コンピュータ基礎演習

第4回

理工学部 情報科学科 隅田 康明

sumida@ip.kyusan-u.ac.jp

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の授業共有ファイルにもアップロードしていますが、
    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/

レポートの遅れ提出

- この授業は、積み上げ式のため、
  欠席した回を飛ばすとついて来れなくなる場合がある
  - 前の回のレポートを利用する場合もある

- 欠席した回のレポートは飛ばさずに出すこと
  - 何度も何度も書いていますが、
    理由があれば遅れて提出でも減点はしないので、
    焦って追い付こうとして、欠席回を飛ばさないように
    - 細かいこと考えて悩む前に**まず相談**しましょう

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の書き方

[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） -- 20AA999 九産太郎 九州産業大学　芸術学部　○○学科 1年	用件 署名

相手に丁寧に対応して欲しいなら、自分も丁寧に！

演習の内容
Processingを用いたプログラミング

- Processingとは
  - プログラミングの統合開発環境の１つ
    - プログラミングの入門に適している
  - 視覚的な表現を(比較的)簡単に実現できる
    - 絵を描いたり、アニメーションを作れる
  - 描画に関するプログラムをシンプルに記述出来る
  - Webブラウザでも実行可能

プログラム・プログラミング

- プログラムとは
  - コンピュータに実行させる処理の記述

- プログラミングとは
  - プログラムを記述すること

- コンピュータで動くソフトウェアは、誰かが作ったプログラム
  - メモ帳、電卓、ワープロ、ゲーム、Webブラウザ…
  - スマートフォンのアプリも同じ

プログラミングを学ぶメリット

- 論理的に考える力が身につく
  - 論理的思考/ロジカルシンキング
  - 動かない原因を探り、論理的におかしい箇所を修正
- 問題を解決する能力を磨く
  - エラーを見つけて修正する
  - 解決する手段を調べる力（検索力）

Processingで出来ること

- 視覚化が必要になる分野で利用されている
  - デザイン・アート・ファッション・建築など

- デザイン・アート
  - CG(コンピュータ・グラフィックス)の作成
  - 静止画、アニメーション
- ゲーム
  - 簡単なゲームを作成できる
    - 高度なゲームになると、他の言語の出番

ジェネラティブ・アート

- コンピュータ等によって生成された芸術作品

- コンピュータで人間が絵を描く
  - 完成図をイメージ ⇒ イメージに近づくように描く

- ジェネラティブアート
  - 作品のコンセプトを決める
    ⇒ ランダム性を加える
    ⇒ 思いがけないような作品が「生成」される

第2回の「ランダム」を使ったアレンジ例が少し近い

ジェネラティブ・アート紹介

所謂プロの作品

- Life in 2050(マット・ピアソン, 2010)
  - https://vimeo.com/10924639
- マリウス・ワッツ
  - http://mariuswatz.com/category/works/

世界中のProcessingユーザの作品

- Open Processing

Processingで○○を作る

- アートを作る
  - 手書きでは出来ないような絵を描ける
    - 特定の規則がある図形等
  - 動きのある作品を作れる
    - アニメーション
    - マウスやキーボード、カメラと連動して動く
- ゲームを作る
  - 簡単な2DゲームならProcessingでも十分
    - 3Dゲームを作るには力不足

プログラミングはモノ作り

- 絵を描く、彫刻を彫る、模型を作る、料理を作る、曲を作る、映像を作る･･･
- プログラミングも同じ

- こんなものを作りたい！　　を実現する
  - 少しずつ完成イメージに近づけていく作業
  - 出来上がったときの達成感を体験しよう
  - 想像外の作品が出来上がることも

Processingで作られる作品の種類

- 静止画の描画：一枚の絵を作る（動かない）
  - 今回の内容の発展形
- アニメーション：絵を動かす、動かした軌跡を残して絵にする
- ブラッシングツール：絵を描くためのツールを作る
  - マウス等で操作して絵を描くためのブラシを作る
  - 2・3回のプログラムの発展形
- ゲーム：ゲームを作る
  - ブラウザ上でも動かせるためWeb系では需要がある
  - 3Dゲームなど高度な内容になると力不足
  - この授業では、簡単なゲーム作りを体験
    - ゲームは結構難しい・・・

今回のレポート

- Processingで静止画を作成
  - テーマは自由、意味のある形にすること
  - 詳細は資料をよく読むこと

- レポート提出
  - Moodle上でレポート(PowerPoint)提出

- ミニテスト
  - 今回は簡単な内容
  - 受験可能になるのは月曜日1限から
    - それ以前に受けられるようになっていたら受けてもいいです

レポート（PowerPoint）の注意

- 「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ⅰ、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

