たのしいコンピュータプログラミング

君にもできる
シューティングゲーム

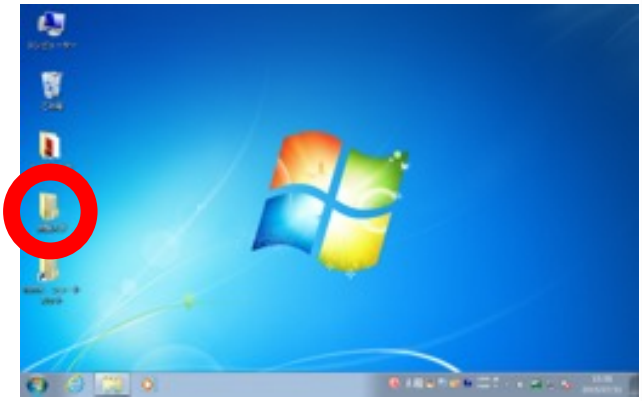
木更津高専 情報工学科
1日体験入学

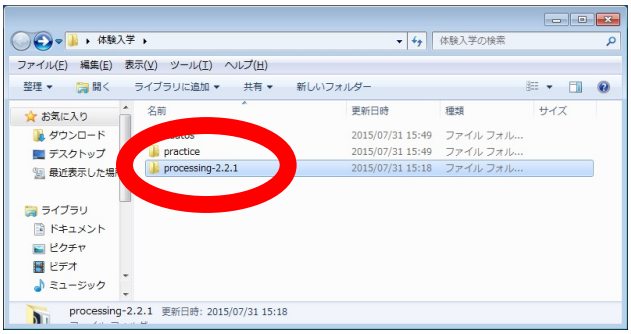
Ver. 2019年

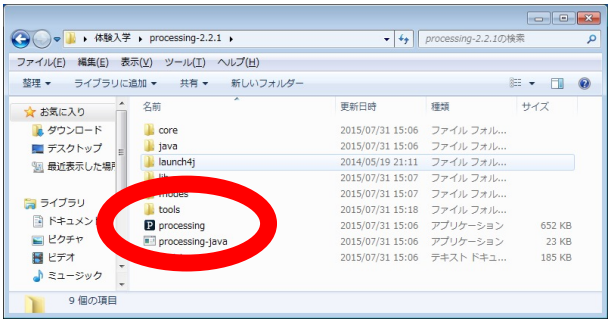
目次

1. ゲームで遊んでみよう！

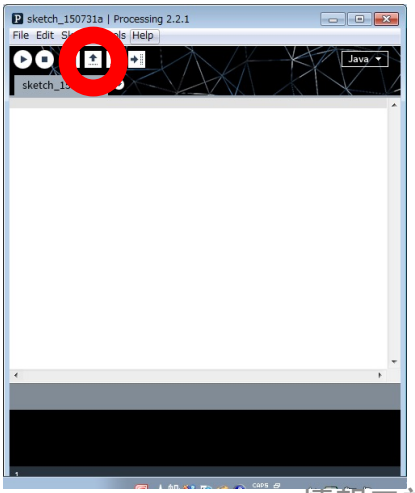
まず，Processingを起動する.

