2,random(width),random(height));
}
```

問題：ランダムに円を描く

- ランダムに円を描くプログラムを作成せよ
- 円の座標、円のサイズ、円の色（透明度も）、全てをランダムにすること
- ミニテストの問題

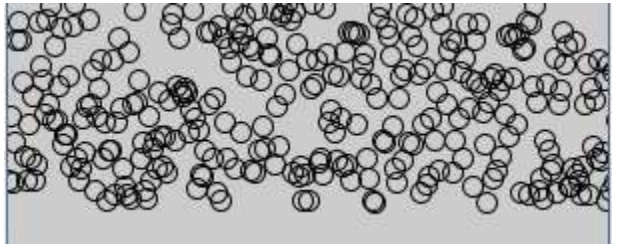


アレンジ

- 四角や丸、三角形など、
  別の図形をランダムに表示させてみる
- 座標だけでなく、色をランダムに変えてみる

ノイズ（連続性のあるランダム）

- ランダム(random)：前の値との連続性がない
  - 全く予想がつかない、
    無秩序な結果になる



- ランダム要素は欲しいが、ある程度のまとまりは欲しい
  ⇒　パーリングノイズを利用する

パーリンノイズ

- 連続性のあるノイズ
- 自然現象を表現するためによく用いられる
  - 水、雲、炎、煙、地形…
    - マイクラのマップ生成もパーリンノイズ利用
      - ランダム生成だけど、まとめりのある地形が生成されてますよね

- Processingでは、
  noise()メソッドを使うとパーリンノイズを生成できる

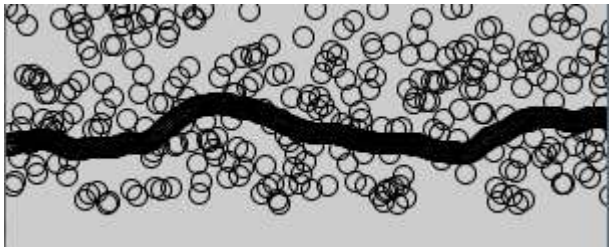
noise()：パーリンノイズを発生

- noise(シード値);
  - シード値に基づくノイズを発生
  - パーリンノイズと呼ばれる乱数発生技法
  - 前の値と近い値をランダムに発生

- シード値：乱数の元になる数
  - シードが同じ値だとノイズも同じ値になる
    - シードを少しずつ変えないといけない
  - シード値が小さいと連続性が高まる

ランダムとパーリンノイズの違い

- xを1ずつ増やして、yをランダムかノイズで円を描く



真ん中に集中しているのがノイズ

ランダムより纏まりはあるが、ランダムに変化する