レポートの評価

- 今回からレポートの内容によって点数を付ける
  - 前回までは出していれば満点
    - ただし、遅れ提出や、メールの宛先、件名間違いなどを指摘されても直していないケースを除く
  - 採点基準
    - 1回～3回：1･2回は3点、3回は5点
    - 4回～10回
      - 基本点：3点（最低限の基準を満たしている）
      - 追加点：2点：基本点以外の工夫点1つにつき0.5点
        - チェックリスト外の工夫をレポートで説明、で評価
      - 最高で5点となる
    - 11回･12回：各回2点（プログラムの途中経過を提出）
      - 出せば2点、出さなければ0点

各回のレポートの確認

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)）

単位取得ライン

- 以下の条件を満たせば可(60%以上)となる
  - 毎回、最低基準を満たす：38点
    - チェックリストを埋めていけば最低基準は満たせる
    - 遅れ提出でも最低点は取れる
  - ミニテストで50%以上の正解率：5点
    - 何度解き直しても良いテスト(回答後に正解も出る)
    - 何度も受ければ誰でも全問正解できる
  - 制作課題で40点中20点を獲得
    - レポートを受け取る（採点する）最低基準
    - 20点未満になるようなレポートは出し直し
      - 余程の手抜きでなければ20点以上は取れる
  - これで合計63点：単位取得ライン＋3点余裕がある

ミニテスト

- Moodle上で実施

- 今回はテストのテスト
  - 問題は少なくても点数はつく
  - 何度でもやり直せるので、正解になるまで解きなおそう
    - 最後の受験の点数で評価します
  - 最終回までの小テストの点数を合計して、全体評点の10%として評価する
    - つまり、全問正解にしておけば10点になる
      - Moodle上の評価の点数と差が出る場合があるので注意

今回以降の演習の流れ（予定）

1. 資料を見る、（動画を見る）
   - プログラムに関する説明が毎回入る
2. 小テストを解く（何度でも解き直し可）
   - これはもう何回か後にするかも知れません
3. プログラムの雛形をコピー＆ペースト
   - 雛形の一部を変更してアレンジ
4. Power Pointでレポート作成
5. Moodleでレポートを提出

今回で、遠隔授業の流れをしっかり掴んでおこう！

プログラミングについての注意

- プログラミングは**最初は上手く行かなくて当たり前**です
  - 何度も失敗して、修正して、段々完成に近づけていきます
    - プログラムにエラーが出る、思ったとおりに動かない、は、日常茶飯事、ごく普通のこと
- 難しい内容も出てきますが、**最初は分からなくて当たり前**です
  - 試しに動かして、少し数値等を変えてみて、また試して、少し変えてみて、繰り返して理解していきます
  - 最初分からなくても、何となく動くものを作って行くうちに、だんだんと理解していくものです
- よく理解できない、自分にはプログラミングは向いていない…
  - そんなことないから、細かいことを気にする前に、プログラムを書いて、試してを繰り返しましょう
  - 最初は、**よく分かってないけど、書いたら動く、面白い！**でOK
    - 深く理解するのは、その後で

Processingを起動

- Processingを起動して、白紙の状態にしておく

- 新しいスケッチの作り方などは第３回を参照
  - 講義資料は講義HPを見ましょう
  - 操作が分からなければ第3回の演習動画を見ましょう

Processingの基本形

- まずはこれを書いてからプログラミングスタート
  - 絶対に書く決まり事と思っておけば良い

```
void setup() {


}

void draw() {


}
```

setupメソッド
{} の中に最初に1度だけ
実行する内容を記述する
動かさないならここだけ使えばOK

drawメソッド
{} の中に何度も繰り返し
実行する内容を記述する
アニメーションを作るときに使う

これを書かないと、図形を動かせない（静止画を書くだけなら、なくてもいい）

setupとdraw

- setup(){　} と draw(){　}は、プログラムの中に、それぞれ一つしか書いてはいけない
  - setupが2つある、drawが2つあるは、プログラムが動かない

- setup(){　}の中に、或いは draw(){　}の中に、setup(){　} や draw(){　} を書いてはいけない
  - { から　} までで一つの括りになる
  - { 波括弧 を閉じ忘れないように気を付けること
    - **カッコは開いたら必ず閉じる！！**

Processing最初の一歩

- 下記のプログラムを入力して実行

```
void setup(){
  size(400,400);
  line(0, 0, 100, 100);
}
void draw(){


}
```