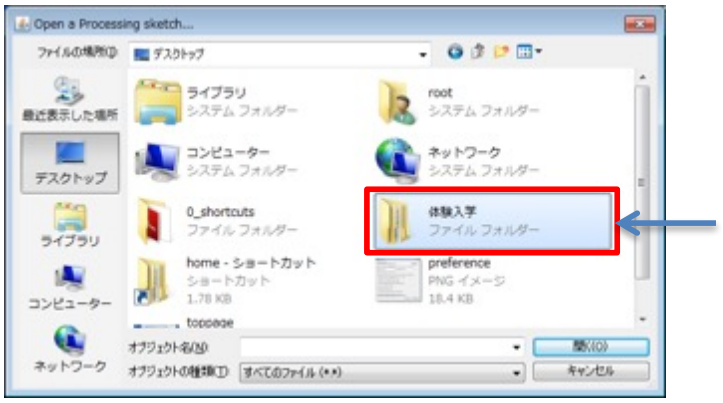


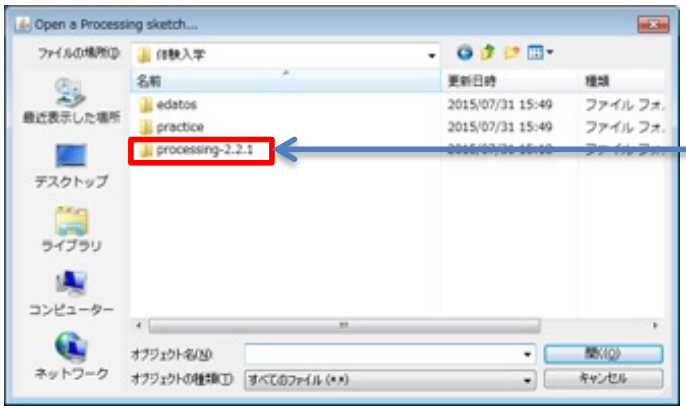


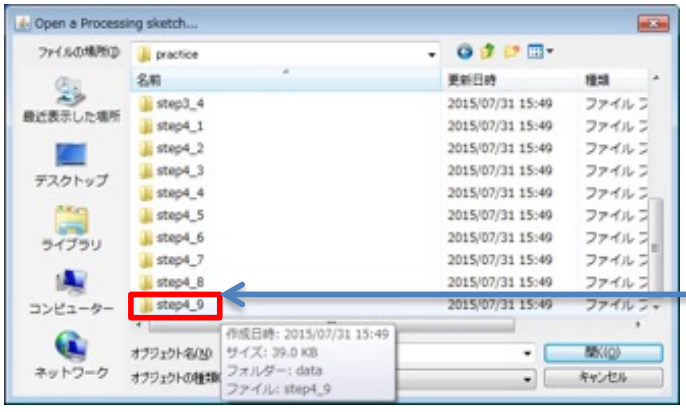


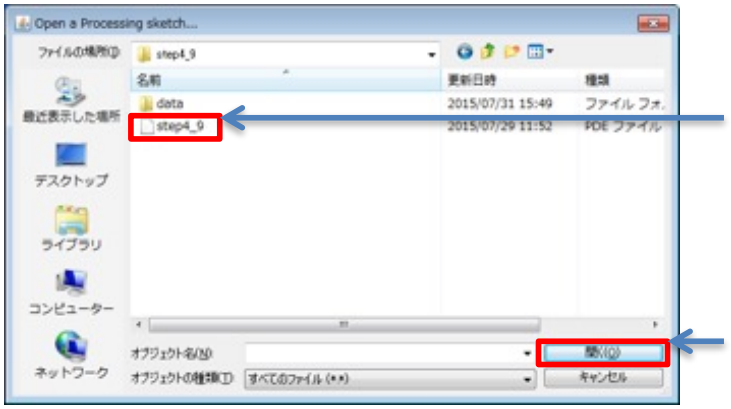
Processingが起動！次にゲームのファイルを開こう.







