

プログラミング入門 Processingプログラミング 第6回

火曜3限（建築学部）
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、課題プログラムを作成するため
 - 課題プログラムは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

この授業は当面遠隔授業を継続します

- パソコン教室は、感染リスクが高くなる場所です
 - 室内に多くの学生が集まる
 - 教室に入室人数制限がかかっている
 - そもそも、大学の規則からして全員集めての実施は難しい
 - 不特定多数の人が触るPCを使う
 - 窓も1面しかなく、換気も不十分になる
- 遠隔でも出来る授業は、遠隔で実施していきます
 - 感染防止対策が不要になったら、別ですが
- 対面での質問受付などは、状況を見ながら検討
 - 仮に実施出来たとしても、人数制限は必須になります
 - 事前予約制などにしないと無理
- まだ予断を許さない状況なので、
それぞれで感染防止対策はしっかりしていきましょう

大学の自習室等について

- 6月1日から、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/ppt3/>

- Moodleに移行しました
 - Moodleに行けば全部あります

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	提出すると「評定のために提出済み」になる
評定ステータス	未評定	採点されると、「評定済み」になる
終了日時	2020年 05月 25日(月曜日) 00:00	
残り時間	4 日 2 時間	採点 = 教員（私）が手動で点数を付ける
最終更新日時	2020年 05月 20日(水曜日) 21:19	

← ← ← ← ←

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

提出回数	これは 1 回目の提出です。
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 22日(金曜日) 18:00
残り時間	課題は 2 日 18 時間 遅く 提出されました。
最終更新日時	2020年 05月 25日(月曜日) 12:08
オンラインテキスト	+ https://ksuemail-my.sharepoint.com/:p/g/personal/p-417646_r_u_ac_jp/EVqEBqdsMQFEspKHNGa5seMBocK0sXReYFE4S_nXPWe=0NlehS
提出コメント	+ コメント (0) 感想などがあればここに書いて下さい。 (メール提出やめたら感想が減って寂しい教員より) コメントを保存する キャンセル 提出を編集する あなたはまだ提出に変更を加えることができます。

- 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）

今回の授業内容

- メソッドを自分で作る
 - オリジナル図形を作って動かす

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

[illegible]

字が読めないほど小さい（問題なし）

Moodleからファイルをダウンロード→アプリ版のPowerPointで開く
→ 全選択してコピー → Processingに貼り付け で動けばOK

「プログラム」スライド、駄目な例

```
sketch_12_3D_enkei
1 float angle = 0;
2
3 void setup() {
4   size(1000, 1000, P3D);
5   colorMode(HSB, 360, 100, 100, 100);
6 }
7
8 void draw() {
9   background(255);
10  translate(width/2, 0);
11  scale(0.5);
12  for (int x = 0; x < 10; x++) {
13    translate(100, 100);
14    rotateY(radians(angle));
15    angle += 0.5;
16    enkeiMethod(0, 0);
17    rotateY(radians(30));
18    enkeiMethod2(0, 0);
19    rotateY(radians(30));
20    enkeiMethod3(0, 0);
21    rotateY(radians(30));
22    enkeiMethod3(0, 0);
23    rotateY(radians(30));
24    roseCurve(0, 0, 360);
25  }
26 }
27
28
29
30
31
32 void enkeiMethod(float x, float y) { //メソッドの名前は分かり
33   pushMatrix(); //現時点の座標系を保存
34   translate(x, y);
```

- 画像として貼り付けている
- これは点数を付けません。
出し直しを指示しているはずですが。
 - 出し直さなければ0点になります。
 - 最終回までに出し直せば、減点もなしで受け付けますが。