線が一本引かれるのを確認したら次へ

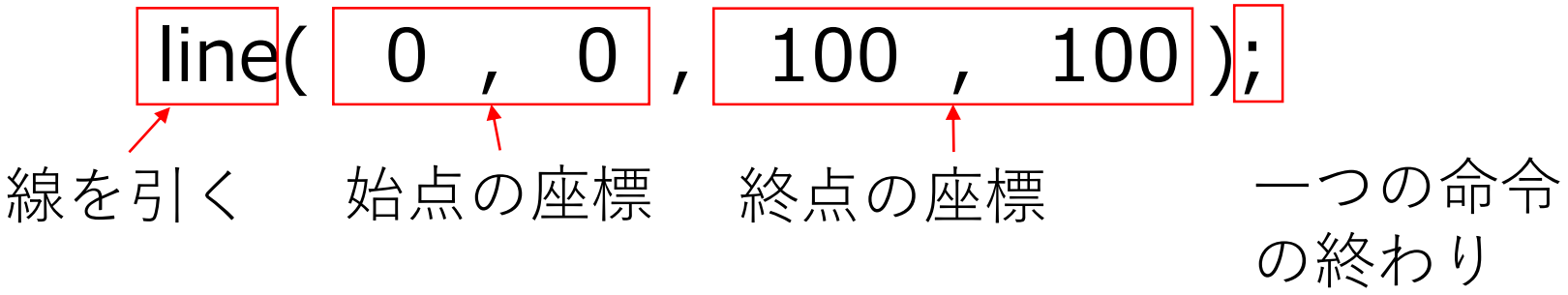
線が引かれなければ打ち間違っている、よく見直そう

プログラムの説明：size(400,400);

- size(横, 縦);
  - Processingを実行した時の、
    実行画面の大きさを設定する命令

- 全画面で描画したい場合(黒の余白が気になる場合)
  - iPhone：size(screen.width,screen.height);
  - Android：size(displayWidth,displayHeight);
    - PCでも同じだが、全画面表示だと大きすぎて作業しにくい場合があるので注意
  - iPhoneと他で命令の仕方が違う理由
    - iPhoneは裏ではjavaScriptで動いているから
      - 更に細かい話を聞きたい場合はメールで聞いてください。
        全体向けにするにはまだ早すぎる内容が多く含まれます。

今書いたプログラムの解説



- 意味：(0, 0) から (100,100)をつなぐ線を引く

- 一つの命令を書いたら、[;](セミコロン)を記述
  - 文章を。で区切るようなイメージ
- 今回使う命令は、全て”メソッド”という種類の命令

命令には、それぞれ意味があり、厳格にルールが決められている

他の図形も描いてみる

- line(0, 0, 100, 100);　の下に長方形を描いてみる

  line(0, 0, 100, 100);
  rect(200,200,100,50);

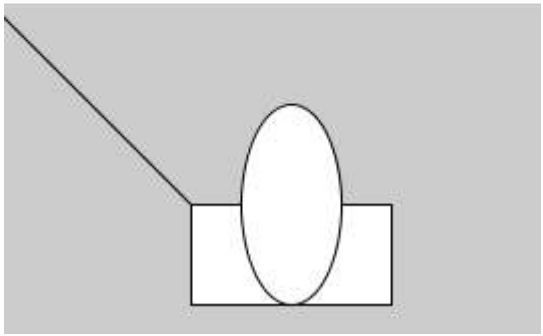
- rect(200,200,100,50);
  - rect(左上x,左上y,幅,高さ); は長方形を描く命令
  - (200,200)を左上に
    横100、縦50の大きさの長方形を描く
    - 縦の方が小さいので、横長の四角形になる

更に図形を追加してみる

- 円を追加
  line(0, 0, 200, 200);
  rect(200,200,100,50);
  ellipse(150,100,50,100);

```
sketch_200520a
1 void setup() {
2   size(400, 400);
3   line(0, 0, 100, 100);
4   rect(100, 100, 100, 50);
5   ellipse(150, 100, 50, 100);
6 }
7 void draw() {
```