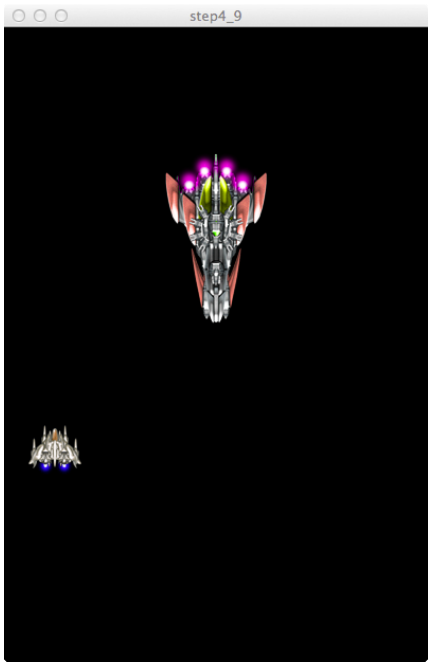




プログラムが開いたら，実行して遊んでみよう．

ここを
クリック

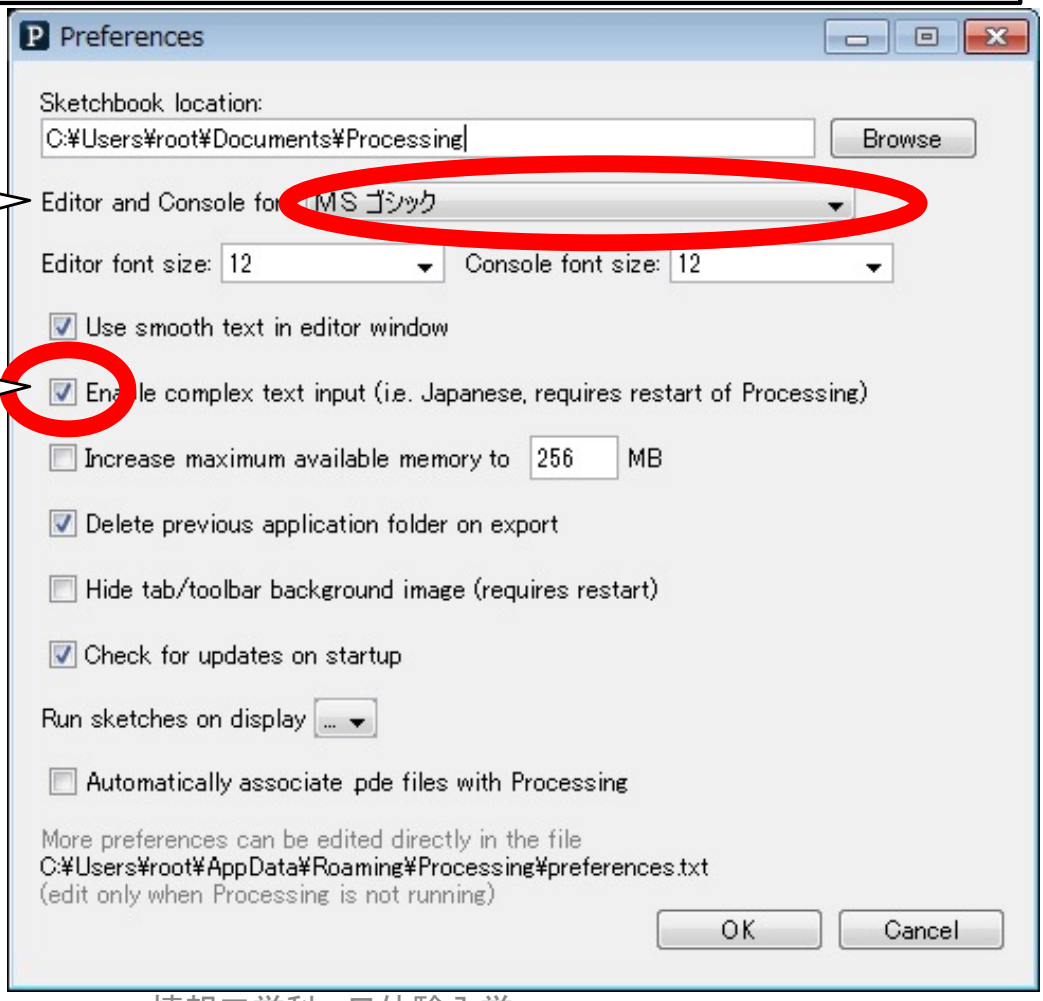
するとゲームが起動！



日本語の設定を行う

MS ゴシック
を選択

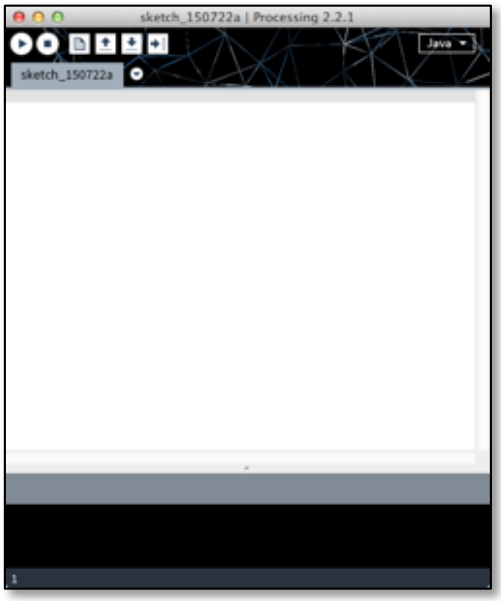
ここをチェック



2. プログラミングを体験してみよう

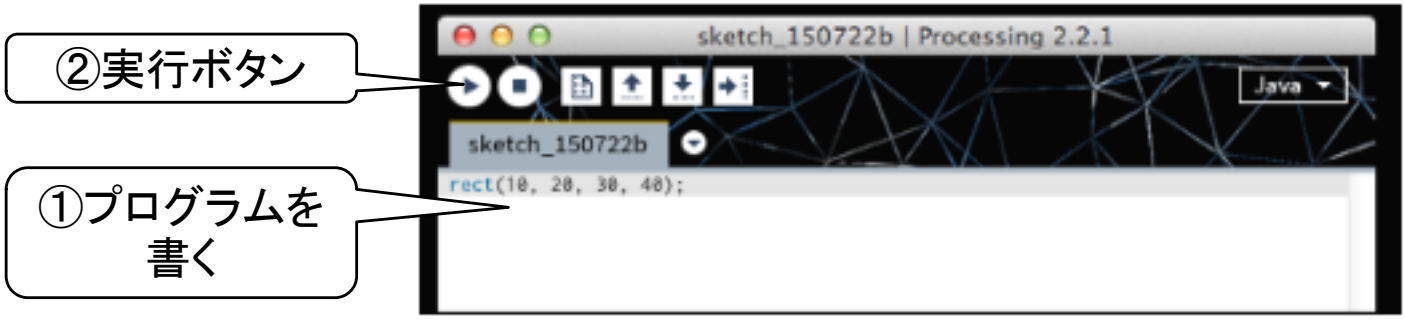
Step2-1. Processingの起動

とにかく，まず動かしてみよう！
先ほどと同様にして，
Processingを立ち上げてみよう.



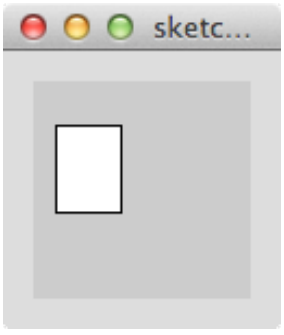
Step2-2. 四角形を表示

先ほど起動したウィンドウの入力部分①に次のプログラムを入力して，
実行ボタン②を押してみよう.




```
rect(10, 20, 30, 40);
```

すると，次のようなウィンドウが表示されます.
これで「初めてのプログラム」が完成しました.

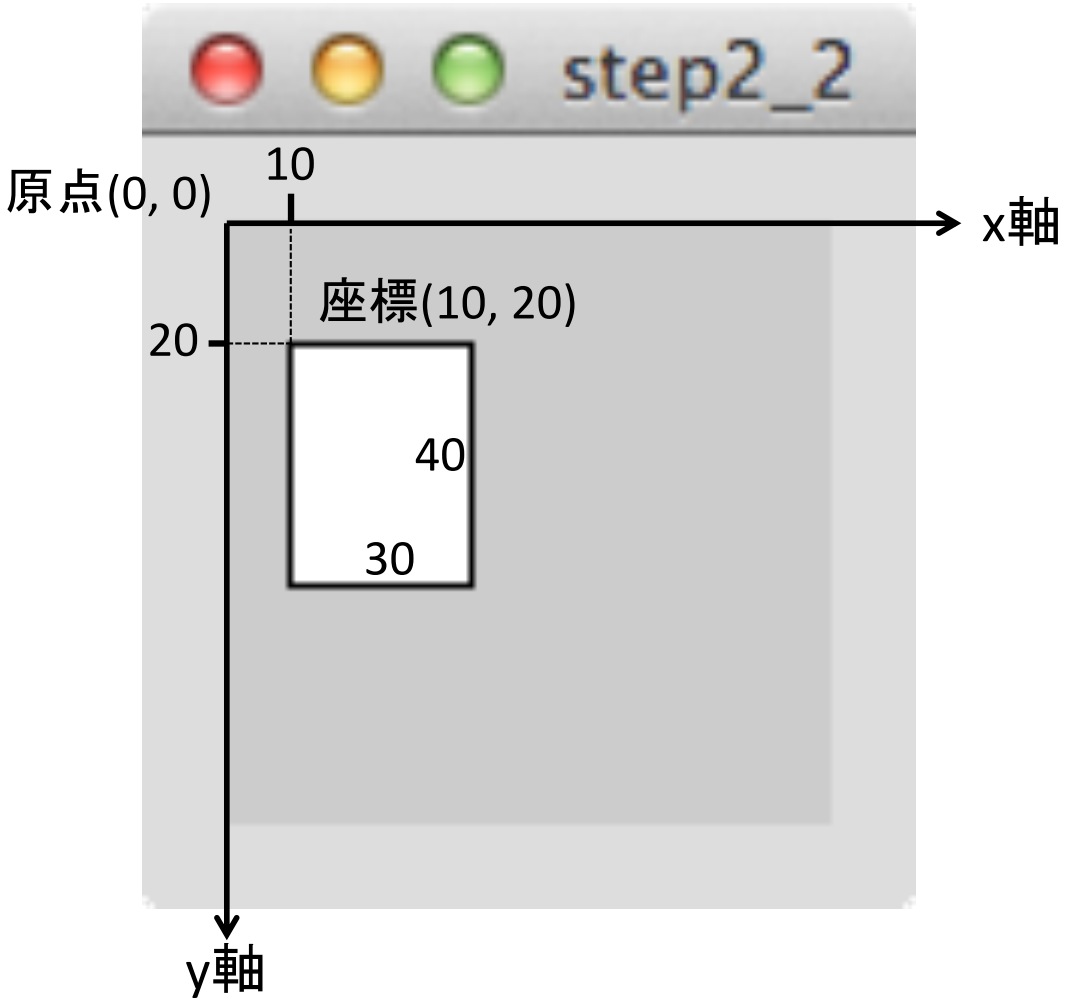


Step2-3. プログラムの意味を理解しよう

このプログラムの意味は次のようになります.

```
rect(10, 20, 30, 40);
```

rect(四角形の左上頂点のx座標,
　　四角形の左上頂点のy座標,
　　四角形の横（x方向）の長さ,
　　四角形の縦（y方向）の長さ);



