プログラミング教育

九州産業大学 情報科学部 情報科学科 教授
米元 聡

■利用するサポートページの確認

- http://www.is.kyusan-u.ac.jp/~yonemoto/
  - ✓受講者用PCの関連リンク集からたどる
  - ✓学内限定（一部公開）

Google Chromeを開き 「**米元研究室**」 と検索

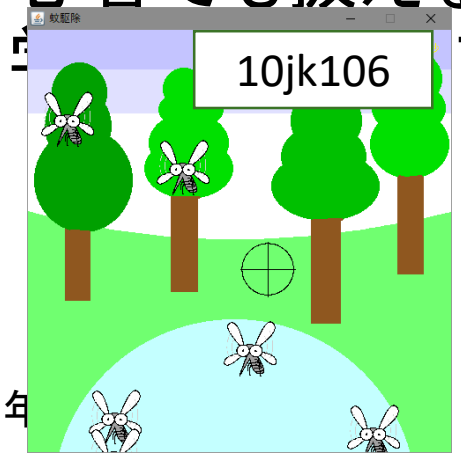
- 演習時間を節約するために命令文をコピー＆ペーストするために利用
- この資料（スライド）以外の補足事項を公開

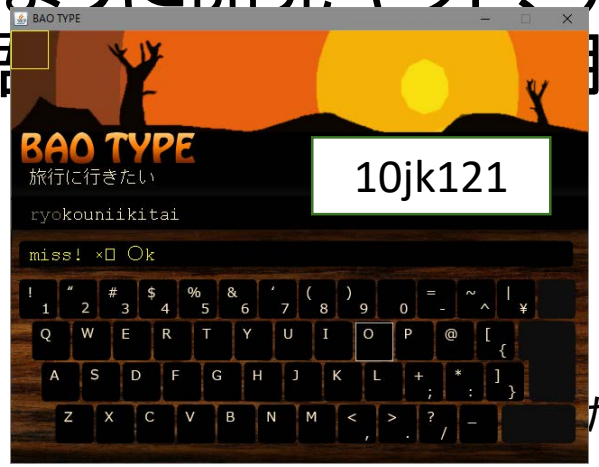
■Javaによる2Dグラフィックスプログラミング

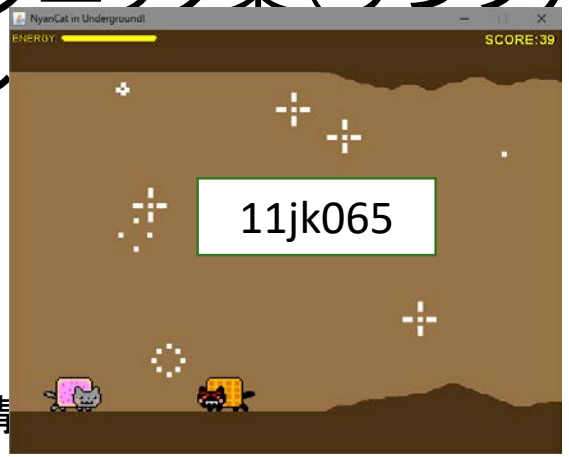
本学の情報科学部の演習（3年前期）・選択科目

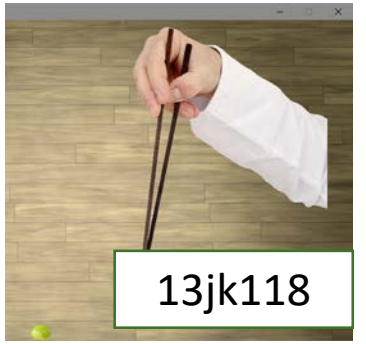
- プログラミングを専門に学ぶ学生向け
  - ✓グラフィックスの理解が目的ではない
  - ✓プログラミングスキル（設計および実装）の向上が目的
  - ※プログラミングの自由課題を扱える科目は他にない
- 最後に発表会（プレゼン）を実施
  - ✓自由課題について口頭発表（2,3分程度）
- Javaはグラフィックスプログラミングには向いていないため
  初心者でも扱えるように開発キット、テクニック集（サンプルコード）を準備
  ※学[illegible]語[illegible]用し[illegible]