sketch_200520a

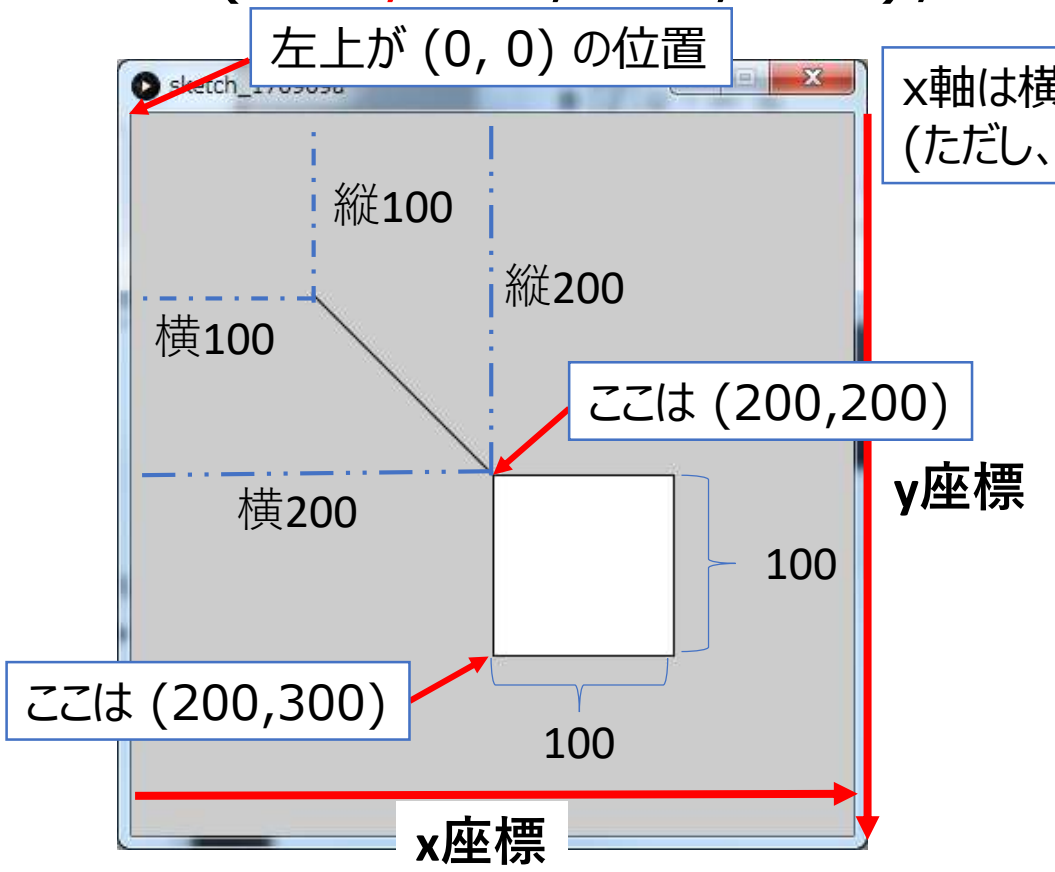


- ellipse(150,100,50,100);
  - ellipse(中心x,中心y,幅,高さ); は円を描く命令
  - (150,100)を左上に
    横50、縦100の大きさの長方形を描く
    - 横の方が小さいので、縦長の楕円になる

位置の指定：座標

```
line(100,100,200,200);
rect(200,200,100,100);
```

こんな命令をしたとする



x軸は横、y軸は縦 (数学のグラフと同じ)
(ただし、y の向きは逆(数値が大きいほど下)

一番左が x座標 0
一番上が y座標 0

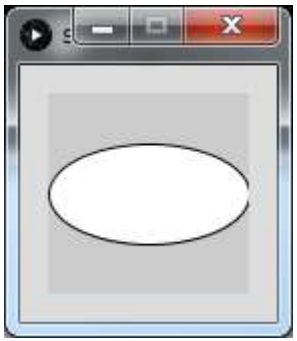
Processingでは、
位置は0から数え始める

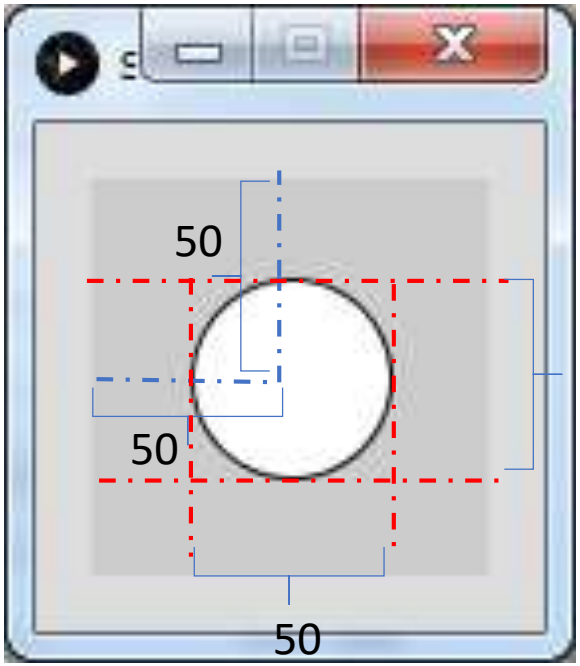
右に何歩、下に何歩と
移動してから描くイメージ

（例）楕円を描く命令

ellipse(中心のx,中心のy,幅,高さ);
ellipse(50,50,50,50);
(50,50)を中心に、幅が50、高さが50の円を描く

ellipse(50,50,100,50);
幅が100、高さが50の円＝楕円





命令にはそれぞれ意味がある

- line(100,100,100,100);
  rect(100,100,100,100);
  ellipse(100,100,100,100);
- カッコ内の数値は，(カンマ)で区切って4つずつ
  - 命令によって、いくつ値を書いていいかも決まっている
  - しかし、描かれる図形は全く違う
- line(始点x, 始点y, 終点x,終点y);
  rect(左上x,左上y,幅,高さ);
  ellipse(中心x,中心y,幅,高さ);