y軸は正負の向きが違うので注意.
下方向がプラスになります

問題2-1　座標(25, 50)に，横が40，縦が5の長方形を描いてみよう

解答はanswer2_1

Step2-4. 変数を使えるようになろう

好きな名前の変数を作って，数値を格納することができる．

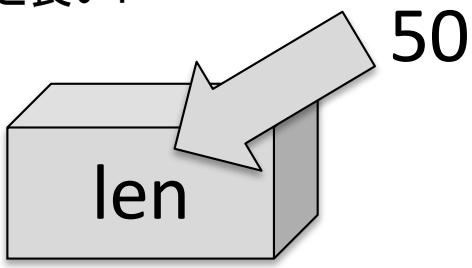
```
int len;
len = 50;
rect(10, 20, len, len);
```

int len;

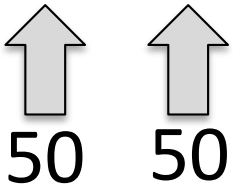
この1行で，プログラム内部に数値を格納できる箱ができる．変数の名前は自分で決めることができる．

len = 50;

この1行で，変数に数値を格納する．“=”の記号は数学の等号ではなく，
値を代入する意味である．“←”の矢印だと思うと良い．



rect(10, 20, len, len);



変数lenには50が入っていて，呼び出して何度でも使うことができる．

問題2-2　座標(50, 10)に，1辺が100の正方形を描いてみよう

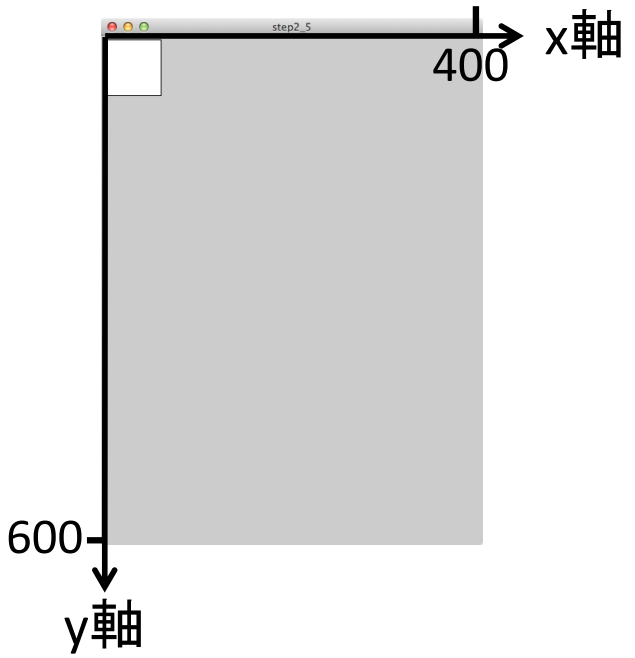
解答はanswer2_2

Step2-5. 画面のサイズを変えよう

```
size(400, 600);
int len;
len = 100;
rect(50, 10, len, len);
```

size(400, 600);

画面の横（ｘ方向）が400,
縦（y方向）が600になる.

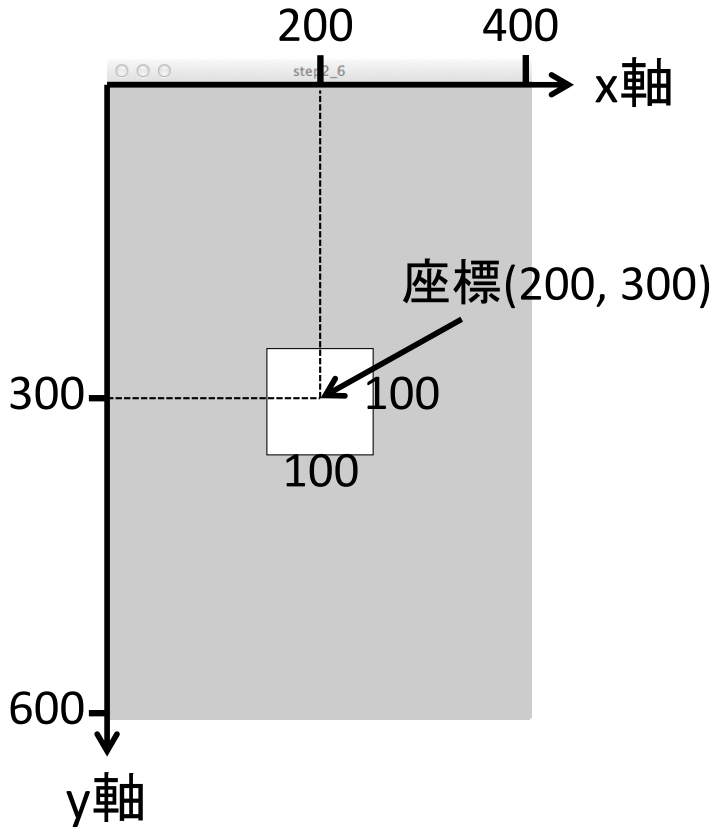


Step2-6. 表示モードをセンターにしよう

```
rectMode(CENTER);
size(400, 600);
int len;
len = 100;
rect(200, 300, len, len);
```

rectMode(CENTER);

四角形の中心が，指定した座標になる.