「プログラム」スライド、駄目な例 (分割してしまっているケース)

```
float angle = 0;

void setup() {
  size(1000, 1000, P3D);
  colorMode(HSB, 360, 100, 100, 100);
}

void draw() {
  background(255);
  translate(width/2, 0);
  scale(0.5);
  for (int x = 0; x < 10; x++) {
    translate(100,100);
    rotateY(radians(angle));
    angle += 0.5;
    enkeiMethod(0, 0);
    rotateY(radians(30));
    enkeiMethod2(0, 0);
    rotateY(radians(30));
    enkeiMethod3(0, 0);
    rotateY(radians(30));
    enkeiMethod3(0, 0);
    rotateY(radians(30));
    roseCurve(0, 0, 360);
  }
}
```

```
void enkeiMethod(float x, float y) { //メソッドの名前は分かりやすい名前にする
  pushMatrix(); //現時点の座標系を保存
  translate(x, y);
  float size = 50;
  for (int i= 0; i< 10; i++) { //10回繰り返す
    stroke(100, 100, 100, 50);
    fill(50, 100, 100, 50);
    rect(size, size, 10, 10);
    ellipse(size, size, 10, 10);
    rotate(radians(36)); //360度/繰り返し回数になるようにすると円形になる
  }
  popMatrix(); //保存した座標系に戻す
}
```

```
void enkeiMethod2(float x, float y) { //メソッドの名前は分かりやすい名前にする
  pushMatrix(); //現時点の座標系を保存
  translate(x, y);
  float size = 80;
  for (int i= 0; i< 10; i++) { //10回繰り返す
    stroke(300, 100, 100, 50);
    fill(200, 100, 100, 50);
    if (i % 2 == 0) { //もしも、iを2で割った余りが0なら (iが偶数なら)
      rect(size, size, 10, 10);
    } else { //そうでないなら (iが奇数なら)
      ellipse(size, size, 10, 10);
    }
    rotate(radians(36)); //360度/繰り返し回数になるようにすると円形になる
  }
  popMatrix(); //保存した座標系に戻す
}
```

減点まではしないけど、なるべく止めて下さい。
(これをする人が増えたら、この形式は禁止にします)

プログラム：2枚目

```
void enkeiMethod3(float x, float y) { //メソッドの名前は分かりやすい名前にする

    pushMatrix(); //現時点の座標系を保存
    translate(x, y);
    float size = 20;
    for (int i = 0; i < 10; i++) { //36回繰り返す
        stroke(i*50, 100, 100, 50);
        fill(i*10, 100, 100, 50);
        if (i % 2 == 0) { //もしも、iを2で割った余りが0なら (iが偶数なら)
            rect(size, size, 10, 10);
        } else { //そうでないなら (iが奇数なら)
            ellipse(size/2, size/2, 10, 10);
        }
        rotate(radians(36)); //360度/繰り返し回数になるようにすると円形になる
    }
    popMatrix(); //保存した座標系に戻す
}
```

```
void roseCurve(float cx, float cy, int loop) {
    pushMatrix();
    translate(cx, cy);
    strokeWeight(10);
    float r1 = 0, r2 = 0;
    for (int i = 0; i < loop; i++) {
        float a = sin(radians(r2)) * 0.8 + 1;
        float x = a * cos(radians(r1)) * 50;
        float y = a * sin(radians(r1)) * 50;
        ellipse(x, y, 1, 1);
        r1 += 1;
        r2 += 7;
    }
    strokeWeight(1);
    popMatrix();
}
```

コピペの時に面倒ですよ（私も面倒です = 採点が遅れます）。

どうしても、プログラムスライドを綺麗に整形したい人は・・・（やっても点数はあげません）

- **二度手間になるし、評点も上がらないので、スライドが汚くて我慢ならない！！という場合に限る**
- オンラインテキストにもプログラムを貼り付けておく
 - これでもOK。
 - プログラムスライドもちゃんと作っておくこと
 - 作品とプログラムが一つのPowerPointファイルで完結していると、後から見直すのが簡単
 - あのプログラムどこ行ったっけ・・・はありがち、ファイル名付けててもすぐ忘れます
 - PowerPointなら、エクスプローラー（MacならFinder）のプレビュー機能で絵からプログラムを探しやすくなる
 - 今はスマホで作業していても、そのうちPCを買ったときにプログラムを移せる
 - ちゃんと理由があって指示しているので、指示に従ってレポートを出すこと
 - 気になるならメールで聞いて下さい

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おけば良い

```
void setup() {  
  
}
```

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

```
void draw() {  
  
}
```