- ランダム性がないと真っ直ぐ
- ただのランダムだと規則性がない
- ノイズを使うと連続性のあるランダムに
- どちらがいいかは用途によって異なる

パーリンノイズで線を描く

- 実行画面の中心から線を引くだけのプログラム
- ランダムで決めていた座標をノイズに変更する

```
float nx;
float ny;
float xnoise = random(1);
float ynoise = random(1);
void setup(){
  size(300,300);
}
void draw(){
  nx = noise(xnoise) * width;
  ny = noise(ynoise) * height;
  line(width/2,height/2,nx,ny);
  xnoise += 0.01;
  ynoise += 0.01;
}
```

x,y座標のノイズを発生させるための
シード値を格納する変数を宣言
シードの初期値はランダムにしておく

nx,ny座標を決定
最大値をノイズにかける

線の終点座標をx,yに

ノイズ用の変数を変化させる
変化量が大きいほど大きく変化

プログラムの解説

- noise(xnoise);で得られるのは、0～1の小数
  - 最小値が0、最大値が1
- 座標等にパーリンノイズを使いたい場合は、noise()の結果に最大値をかける
  - noise(xnoise) * width
    画面の横幅を最大値にしている。
    0～画面幅の数値を得られる
- ノイズのシード値を変化させないと同じ値になる
  - xnoise += 0.01;
    で少しずつシード値を変化させている

プログラムの解説

- 画面中心を始点にランダムに線を引いているだけ
- ただのランダムだと無秩序な線が重なるだけだが、
  連続性のあるランダムになることで面白い動きに

- パーリンノイズを利用することで、