Step2-7. 画像を表示しよう

```
PImage shipImg;
size(400, 600);
imageMode(CENTER);
shipImg = loadImage("ship.png");
image(shipImg, 80, 300);
```

PImage shipImg;

Pimage型の変数を宣言し，その変数名をshipImgとした.

imageMode(CENTER);

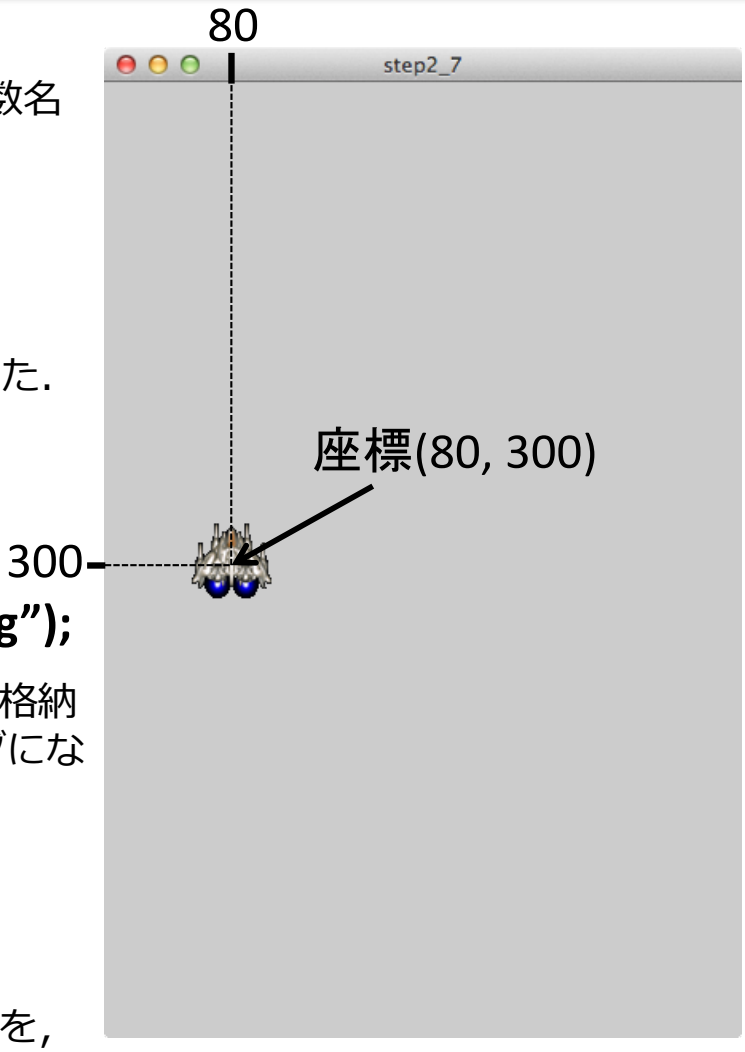
画像の座標指定を，画像の中心にした.

shipImg = loadImage(“ship.png”);

画像を読み込んで，変数shipImgに格納した．読み込む画像はdataフォルダになければならない.

image(shipImg, 80, 300);

変数shipImgに格納されている画像を，x座標80, y座標300を中心として画像を表示する.



問題2-3　座標(200, 450)に，自機を表示してみよう．このとき自機のx座標をshipX，y座標をshipYとして変数を用意し，これらの変数を使って表示してみよう.

解答はanswer2_3

3. プログラミングに慣れよう

Step3-1. 画像を動かしてみよう

ゲームはパラパラ漫画です．実は，1秒間に60枚の画像をパラパラめくりながら動いているように見せています．

```
PImage shipImg;
int shipX;
int shipY;
```

→ 全体で使う変数を宣言

```
void setup() {
  size(400, 600);
  imageMode(CENTER);
  shipImg = loadImage("ship.png");

  shipX = 200;
  shipY = 450;
}
```

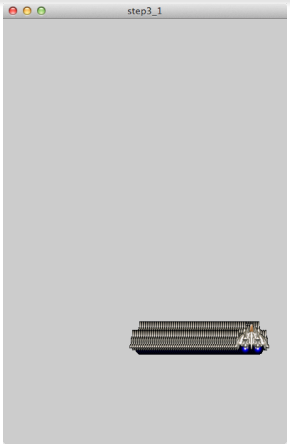
→ 実行した時に，最初に一度だけ行う．ここで，ウィンドウサイズや画像の読み込みを済ませておく．

```
void draw() {
  shipX = shipX + 3; // 右に移動
  image(shipImg, shipX, shipY);
}
```

→ 何度も繰り返す部分．ここにゲームの中身を書いていく．

shipX = shipX + 3;

現在の自機のx座標に3を足す．そして，足した値をshipXに代入する．

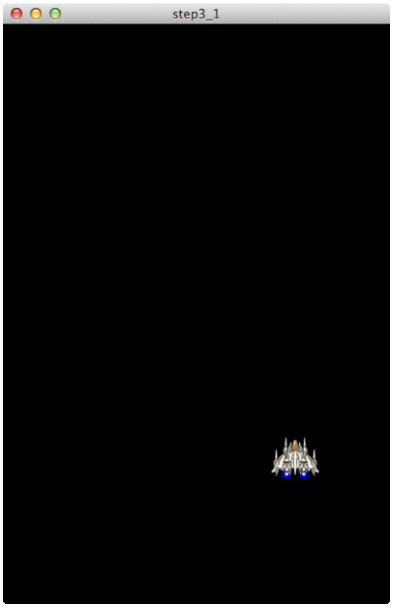


これで自機が動く．
しかし，ちょっと表示がおかしい．．．

Step3-2. 背景を書こう

画面を表示するときに，毎回背景を書いて，画面を黒く塗りつぶすと自機が動いているように見える．

```
PImage shipImg;
int shipX;
int shipY;

void setup() {
  size(400, 600);
  imageMode(CENTER);
  shipImg = loadImage("ship.png");

  shipX = 200;
  shipY = 450;
}

void draw() {
  background(0);
  shipX = shipX + 3; // 右に移動
  image(shipImg, shipX, shipY);
}
```



何度も繰り返す部分．

繰り返すときに，背景を黒く塗りつぶす．これで，前回書いた自機を塗りつぶすことになるので，正しく動いているように見える．0が黒，255が白になる．

しかし，自機が画面の外に行ってしまう．．．

Step3-3. if文を使えるようになろう

条件に応じて動作が変化するようにしてみよう.

```
void draw() {
  background(0);
  if (shipX<400) {
    shipX = shipX + 3;
  }
  image(shipImg, shipX, shipY);
}
```

もし，shipXの値が400未満なら，{ から } までの中身を実行する.