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。アニメーションでは必須！！

背景の書き方の違い

- `setup(){ }` 内に背景のプログラムを書く場合
 - 背景は**一度しか描かれない**
 - 最初から背景が描かれている紙に絵を描いていく
 - 完全に背景扱い（背景の上に絵を書いていく）
- `draw(){ }`内に背景のプログラムを書く場合
 - 背景が**毎回描かれる**
 - 常に上書きされる背景の上で絵を描く
 - 実際には背景もアニメーションの一部になっている
 - `draw(){ }` 内の一行目に `background()` を書くと、軌跡を残さないプログラムになる

setup と drawについて注意

- setupメソッド
 - プログラム実行時に最初に実行する
 - 最初に1度だけ実行する命令を記述する場所
- drawメソッド
 - 何度も繰り返し実行する命令を記述する場所
 - setupの後に実行される

drawの{ }内の命令は何度も実行される
⇒ 動きのあるプログラムが作れる！

今回の、演習の進め方

- まず、資料を読む、必要なら動画も見る
- 資料に書いてあるプログラムを書いて動かしてみる
- ミニテスト受験1回目
- レポート作成
 - 今回、ひな形はなし
- レポートに取り組む
- レポート提出、ミニテスト受験（2回目）
 - ミニテストは何回も受けて理解を深めましょう

メソッド

- メソッドの構造
 - メソッドの名前(引数1,引数2,...);
size(400,400);
()の中に書くものは命令によって違う
 - 引数がないメソッドもある
noStroke();
noFill();
- size(400,400); や size(400,400); のように、
メソッドをプログラムに書いて実行することを、
メソッドを呼び出すという

メソッドを自分で作る

- メソッド：複数の処理をまとめて 1 つの命令にする機能
 - 関数ともいう
 - メソッドの名前(値1,値2,...); の構文
 - `setup()`、`draw()`、`line()`、`rect()`、`ellipse()`など
- 複数の命令を 1 つにまとめたメソッドを自分で作れる

メソッド：処理をまとめて名前をつけた物

青ペンを持つ

三角形を描く

円を描く

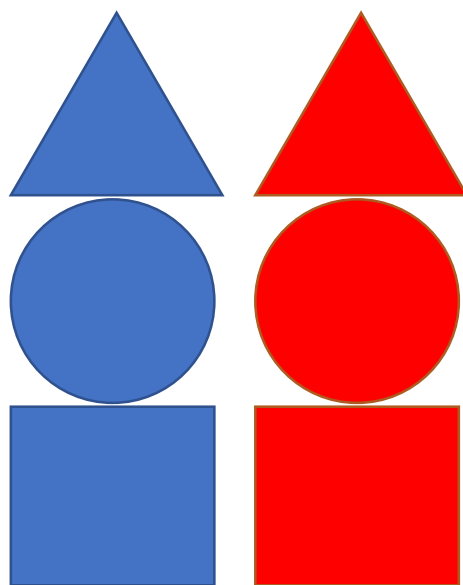
四角形を描く

赤ペンを持つ

三角形を描く

円を描く

四角形を描く



```
void oden() {  
    三角形を描く  
    円を描く  
    四角形を描く  
}
```

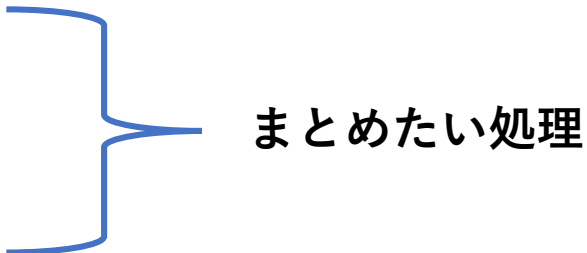
```
fill(0,0,255);  
oden();  
fill(255,0,0);  
oden();
```

色を変える度に同じ命令を書かなくてもよくなる

メソッドの作り方、呼び出し方

- メソッドの作り方

```
void メソッド名(引数1,引数2,引数3,...) {  
    処理 1 ;  
    処理 2 ;  
    処理 3 ;  
}
```



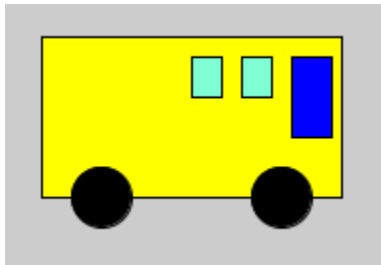
まとめたい処理

- メソッドの呼び出し方

- setupやdrawの中で、
メソッド名(引数1,引数2,引数3,...);

オリジナルの図形を描くメソッドを作る

- 大きさの変更は今回はなし
 - 挑戦したい人はしましょう
- 位置だけ指定出来るメソッドを作る
- 例として、こんなのを作ってみる



- 後のゲーム作り回のキャラクターにするので、そのつもりでデザインすること

まずは、アニメーションの基本形

- setupとdraw

```
void setup(){  
  size(400,400);  
}  
void draw(){  
  background(255);  
}
```