  デタラメではないが、ランダムな動きが得られる
  - 作成者も予想できない動きを実現できる
    ⇒　作品がプログラムによって生成される

回転をプラス

```
float angle = 0;
int nx,ny;
float xnoise = random(1);
float ynoise = random(1);
void setup() {
  size(300,300);
  noFill();
  stroke(0,10);
}
void draw() {
  translate(width/2,height/2);
  rotate(angle);
  nx = (int)(noise(xnoise) * 200);
  ny = (int)(noise(ynoise) * 200);
  line(0,0,nx,ny);
  xnoise += 0.01;
  ynoise += 0.01;
  angle += 0.005;
}
```

原点を中心に移動

原点を少しずつ回転

線の長さをパーリンノイズで決める
200は線の長さの最大値

ノイズのシード値と角度を増やす

アレンジしてみよう

- 色を変えてみる
  - ランダムで色を変える、透過率を変える
- 図形を変えてみる
  - 三角形(triangle)、四角形(quad)、楕円(ellipse)
- 図形を増やしてみる
- ランダムに描画する図形も足してみよう
- ノイズのシード値の増加量を変えてみる
  - xnoise += 0.01; を xnoise += 0.05; など
  - 大きくするとランダム性が増す、
小さくすると連続性が増す

今回のレポート

- ひな形プログラムをアレンジしてレポートを提出
  - 最低基準
    - 色を付ける、図形を線から変更、図形を追加
    - ランダムに図形を配置するプログラムを追加