このプログラムは書かなくてもいい（既にあるellipseの命令を書き換えてみよう）

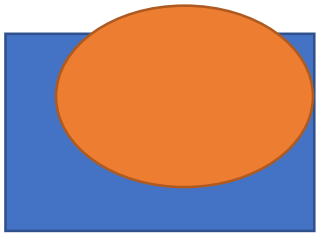
プログラムの逐次実行

- プログラムは１行目から順々に実行(例外もある)

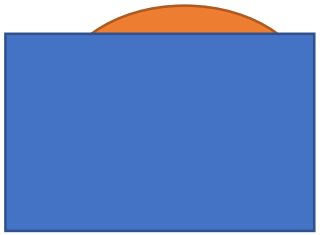
```
size(400,400);
line(100,100,200,200 );
ellipse(200,200,100,100);
rect(200,200,100,100);
```

画面の大きさを変える
線を引く
丸を描く
四角を描く

１.四角を描く
２.丸を描く



１.丸を描く
２.四角を描く



下に書いた命令の図形が、
上に重なっていく

上から順に、一つずつ実行していく

```
void setup(){
  size(400,400);
  line(0,0,50,50);
  rect(50,50,50,100);
}
```

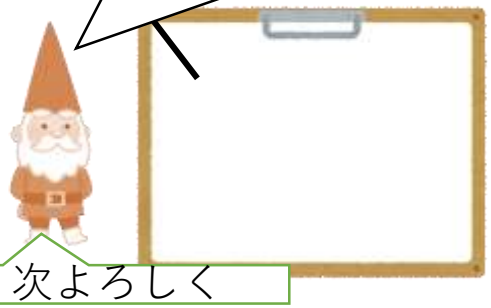
(400,400) のキャンバス作ったよ

次よろしく

sizeさん

幅と高さの2つの数値を受け取ると、
その幅高さの描画領域を作る

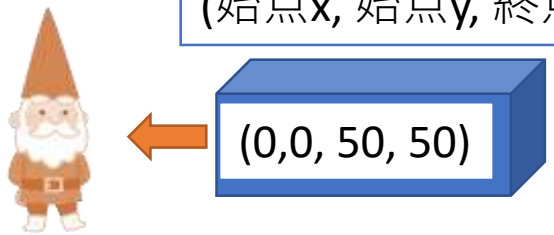
(0,0) から (50,50)に線を引いたよ



次よろしく

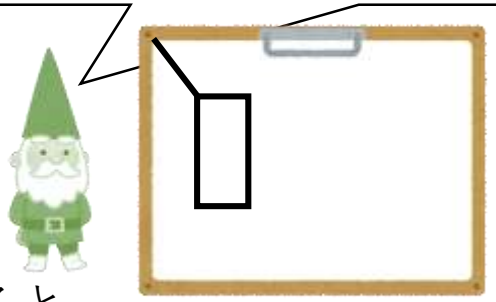
lineさん

(始点x, 始点y, 終点x,終点y)



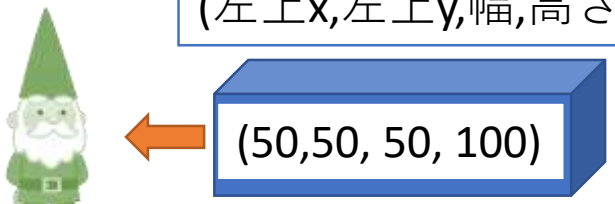
始点座標と終点座標を受け取ると、
その2点を結ぶ線を描いてくれる

(50,50)に幅50, 高さ100の四角を描いたよ



rectさん

(左上x,左上y,幅,高さ)



左上の頂点の座標と幅と高さを受け取ると、
左上座標を基点に、その幅と高さの長方形を描いてくれる

メソッドを書く(呼び出す)ときの決まり：ルールを守って命令を出そう

(50,50)の位置に
幅50高さ100か。
まかせとけ！

rectさん

(左上x,左上y,幅,高さ)

(50,50, 50, 100)