しかし，右端で止まってしまう. . .

Step3-4. 自機をマウスで動かしてみよう

マウス入力を受け取ってみよう.

```
void draw() {
  background(0);
  shipX = mouseX;
  shipY = mouseY;
  image(shipImg, shipX, shipY);
}
```

mouseX, mouseYでマウスが画面内にあるときの座標を読み取ることができる.

4. 簡単シューティングゲームを作ってみよう

Step4-1. 弾を打ってみよう

```
PImage shipImg;
int shipX;
int shipY;

PImage laserImg;
int laserX;
int laserY;

void setup() {
  size(400, 600);
  imageMode(CENTER);
  shipImg = loadImage("ship.png");
  laserImg = loadImage("laser.png");
}

void draw() {
  background(0);
  shipX = mouseX;
  shipY = mouseY;
  image(shipImg, shipX, shipY);

  if (mousePressed == true) {
    laserX = shipX;
    laserY = shipY;
  }
  image(laserImg, laserX, laserY);
  laserY = laserY - 5;
}
```

弾の画像，x座標，y座標を格納する変数を宣言する.

弾の画像を読み込む.

もし，マウスが押されたら

弾の座標を，自機の座標からコピーする.

表示した後，弾のy座標を-5する.

しかし，弾が画面にあるときに，発車すると途中で消えてしまう.

Step4-2. 弾が途中で消えないようにする

```
PImage shipImg;
int shipX;
int shipY;

PImage laserImg;
int laserX;
int laserY;
int laserAlive; // 弾が画面にあるとき1，ないとき0にする

void setup() {
  size(400, 600);
  imageMode(CENTER);
  shipImg = loadImage("ship.png");
  laserImg = loadImage("laser.png");

  // laserAliveが1のとき存在する．0のとき存在しない．
  laserAlive = 0;
}

void draw() {
  background(0);
  shipX = mouseX;
  shipY = mouseY;
  image(shipImg, shipX, shipY);

  if (mousePressed == true) {
    // 画面内に弾が存在しないなら発射
    if (laserAlive == 0) {
      laserX = shipX;
      laserY = shipY;
      laserAlive = 1; // 発射したので，1にする
    }
  }
  image(laserImg, laserX, laserY);
  laserY = laserY - 5;
  // 弾が画面の外に出たら，レーザーを消す
  if (laserY<0) {
    laserAlive = 0;
  }
}
```

弾が画面にあるのに，再び弾を発射できることが原因である．

そこで，画面に弾があるときには，弾を発射できないようにすれば良い．

そのために，laserAliveという変数を用意する．これは，フラグと呼ばれるものである．

フラグが立っているかどうかで処理を変更する．

laserAliveが1のとき，画面に弾が存在し，0のとき存在しないことを示す．

弾が存在しない時だけ，弾の座標を自機の座標をコピーできるようにすればよい．

そして，弾が発射されたことにするためにlaserAliveを1にする．

さらに，画面の外に弾が出たら，再び弾を発射できるようにしなければならない．

そこで，弾のy座標が0未満になったら，laserAliveを0にして，弾が画面内に存在しいことにする．

Step4-3. 敵を出現させてみよう

```
PImage shipImg;
int shipX;
int shipY;

PImage laserImg;
int laserX;
int laserY;
int laserAlive;

PImage enemyImg; // 敵の画像
int enemyX; // 敵のx座標
int enemyY; // 敵のy座標

void setup() {
  size(400, 600);
  imageMode(CENTER);
  shipImg = loadImage("ship.png");
  laserImg = loadImage("laser.png");
  enemyImg = loadImage("enemy.png"); // 敵の画像を読み込む

  laserAlive = 0;
}

void draw() {
  background(0);

  // 敵を表示
  enemyX = 200;
  enemyY = 80;
  image(enemyImg, enemyX, enemyY);

  shipX = mouseX;
  shipY = mouseY;
  image(shipImg, shipX, shipY);

  if (mousePressed == true) {
    if (laserAlive == 0) {
      laserX = shipX;
      laserY = shipY;
      laserAlive = 1;
    }
  }
  image(laserImg, laserX, laserY);
  laserY = laserY - 5;
  if (laserY<0) {
    laserAlive = 0;
  }
}
```

これまでと同様に，敵の画像と座標を用意する．

敵の画像を読み込む．

敵の画像を表示する．

しかし，動かない．．．

Step4-4. 敵を動かしてみよう

ここからは，プログラムは必要な部分だけ抜き出して説明します．

```
PImage enemyImg;
int enemyX;
int enemyY;
int enemyAlive; // 敵が画面にあるとき1，ないとき0にする

void setup() {
  size(400, 600);
  imageMode(CENTER);
  shipImg = loadImage("ship.png");
  laserImg = loadImage("laser.png");
  enemyImg = loadImage("enemy.png");

  laserAlive = 0;
  enemyAlive = 0; // 1のとき敵が画面内に存在する．0のとき存在しない
}

void draw() {
  background(0);

  // 敵を表示
  // 敵がいないなら
  if (enemyAlive == 0) {
    enemyX = 200;
    enemyY = -100;
    enemyAlive = 1;
  }
  // 敵がいるなら
  if (enemyAlive == 1) {
    image(enemyImg, enemyX, enemyY);
    enemyY = enemyY + 1;
  }
  // 敵が画面の外に出たら
  if (enemyY > 800) {
    enemyAlive = 0; // 敵を消す
  }

  shipX = mouseX;
  shipY = mouseY;
  image(shipImg, shipX, shipY);

以下，省略.
```