  - Power Pointでスライドを作って提出

- 加点項目
  - 同じ画面にもう一つ模様を描くプログラムを追加した
    - 実行例は後のスライドを参考に
  - その他、独自の工夫をした
    (ヒントに載っている工夫は対象外)

アレンジのヒント

- 線ではなく他の図形も描いてみる
  - line() を ellipse()や rect()に変える
- 回転角度やノイズのシード値の増加量を変えてみる
- 中心座標もランダム(ノイズ)にしてみる
  - line(0,0,x,y); の0,0もノイズで動かす
    - 変数を一通り追加する
- 座標をマウス座標にしてみる
  - line(mouseX,mouseY,x,y);
- rect(x,y,s3,s3); のように、自分で宣言した変数を使う

PowerPointでレポート作成

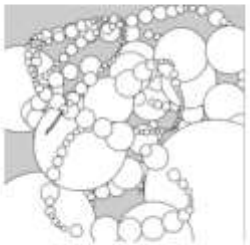
- 1枚目：タイトル

PC基礎 05

学籍番号　氏名

- 2枚目：実行画像

タイトル：（作品のタイトル）



- 説明
  - ～～をイメージして作った
  - マウスの動きに合わせて、丸の大きさが変わるように工夫した

どんなものを作ろうとしたのか？

どんな工夫をしたのか？

簡単でいいので説明を書く

- 3枚目：プログラム

プログラム

プログラムを全て貼り付ける

```
void setup(){
  size(400,400);
  colorMode(HSB,360,100,100,100);
}
void draw(){
  float d = dist(mouseX, mouseY, pmouseX, pmouseY);
  ellipse(mouseX,mouseY,d,d);
}
```

貼り付けたときに、文字が見えない程小さくなっても構わない

これは第3回のレポートの例