- この構造でアニメーションを動かすモードを、ダイナミックモードと呼ぶ
 - この授業では使わないが、setupとdrawなしで書くスタティックモードもある
 - スタティックモードは、静止画しか描けない

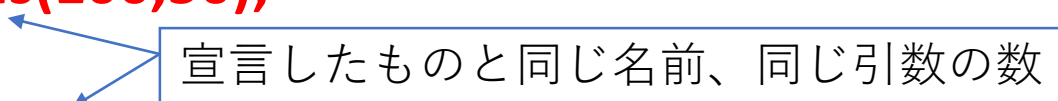
メソッドを宣言

```
void setup(){  
  size(400,400);  
}  
void draw(){  
  background(255);  
}  
void bus(float x, float y){  
  
}
```

- メソッドの名前は自分で作る図形に合わせて変えること

メソッド呼び出し

```
void setup(){  
  size(400,400);  
}  
void draw(){  
  background(255);  
  bus(100,50);  
}  
void bus(float x, float y){  
  
}
```



宣言したものと同じ名前、同じ引数の数

まだ何も書かれない(何も書いていないから)

図形を描いていく

- まず、中心点を描画（ガイドとして描き、後で消す）

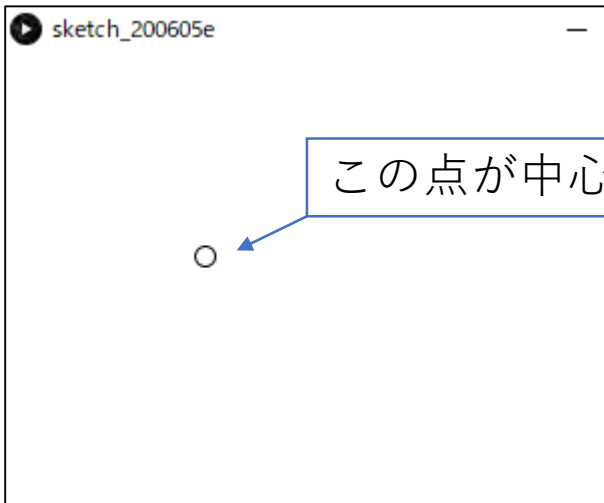
```
void bus(float x, float y){
```

中心点 (x, y) を変えないこと

```
ellipse(x, y, 10, 10);
```

```
}
```

中心点 (x, y) の座標は、メソッド呼び出しで入力した引数,つまり(100,50)



この点が中心になるように設計する

中心点を描くだけのメソッド（今の状態）

数値を2つ貰ったら、その場所に図形を描いてくれる

busさん

(中心x, 中心y)



(100, 50)

描いといて。
(x, y) の中身は自分で見てね

(x, y, 10, 10)



ellipseさん

(中心x, 中心y, 幅, 高さ)



(x, y)を中心に、幅10,高さ10・・・
xには100が、yには50が入ってるから
(100, 50) の位置に幅10,高さ10の丸を描きます

busの設計図



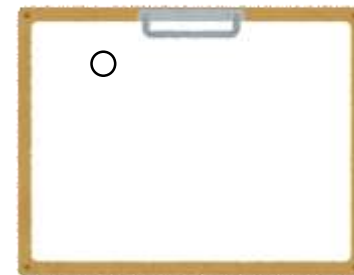
```
ellipse(x, y, 10, 10);
```



メソッド内だけで
共有で使える変数になる
(メソッドの外では使えない)

何の図形を描くかは、メソッド内の命令によって変わる

busさんの場合は、(x, y)の位置に幅10,高さ10の円を描く



設計図の中の命令が増えても、基本は同じ

中心点を中心に設計(1)

- 中心点が中心になるように図形を配置していく

```
void bus(float x, float y){  
  rect(x, y, 100, 50);  
  ellipse(x, y, 10, 10);  
}
```

中心点は最後に描く

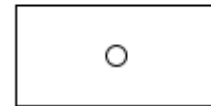
x座標、y座標を指示する引数には、必ずxとyを使うこと



```
void bus(float x, float y){  
  ellipse(x, y, 10, 10);  
  rect(x-50, y-25, 100, 50);  
}
```

上手くいかないので修正

sketch_200605e



中心点が中心になったら、次の図形へ

中心点を中心に設計(2)