この位置にこのサイズの
四角を描いてください

命令の出し方を間違えると・・・

何を言っているのか
さっぱり分からない

rectさん

(50,50, 50)

ちょっと違うだけでも働いてくれない

この位置にこのサイズの
四角を描いてください

ちょっとくらい
省略しても伝わ
るだろう・・・

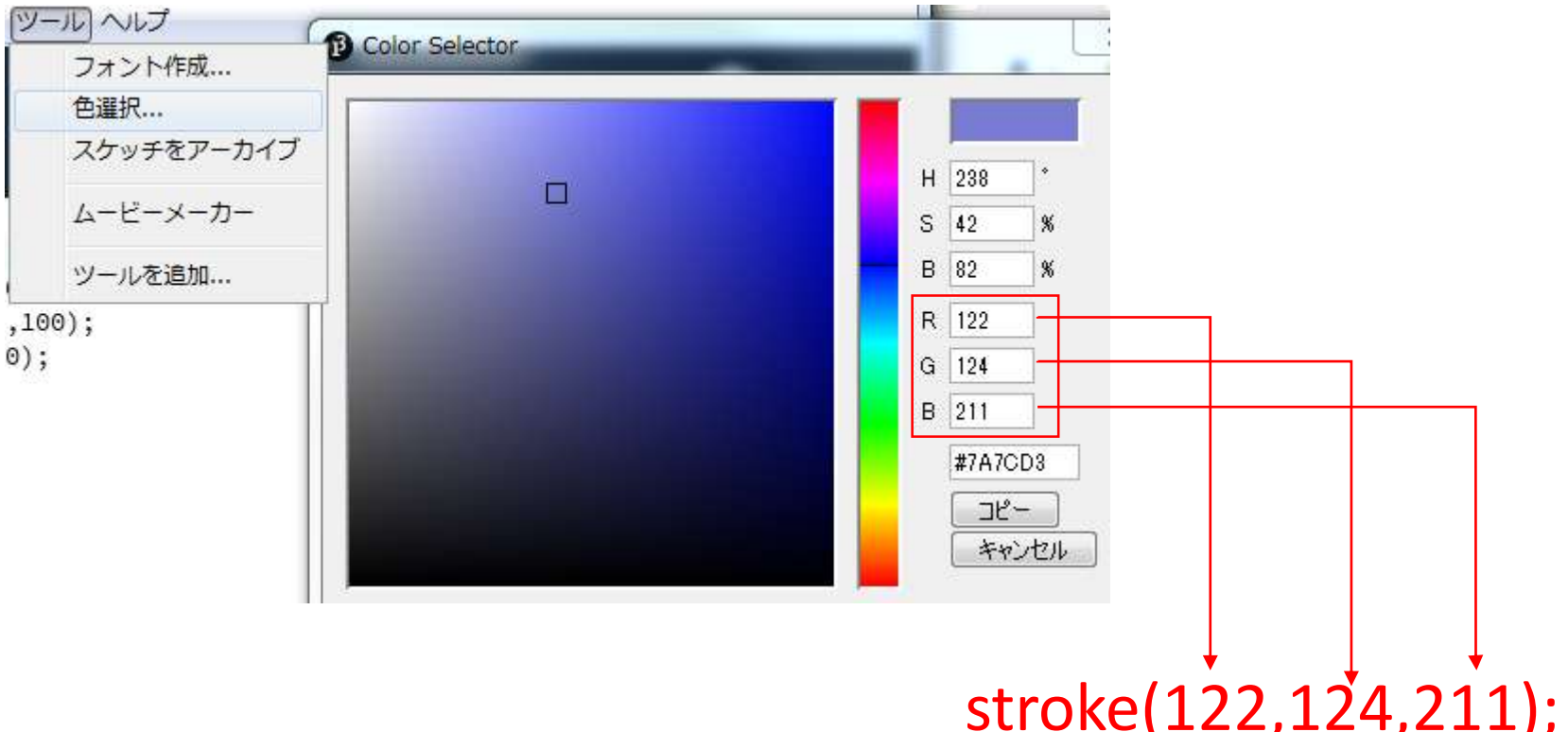
図形に色を付ける

- 何も指示しないと、線の色は黒色、塗りつぶしは白色になる
- 色を変えたければ、何色にするか**命令する**必要がある

- 線の色を変える命令：**stroke(赤, 緑, 青,透明度);**
  - それぞれ、0～255の数値で入力する
    - 透明度は数値が小さいほど薄い
- 塗りつぶしの色を変える命令：**fill(赤, 緑, 青,透明度);**
  - それぞれ、0～255の数値で入力する
    - 透明度は数値が小さいほど薄い
- 線を書かない：noStroke();
- 塗りつぶさない：noFill();