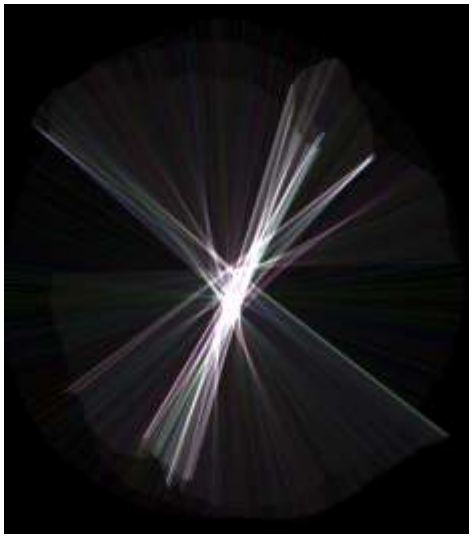
実行画面やプログラムは、
（当然）今回の内容にすること

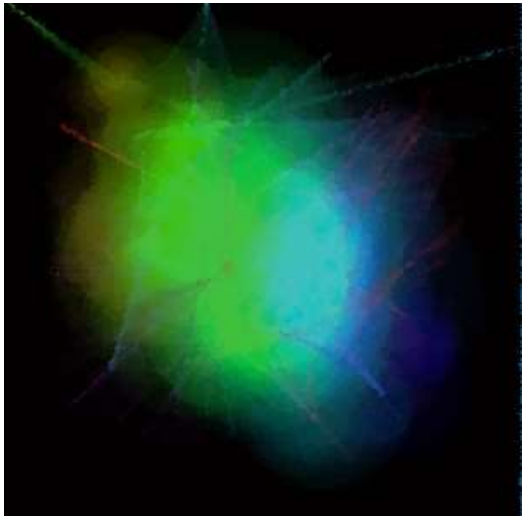
必要事項が揃っていれば、レイアウトは自由

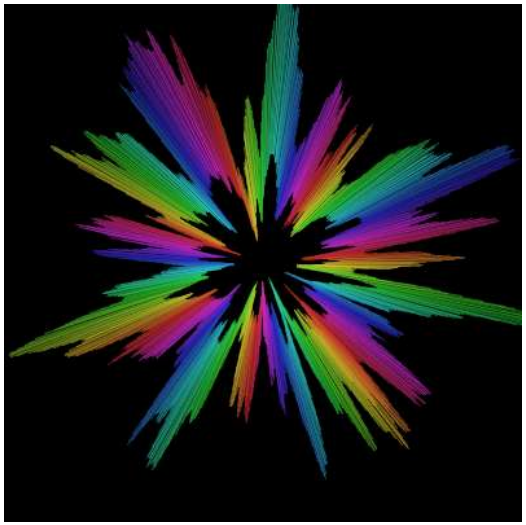
レポートチェックリスト（第6回）

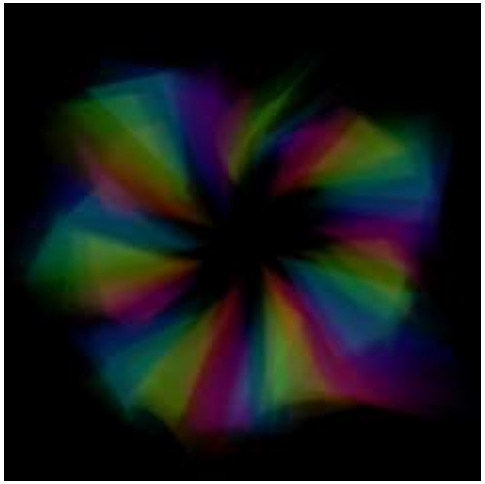
- ミニテストを受験した(レポート提出の前でも後でも可)
- 雛形プログラムをコピーしてProcessingに貼り付けた
- ひな形を自分なりにアレンジした
  - 図形を変更または追加した
  - 図形に色を付けた
    - ランダムに図形を描くプログラムを追加した
      ※これはあってもなくてもいい
- PowerPointでレポートを作成した
  - タイトル、作品紹介(工夫点)、プログラムの3枚
- Moodleでレポートを提出した

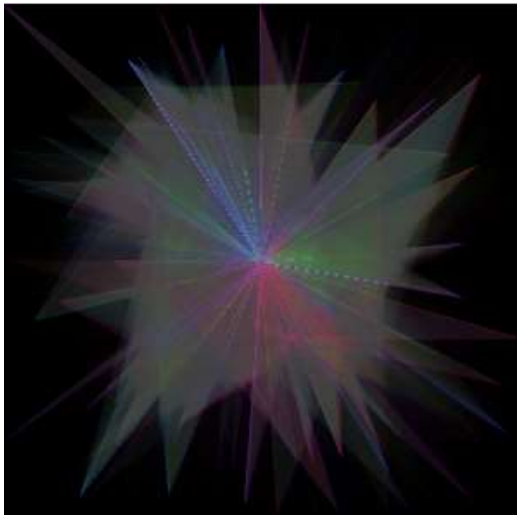
アレンジ例











回転体を増やしたいとき(1)

```
float angle = 0;
int nx,ny;
float xnoise = random(1);
float ynoise = random(1);
void setup() {
  size(300,300);
  noFill();
  stroke(0,10);
}
void draw() {
  translate(width/2,height/2);
  rotate(angle);
  nx = (int)(noise(xnoise) * 200);
  ny = (int)(noise(ynoise) * 200);
  line(0,0,nx,ny);
  xnoise += 0.01;
  ynoise += 0.01;