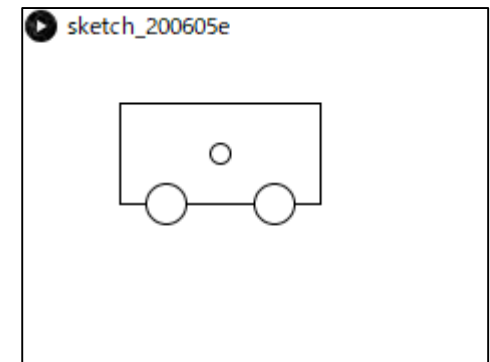
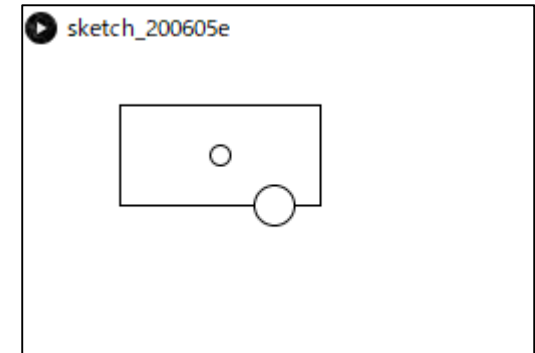
- 微調整しながら図形を配置していく

```
void bus(float x, float y){  
  rect(x-50, y-25, 100, 50);  
  ellipse(x+27, y+25, 20, 20);  
  ellipse(x, y, 10, 10);  
}
```

中心点は最後に描く

x座標、y座標を指示する引数には、必ずx と y を使う

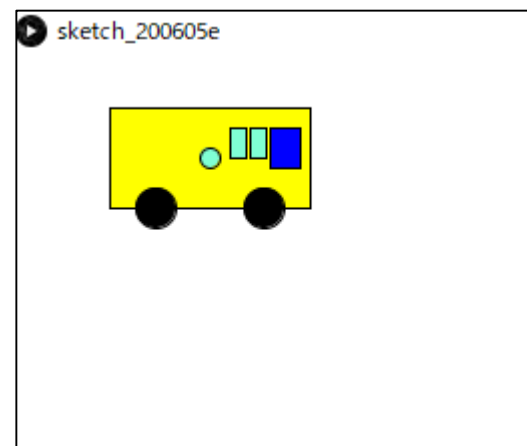
```
void bus(float x, float y){  
  rect(x-50, y-25, 100, 50);  
  ellipse(x+27, y+25, 20, 20);  
  ellipse(x-27, y+25, 20, 20);  
  ellipse(x, y, 10, 10);  
}
```



色付き図形を配置していく

- 色を付け、図形を配置、を繰り返して動かす図形を作る

```
void bus(float x, float y) {  
  fill(255, 255, 0);  
  rect(x-50, y-25, 100, 50);  
  fill(0, 0, 0);  
  ellipse(x+27, y+25, 20, 20);  
  ellipse(x-27, y+25, 20, 20);  
  fill(0, 0, 255);  
  rect(x+30, y-15, 15, 20);  
  fill(127, 255, 212);  
  rect(x+20, y-15, 8, 15);  
  rect(x+10, y-15, 8, 15);  
  ellipse(x, y, 10, 10);  
}
```