思い通りの色を作れない(PCの場合)
カラーセレクターで色を選ぶ

- RGBの数値だけだと、何色になるのか分かりにくい
- [ツール]メニューの「色選択…]



思い通りの色を作れない(その他の場合)
カラーセレクターで色を選ぶ

- オンライン上のツールを利用させてもらう
  - RGBとHSV・HSBの相互変換ツールと変換計算式 - PEKO STEP
    - https://www.peko-step.com/tool/hsvrgb.html

- 色の見本から選ぶ
  - **Webで使えるカラーネームと、そのカラーコード・RGB値一覧**
    - https://www.webcreatorbox.com/webinfo/color-name

図形を描く手順

1. strokeやfillで線、塗りつぶしの色を指定
2. 図形を描画
3. この繰り返し

```
stroke(255,0,0,50);   //半透明の赤色のペンを持つ
fill(0,0,255,255); //青色の塗りつぶしブラシを持つ
rect(10,10,100,50); //長方形を描く
```

他の色の指定方法

- Processingでの色の指定方法
  - RGBモードで指定：光の三原色 (赤,緑,青)で指定
    - デフォルトではこのモード。コンピュータの世界では、色はRGBモードモードが標準(ディスプレイがRGB)
      - 0～255で、各色の強さを指定
  - HSBモードで指定：色相(H)、彩度(S)、輝度(B)
    - ドロー系ソフトで絵を描く人は馴染みがある？
    - HSVやHSL(厳密には違う)などの呼び方も
    - 直感的に色を作りやすい
      - H：0～360で赤～黄～緑～シアン～青～紫～赤で１周する
      - S：0～100で、大きいほど鮮やかな色に
      - B：0～100で、大きいほど明るい色に
  - カラーコードで指定
    - HTMLではお馴染みの方法(考え方はRGBと同じ)
      - 色を変化させられないので、この授業での出番はない。紹介のみ。

HSBモードでの色指定 （今回は基本的にはRGBで書けばいい）

- **colorMode(HSB,360,100,100,100);** の命令を入れると、色の指定がHSBモードになる
  - colorModeは最後に実行したモードになる

```
colorMode(HSB,360,100,100,100);
fill(0,100,100);  //HSBでは赤色
ellipse(0, 0, 100, 100); //赤色の円
colorMode(RGB,255,255,255,100);
fill(0,100,100);  //RGBでは暗い緑色
ellipse(0, 0, 100, 100); //暗い緑の円
```

同じfill(0,100,100); でも全く違う色になる。
どちらのモードで色を指定しているのかを把握しておくこと

その他の命令（メソッド）

- Processingには、
  ここまでに紹介した以外にも沢山の命令がある
- 次のページに、よく使う【図形を描く命令】を載せておくので、
  - 命令を実際に書いて、
  - 数値を変えるとどう変わるのか確かめながら、
  - 命令の意味を理解していこう

- 打ち間違い、カッコの中の数値の数、に特に気をつける

手を動かして、書いて、実行してみて、理解する。
最初は上手く行かなくてもいい、まず試してみよう！

命令の意味	命令文	例
HSBモードにする		
線の色を設定する	stroke(赤,緑,青);	stroke(255,0,0);
透明度も設定する	stroke(赤,緑,青,透明度);	stroke(255,0,0,128);
線の太さを設定する	strokeWeight(線の太さ);	strokeWeight(5);
塗り潰しの色を設定	fill(赤,緑,青);	fill(0,0,255);
透明度も設定する	fill(赤,緑,青,透明度);	fill(0,0,255,128);
線を引かない	noStroke();	
塗り潰さない	noFill();	
線を引く	line(始点x, 始点y, 終点x, 終点y);	line(100, 100, 200, 200);
長方形を描く	rect(左上のx,左上のy, 幅, 高さ);	rect(100,100,50,80);
三角形を描く	triangle(x1,y1,x2,y2,x3,y3);	triangle(10,50,30,40,70,90);
四角形を描く	quad(x1, y1, x2, y2, x3, y3, x4, y4);	quad(30, 30, 80, 20, 70, 60, 30, 80);
円を描く	ellipse(中心のx,中心のy, 幅, 高さ);	ellipse(100, 100, 50, 80);
円弧を描く	arc(中心x,中心y,幅,高さ,開始角,終了角)	arc(100,100,50,50,0,PI/2);
文字を書く	text("書きたい文字列",左上のx,左上のy)	text("Hellow",100,50)
文字の大きさを設定	textSize(フォントサイズ);	textSize(24);
点を打つ	point(x座標,y座標);	point(10,10);