  angle += 0.005;
}
```

```
float angle = 0;
int nx,ny;
float xnoise = random(1);
float ynoise = random(1);
float angle2 = 0;
int nx2,ny2;
float xnoise2 = random(1);
float ynoise2 = random(1);

void setup() {
  size(300,300);
  noFill();
  stroke(0,10);
}
void draw() {
}
```

変数を一通り宣言

回転体を増やしたいとき(2)

```
void draw() {
  translate(width/2,height/2);
  rotate(angle);
  nx = (int)(noise(xnoise) * 200);
  ny = (int)(noise(ynoise) * 200);
  line(0,0,nx,ny);
  xnoise += 0.01;
  ynoise += 0.01;
  angle += 0.005;
  resetMatrix();
  translate(width/2,height/4*3);
  rotate(angle2);
  nx2 = (int)(noise(xnoise2) * 200);
  ny2 = (int)(noise(ynoise2) * 200);
  line(0,0,nx2,ny2);
  xnoise2 += 0.01;
  ynoise2 += 0.01;
  angle2 -= 0.005;
}
```

- 座標系を元に戻して
  - resetMatrix();
- 座標系を動かして
  - translate(width/2,height/4*3);
    - translate(150,450); 等でも
  - rotate(angle2);
    - 作った変数を使う
- 変数を使う処理を新しい変数でも追加
  - 書き間違えに注意
- 座標系については次ページから

translateとrotate(座標変換)

- 気になる人向け解説

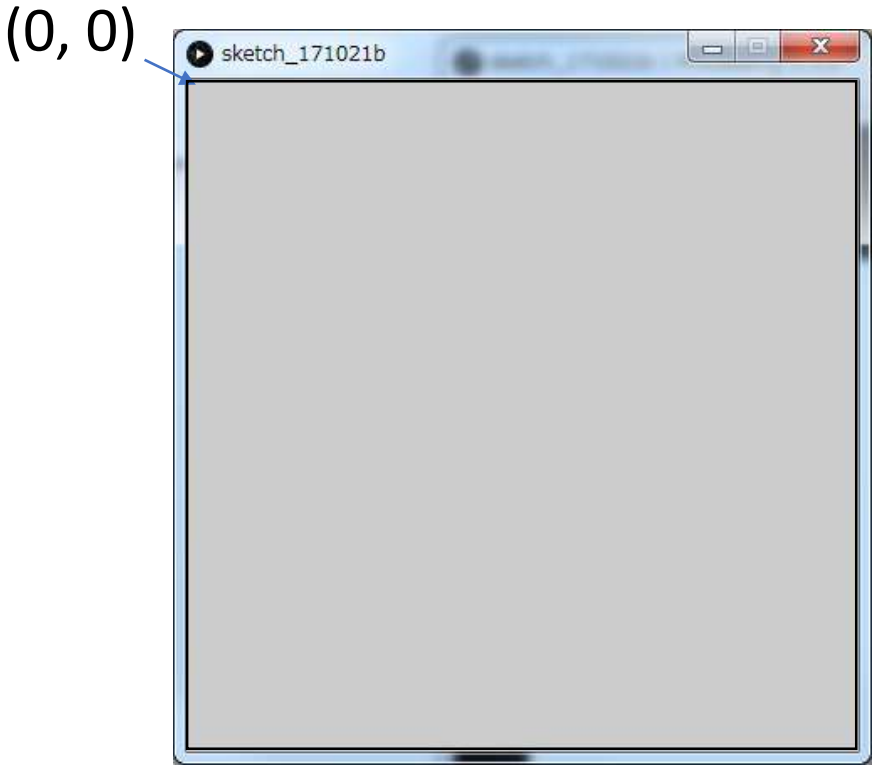
- 前回の変数を使った動きを入れてみようとしたら、
  思い通りに動かない、というときに参考にしましょう

- （これは動画でのみ説明）
  アニメーションなしだと説明も難しいので、
  なるべく動画を見てください

表示領域と描画キャンバス（座標系）

正しい呼び方

- 実行画面の座標と、描画キャンバスの座標が別々
  - 普段はどちらも実行画面の左上が (0, 0)



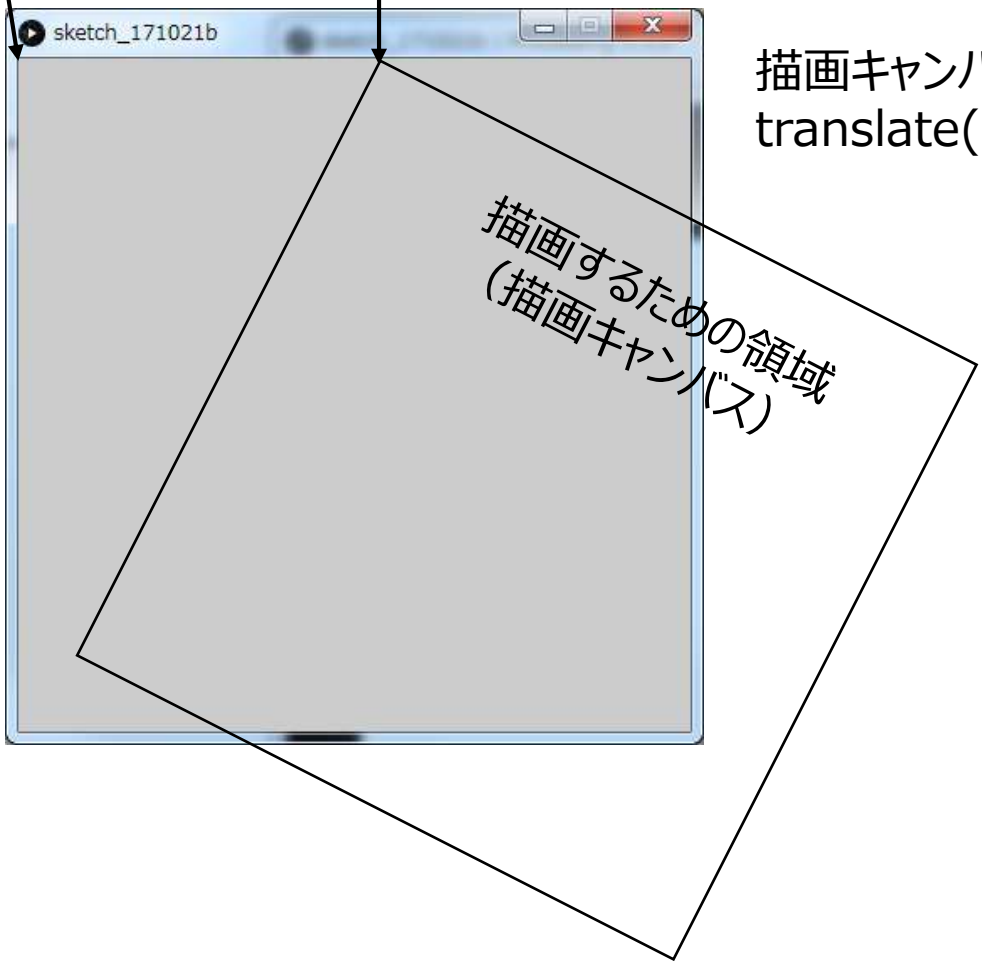
スクリーン座標系とローカル座標系

- スクリーン座標系
  - 左上を原点(0,0)とした座標系
- ローカル座標系
  - 座標変換によって移動・回転・変形した座標系

- スライドや資料での呼び方
  - スクリーン座標系　⇒　実行画面
  - ローカル座標系　⇒　描画キャンバス
  - イメージしやすそうな呼び方で説明する

スクリーン座標系の原点
(通常の(0,0)の位置)

ローカル座標系の原点
(移動した(0,0)の位置)



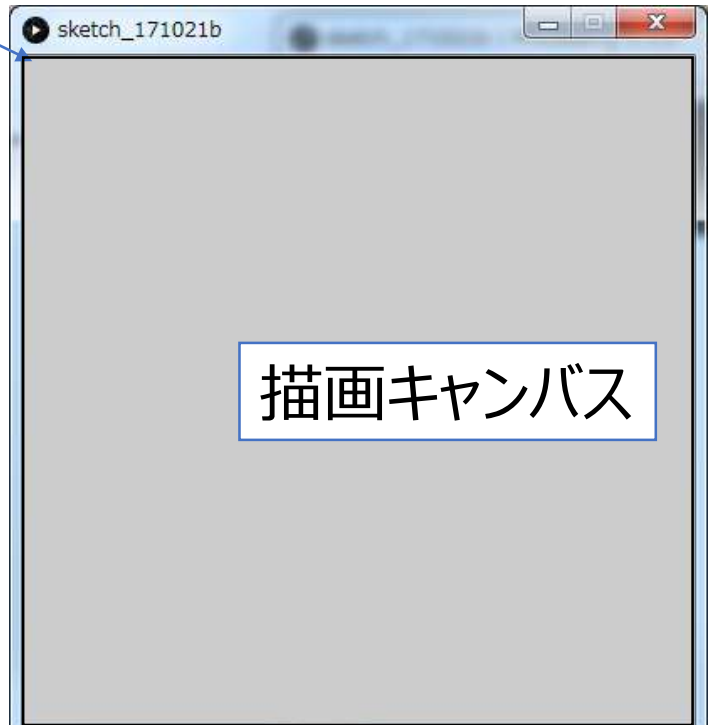
描画キャンバスは
translate()やrotate()で移動できる

座標変換：移動（translate)

- translate(x, y); の命令を実行すると、
  描画キャンバスを動かすことが出来る
  ＝　座標の原点が移動する

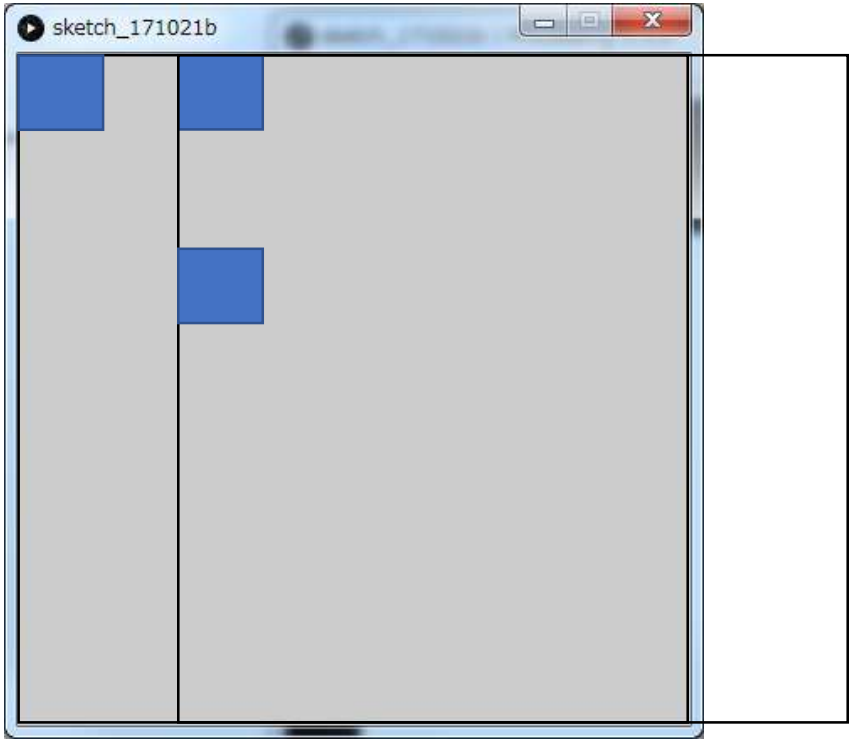
(0, 0)