弾のときと同様に，画面にあるかどうかのフラグを用意する．

最初は敵が存在しないので，enemyAliveを0にしておく．

もし，敵がいないなら，敵を出現させる．
敵のy座標を-100にしているのは，画面外の上部から現れるようにするためである．
敵の存在フラグを1に変える．

もし，敵がいるなら，移動させる．

もし，敵が画面下の外側に出たら，敵を消す．

Step4-5. 敵と弾の当たり判定を行ってみよう

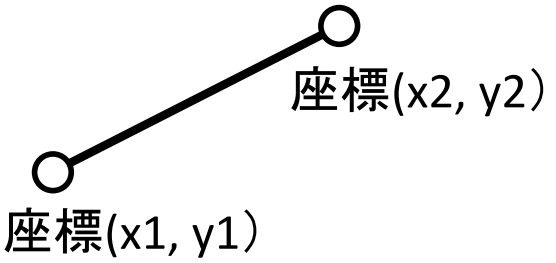
```
void draw() {
 background(0);

 if (enemyAlive == 0) {
  enemyX = 200;
  enemyY = -100;
  enemyAlive = 1;
 }

 if (enemyAlive == 1) {
  image(enemyImg, enemyX, enemyY);
  enemyY = enemyY + 1;
 }

 if (enemyY > 800) {
  enemyAlive = 0;
 }

 // 敵と弾の中心座標間の距離が30未満なら
 if (dist(enemyX, enemyY, laserX, laserY) < 30) {
  enemyAlive = 0;  // 敵を消す
 }

以下，省略.
```

敵と弾の距離をdist()を使って測る．もし，その距離が30未満なら，敵の存在フラグを0にして敵を消す．
30が小さければ当たり判定が厳しくなり，大きければ甘くなる．

ただし，敵を倒した後の弾が消えない．．．

dist(x1, y1, x2, y2);

座標(x1, y1）と座標(x2, y2)の距離を測る



Step4-6. 敵を倒した後の弾も同時に消してみよう

```
void draw() {

 // 弾が存在するなら
 if (laserAlive == 1) {
  // 敵と弾の中心座標間の距離が30未満なら
  if (dist(enemyX, enemyY, laserX, laserY) < 30) {
    enemyAlive = 0; // 敵を消す
    laserAlive = 0; // 弾を消す
  }
 }
```

laserAliveを0にすれば表示を消すことができる．ただし，laserXとlaserYの値は，敵を倒した場所に残る．

そこで，さらにif文「弾が存在するなら」を追加する．

これにより，見えない弾と敵との距離が30未満でも，弾が存在しない状態なら敵を消さない，ということになる．

しかし，真ん中からしか敵が出現しない．．．

Step4-7. ランダムに敵が出現するようにしてみよう

```
void draw() {
 background(0);

 if (enemyAlive == 0) {
  enemyX = (int)random(400);
  enemyY = -100;
  enemyAlive = 1;
 }
```

(int)random(400)で，0から399までの整数値がランダムに生成される．

Step4-8. 敵の移動速度をランダムにしてみよう

```
int moveY; // 敵のy軸の移動方向

void draw() {
 println(enemyX, enemyY);
 if (enemyAlive == 0) {
  moveY = (int)random(10) + 1;
 }
 if (enemyAlive == 1) {
  enemyY = enemyY + moveY;
}
```

(int)random(10)で0から9までの乱数．それに1を足すので，1から10の乱数になる．

もし，1を足さず，0から9の乱数だと，たまたまmoveYが0になってとき，敵が移動しなくなってしまう．

コンソール（エディタの下部）に値を表示する．

moveYに応じて，移動させる．

Step4-9. 完成した簡単シューティングゲームを確認しよう

```
// 自機
PImage shipImg; // 自機の画像を格納する変数を宣言
int shipX;     // 自機のx座標の変数を宣言
int shipY;     // 自機のy座標の変数を宣言

// 弾
PImage laserImg;
int laserX;
int laserY;
int laserAlive; // 弾が画面内に存在するかどうかのフラグ

// 敵
PImage enemyImg;
int enemyX;
int enemyY;
int enemyAlive;
int moveY;     // 敵のy軸の移動方向


//--------------------------------
// ゲーム起動時に1度だけ実行する
//--------------------------------
void setup() {
 size(400, 600); // 画面サイズを400 x 600の長方形にする
 imageMode(CENTER); // 画像座標の指定方法を中心に設定

 // 画像を読み込む
 shipImg = loadImage("ship.png");
 laserImg = loadImage("laser.png");
 enemyImg = loadImage("enemy.png");

 // 存在する場合1，存在しない場合0にする
 laserAlive = 0;
 enemyAlive = 0;
}

//-----------------------------------------
// ゲーム実行時に1秒間に60回画面が更新される
//-----------------------------------------
void draw() {
 background(0); // 画面クリアー（画面を黒を塗りつぶす）

 println(enemyX, enemyY); // 敵の座標をコンソールに出力

 //---------
 // 敵の処理
 //---------
 // 敵が画面内にいないなら，出現させる.
 if (enemyAlive == 0) {
  enemyX = (int)random(400);
  enemyY = -100; // 画面の外から現れるようにするため
 enemyAlive = 1;
  moveY = (int)random(10) + 1; // 1から10の乱数を生成
 }
 // 敵が画面内にいるなら，表示させる
 if (enemyAlive == 1) {
  image(enemyImg, enemyX, enemyY);
  enemyY = enemyY + moveY; // moveYに応じて移動させる
 }

 // 敵が画面下の外側に出たら敵の存在フラグを0にして消す
 if (enemyY > 800) {
  enemyAlive = 0;
 }

 // 弾が存在するなら，
 if (laserAlive == 1) {
  // 敵と弾の中心座標間の距離が30未満なら
  if (dist(enemyX, enemyY, laserX, laserY) < 30) {
   enemyAlive = 0; // 敵を消す
   laserAlive = 0; // 弾を消す
  }
 }

 //-----------
 // 自機の処理
 //-----------
 shipX = mouseX;
 shipY = mouseY;
 image(shipImg, shipX, shipY);

 //---------
 // 弾の処理
 //---------
 if (mousePressed == true) { // マウスが押されたら，
  if (laserAlive == 0) { // 画面内に弾が存在しないなら弾発射
   laserX = shipX; // 弾の座標は自機の座標からコピーする.
   laserY = shipY;
   laserAlive = 1; // 今，弾を打ったので，フラグを立てる.
  }
 }

 // 弾が画面内にいるなら，表示させる
 if (laserAlive == 1) {
  image(laserImg, laserX, laserY);
  laserY = laserY - 5; // 上に移動
 }

 // 弾が画面上の外側に出たら弾の存在フラグを0にして消す
 if (laserY<0) {
  laserAlive = 0;
 }
}
```