プログラムを作っていく上で

- 細かく実行して、動くことを確かめる
  - 1行書いたら実行して確かめる
  - 1行変更したら実行して確かめる

- 少し書いたらすぐ実行して確かめる
  - よくある悪い例
    - 一気に10行くらい書いて実行したら、
      エラーが沢山出てどこが悪いのか分からなくなった

- 細かく、細かく、実行してエラーなく動くか確認していこう

今回のレポート

- Processingで静止画を描き、レポートにまとめて提出
  - 静止画のテーマは自由、ただし意味のある形にすること

- 最低基準（チェックリスト参照）を満たせば3点
- 加点項目(正確には加点とは違うが)
  - 取り組めば＋0.5ずつなどで評点アップ

- 高度な内容（ここまでしなくても満点は取れる）
  - 物足りない学生向けの内容。
  - やる気があって、余裕がある場合にのみ挑戦しましょう。

レポートチェックリスト（第4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で実行した
- プログラミングを行い、静止画を作成した(最低基準)
  - エラーなく実行できた
  - 意味のある形になっている
  - 図形を描く命令が5個以上ある(線や点でも可)
  - 図形の色が3色以上ある
- 実行結果のスクリーンショットを保存した
- 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

加点項目(最低基準点より上の点)

- 下記をすれば最低基準点にプラス（最高5点）
  - 命令表にしか載っていない命令を使った
    - triangle, arc, textなど
    - 1つにつき+0.5点：3個まで
  - 色付き図形の数が5個以上
    - これは1個増えるごとに0.5点ではない
  - 命令表にもない命令を使った：1つにつき+0.5点
    - 自分で新しい命令を調べて使った
    - アレンジ例のfor文を使ってみた(後のページを参照)
  - 前回のプログラムと合体させた：+0.5点
    - 難易度が高い内容。やってみたい人だけ。
    - 詳しくは後のページを参照

高度なアレンジ

これ以降のスライドは、やる気があって物足りない学生向けの内容
（ここまでしなくても満点は取れる）

- 挑戦するなら、慎重に。
- まずレポートを一度出してから、くらいが丁度いい。
  - 2つリンクを送っても、ファイルをアップロードしてもいいですよ

線を並べる

- 繰り返し文を使うと、線や図形を簡単に沢山並べられる
  - 詳しい説明は後の回（第7回あたり）
    - 今は、よく分からないけど、数値を変えたら描けたで十分

縦に線を10本描く

```
for(int i = 1; i <= 10; i++){
  line(i*40, 0, i*40, 400);
}
```

横に線を20本描く

```
for(int i = 1; i <= 20; i++){
  line(0,i*20, 400, i*20);
}
```

□の数値を変えて色々試してみよう

繰り返し(for文)の構文

```
for(ループ変数の初期化; 条件判定; 更新) {
  繰り返したい処理;
}
```

- ループ変数の初期化;
  - 繰り返し条件に使う変数の宣言と初期化を行う
- 条件判定;
  - 繰り返しを続ける条件を書く
- 更新
  - 変数の値を変化させる処理
    - 基本的にはループ変数を変化させる

for文の例

```
for (int i = 0; i <= 5; i++) {
  rect(i*20, 10, 20, 20);
}
```

- ループ変数の初期化;
  `int i = 0;` 整数型の変数 i を初期値 1 で宣言
- 条件判定;
  `i <= 5;` i が 5 以下の間繰り返す
- 更新
  `i++` i に 1 を加算する (1ずつ増やす)

`i+=2` `i = i - 5` の書き方でもOK

前回のプログラムと合体

- 自分で作った静止画を背景にして、マウスで絵を描ける

```
void setup() {
  size(400, 400);
  line(0, 0, 100, 100);
  fill(0, 100, 100);
  rect(100, 100, 100, 50);
}
void draw() {
  float d = dist(mouseX, mouseY, pmouseX, pmouseY);
  ellipse(mouseX, mouseY, d, d);
}
```

ここで「背景」を描く

「背景」部分が今回のレポートのメイン
ここで手を抜くと点数が下がるので、
まず、静止画をしっかり作ること

前回のプログラムのdraw(){ … }の中身をコピーして、貼り付ける
貼り付ける場所を間違えると、エラーで動かなくなるので要注意

アニメーションの仕組み

- 一見動いているように見えるが、
  実際には静止画を連続で表示している
  - fps: 1 秒間に何回描画するかを示す単位
  - frameRate(fps); の命令でfpsを指定出来る

- Processingでのアニメーション描画
  1. drawを実行し、静止画を描画
  2. 画面全体を塗り潰す
  3. drawを実行し、静止画を描画
  - これを繰り返すことで絵が動いているように見せる
    - 動いているように見えるだけで、実際には瞬間移動