x座標、y座標を指示する引数には、必ずx と y を使う

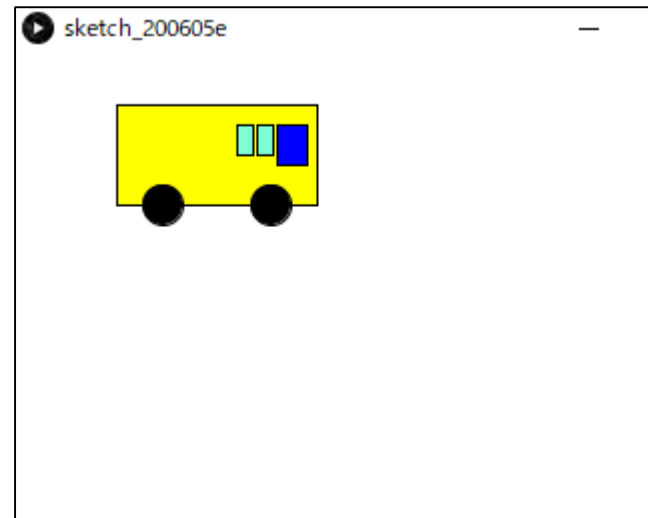
中心点は最後に描く、
中心点の座標は(x, y)のまま変更しない

必ず、中心点が中心になるように配置していくこと！！

中心点をコメントアウトして完成

```
void bus(float x, float y) {  
  fill(255, 255, 0);  
  rect(x-50, y-25, 100, 50);  
  fill(0, 0, 0);  
  ellipse(x+27, y+25, 20, 20);  
  ellipse(x-27, y+25, 20, 20);  
  fill(0, 0, 255);  
  rect(x+30, y-15, 15, 20);  
  fill(127, 255, 212);  
  rect(x+20, y-15, 8, 15);  
  rect(x+10, y-15, 8, 15);  
  //ellipse(x, y, 10, 10);  
}
```

後で確認に使えるので、コメントで残しておく



// の後に書かれた文字は、実行されない
実行しない命令を一時的に消したり、メモを書くときに使う

作ったメソッドで描いた図形を動かす

- 横に動かしたい場合（前回と同様の手順）

```
void setup() {  
  size(400, 400);  
}  
void draw() {  
  background(255);  
  bus(100, 50);  
}
```

ここを変えると左右に動く

```
void bus(float x, float y) {  
  //バスを描く処理  
}
```

ここは各自で作ったメソッド

作ったメソッドで描いた図形を動かす

- 横に動かしたい場合（前回と同様の手順）

```
float x = 100;  
void setup() {  
  size(400, 400);  
}
```

```
void draw() {  
  background(255);  
  bus(100, 50);  
}
```

変数を宣言、初期値を設定（同じにする）

```
void bus(float x, float y) {  
  //バスを描く処理  
}
```

ここは各自で作ったメソッド

作ったメソッドで描いた図形を動かす

- 横に動かしたい場合（前回と同様の手順）

```
float x1 = 100;  
void setup() {  
  size(400, 400);  
}
```

```
void draw() {  
  background(255);  
  bus(x1, 50);  
}
```

変えたい数値の場所に変数を入れる

```
void bus(float x, float y) {  
  //バスを描く処理  
}
```

ここは各自で作ったメソッド

作ったメソッドで描いた図形を動かす

- 横に動かしたい場合（前回と同様の手順）

```
float x1 = 100;  
void setup() {  
  size(400, 400);  
}
```

```
void draw() {  
  background(255);  
  bus(x1, 50);
```

```
  x1 = x1 + 1;
```

変数に数値を足して増やす(x座標が大きくなって右に動く)

```
}  
void bus(float x, float y) {  
  //バスを描く処理  
}
```

ここは各自で作ったメソッド

または、マウス座標と連動

- こちらにする場合は、別に図形を1つ以上動かすこと

```
void setup() {  
  size(400, 400);  
}  
void draw() {  
  background(255);  
  bus(mouseX, mouseY);  
}  
void bus(float x, float y) {  
  //バスを描く処理  
}
```

マウスで動かす図形も含めて、3つ以上のオリジナル図形を動かすこと。

マウスで動かす図形は1つのみ。1つ以上はカウントしない。
(つまり、最低2つの動く図形をプログラミングする)

今回のレポート

1. オリジナルメソッドを作成

- ゲームのキャラクターを意識すると良いかも
- 中心点を中心にして設計すること
 - 中心点を中心にした当たり判定を行うため

2. 作ったオリジナルメソッドを動かす

3. さらに、オリジナルメソッドを動かす

- 作ったものがバスならば、**最低3つ**のバスを動かす
- 前回（変数）の復習

4. 他に動かしたいものがあれば追加（加点項目）

レポートチェックリスト（第6回）

- ミニテストを受験した（1回目）
- オリジナルメソッドを作成した
- オリジナルメソッドを動かした（1回目）
- オリジナルメソッドを動かした（2回目）
- オリジナルメソッドを動かした（3回目）
 - それぞれ別の動き
- 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
- ミニテストを受験した（2回目）

ありそうな質問

- Q:左右反転できないの？
 - 右用と左用を作るのは面倒
- A:ちょっと面倒だけでできます

```
translate(x2,y2); //書きたい座標にする  
scale(-1,1); //左右反転させる  
bus(0,0); //座標は(0,0)にする  
resetMatrix(); //動かしたり反転した座標系を元に戻す
```

- 詳しい説明は割愛。気になるなら聞いてください。
 - 座標系に関しては第8回か9回の内容