5. 本格シューティングゲームを改造してみよう

この調子でシューティングゲームを作り込んでいけば，本格シューティングゲームが出来上がります．しかし，完全なシューティングゲームを作成するには時間がない！

そこで，改造を行ってみましょう．
プログラミング学習として，ソースコードを読んで理解し，改造するというやり方は非常に良い方法です．チャレンジしてみよう！

１．自機がy軸方向も動けるようにしてみよう

２．レーザーの速度を変えてみよう

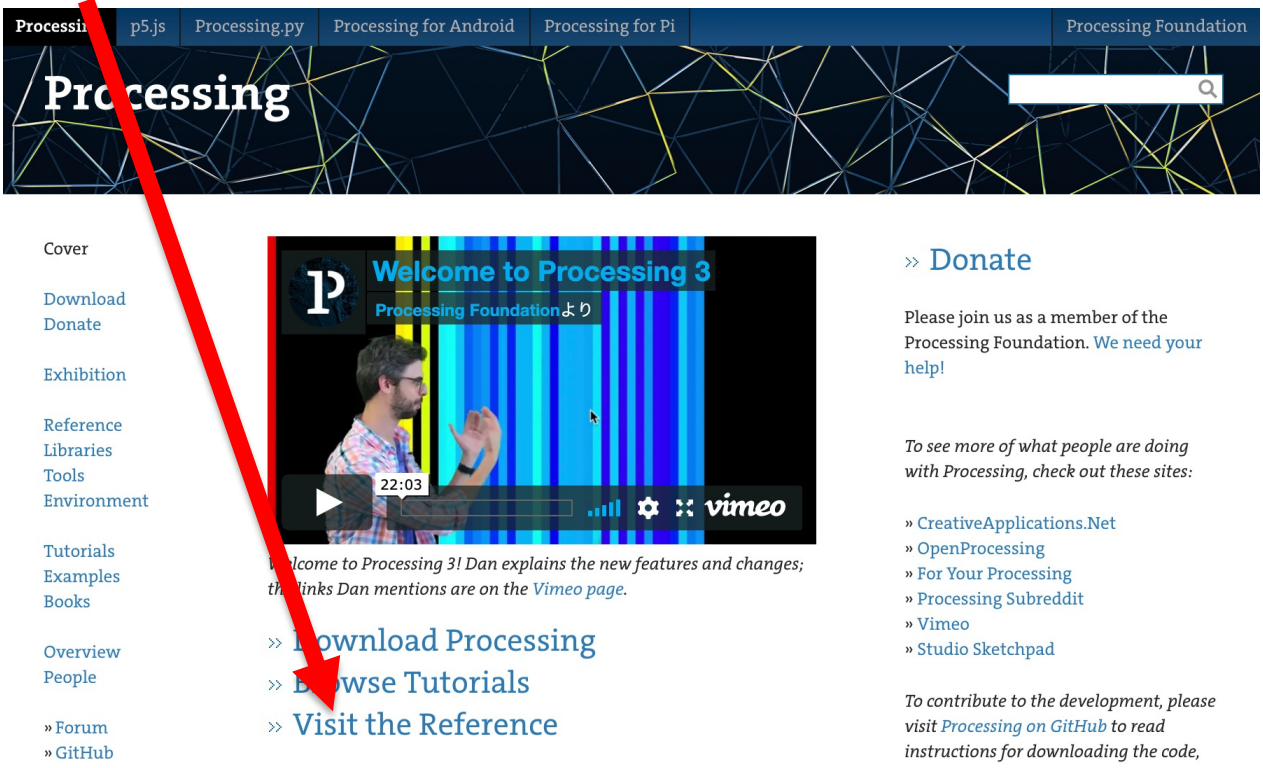
３．敵を倒した時の得点を変えてみよう

４．弾の数を増やしてみよう

５．敵の数を増やしてみよう

・Processingのインストール

1. https://processing.org/ にアクセスして，Download Processing をクリック.



2. Windows (64bit) をクリックして，zipファイルをダウンロード．解凍して実行する．（パソコンのOSに応じて選択する）

・サンプルプログラムの紹介

1. 体験入学で使った教材
https://tinyurl.com/kisarazu-taiken

もし，繋がらなかったら，下のURL
https://www.dropbox.com/sh/dh1owq8xfeqco5o/AAB-J04asDSlfTZyx6PknrqJa?dl=0
（途中にある「…DSlfT…」と「a?dl=0」の「l」は小文字のLです）

2. 情報工学科のホームページ
http://www.kisarazu.ac.jp/gakka/information/
（木更津高専のホームページ http://www.kisarazu.ac.jp/から，学科・専攻科紹介を辿って行けます）

3. 体験入学で作成したシューティングゲームの実況プログラミング
http://youtu.be/W8FQhGHLlC4

・参考文献

1. “Processing アニメーションプログラミング入門”，田中孝太郎，技術評論社，2011.
2. Processingのサイト，https://processing.org
3. フリー素材画像
（現在サイトがなくなっている．2015年時点に下記サイトのフリー素材を使用した．）
http://game.yu-nagi.com/index.htm
http://homepage2.nifty.com/hamcorossam/