translate(100,0);



座標変換後の図形描画

```
rect(0,0,10,10);
translate(100,0);
rect(0,0,10,10);
translate(0,100);
rect(0,0,10,10);
```

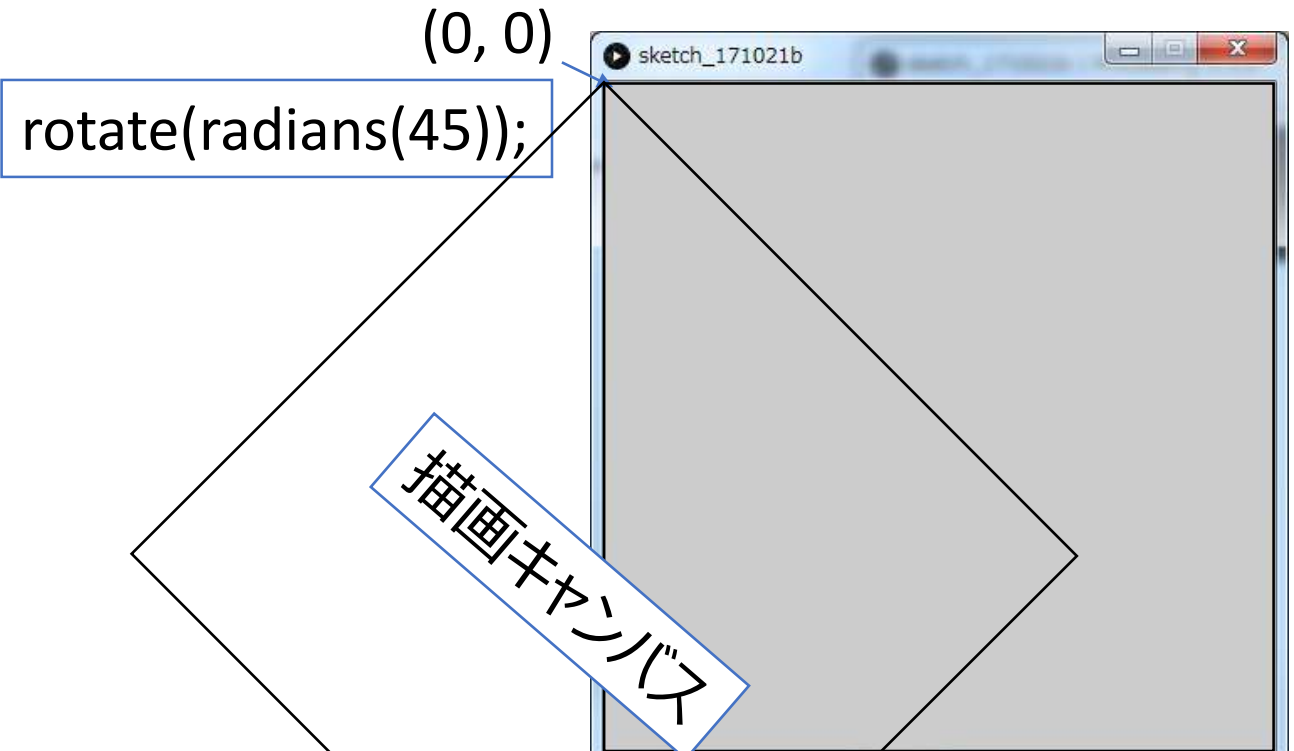


座標変換後にもう一度座標変換すると、その場所から更に移動する

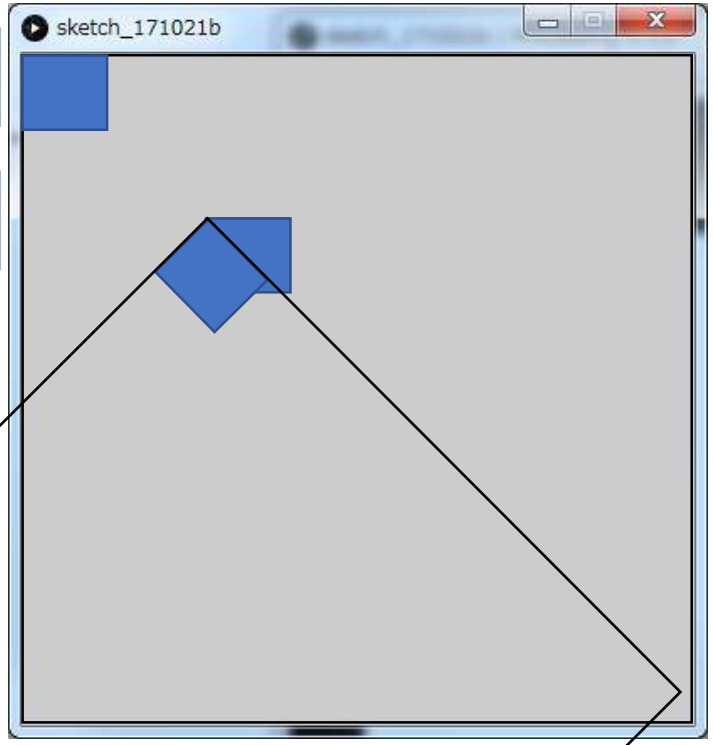
同じ座標でも、座標変換後は移動した描画キャンバス上の座標で描画される

座標変換：回転（rotate）

- rotate(angle); の命令を実行すると、描画キャンバスを回転させることが出来る
  - その時点の原点を中心に回転する



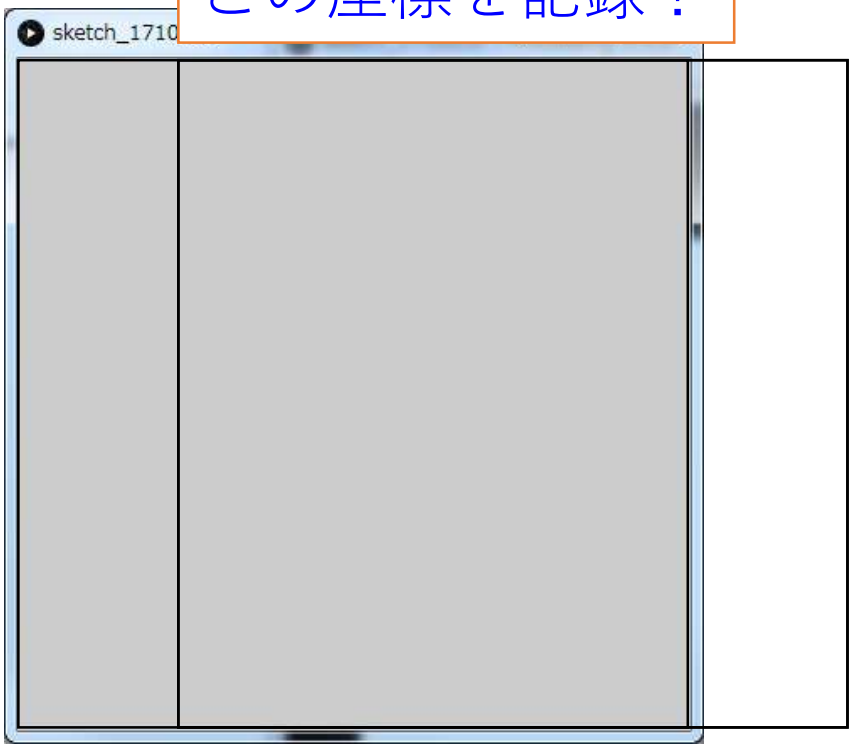
```
rect(0,0,10,10);
translate(100,100);
rect(0,0,10,10);
rotate(radians(45));
rect(0,0,10,10);
translate(100,0);
```



座標を回転させた後に座標を移動すると、回転後の座標に対して移動する

座標を記録する、取り出す

```
translate(100,0);
pushMatrix();
translate(0,100);
popMatrix();
```



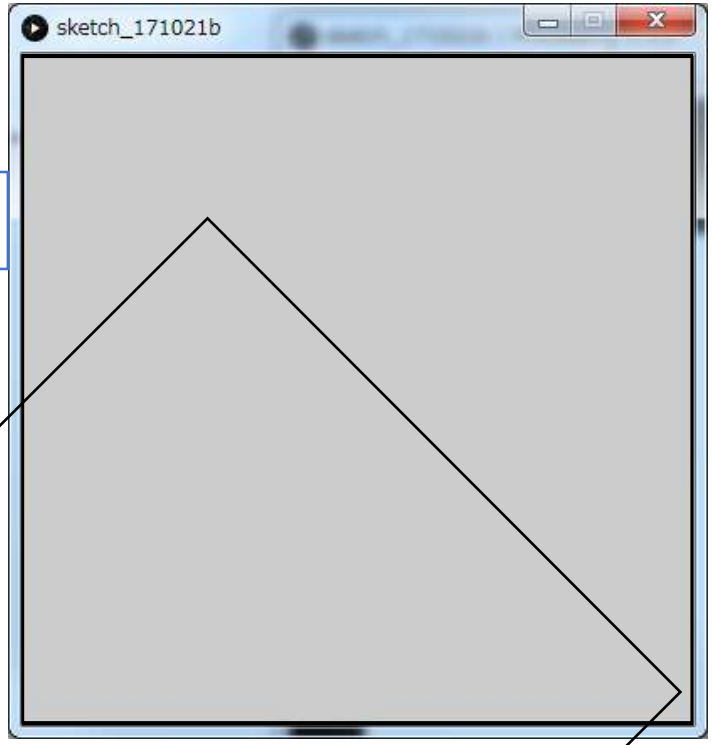
座標をリセットする：resetMatrix()

translate(100,100);

rotate(45);

translate(100,0);

resetMatrix();



resetMatrix(); を実行すると、移動回転させた座標が元に戻る

drawが1回終わる度に座標はリセットされる