自由課題は学生にとって困難な課題に相当！









■学生作品の紹介（Processingの利用）

- ●対象学生：
  - ✓情報科学部１年生 10名程度
    - プログラミング科目履修前の未経験者
    - ※ほぼ未経験者なので高校２・３年生と同じレベル
  - ✓前期・後期の年２回×５年ほど継続実施
- ●取り組み時間：
  - 90分×3コマでメニューに取り組んだ後、
  - 90分×3コマで自由課題完成
  - （90分×2コマでレポート作成）
- ●指導体制：
  - ✓教員１名＋補助のLA（アシスタント学生3,4年）１名
  - ✓基本的に実現したいこと（完成イメージ）は学生決めてもらう。

[デモ]

■Processing入門（アニメーションプログラミング）

メニュー

- **ウインドウ生成**
- **描画命令の利用①②**
- **マウス・キーボードの利用**
- **変数（メモリ）の利用**
- **条件文の利用**
- モード管理（変数＋条件文）

- 例題プログラムの実装・改良
  - ✓国旗の描画 [*複数の図形の配置*]
  - ✓ボールの跳ね返り [*座標と方向の定義・動き*]
  - ✓簡単な物理シミュレーション [*放物運動*]
  - ✓数式（グラフ）の描画 [*浮動小数点の扱い*]

ポイント

- 白紙のプログラムから1行ずつ入力
- 数値（座標）を自由に変える
- 複数の図形を描かせる
- キャンバスに描く順序を意識させる
  先に描いた方が裏側にくる
- 変数（メモリ）
  記憶できる数値、好きに名前をつける
- 条件文
  - ✓ ある条件が成り立つ時のみ命令を実行
  - ✓ 分岐（A成立ならBそうでなければCを実行）は教えない
- 無限ループ（描画の繰り返し）を使って動きを実現

今回は時間がないので1つの図形のみ

■Processing入門1: ウインドウ生成

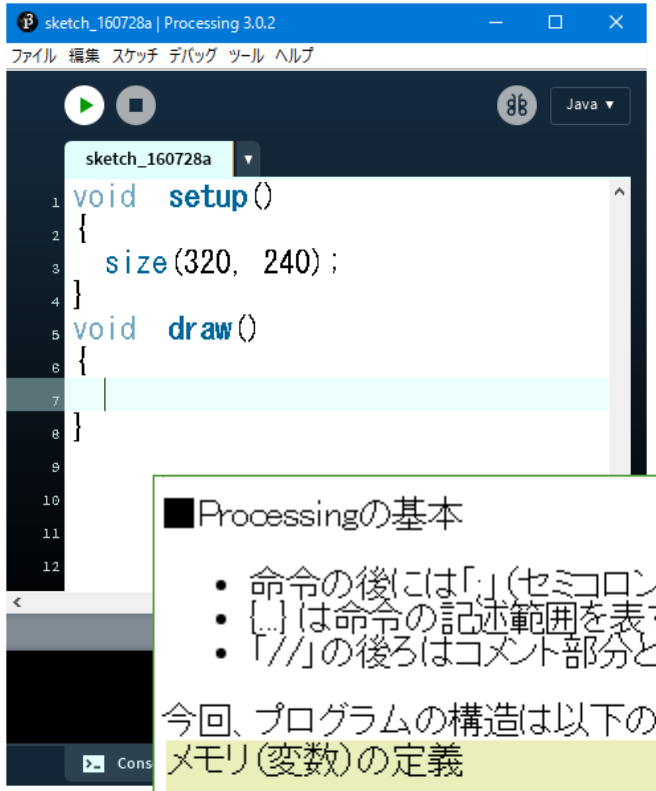
```
void setup()
{
   size(320, 240);
}
void draw()
{

}
```

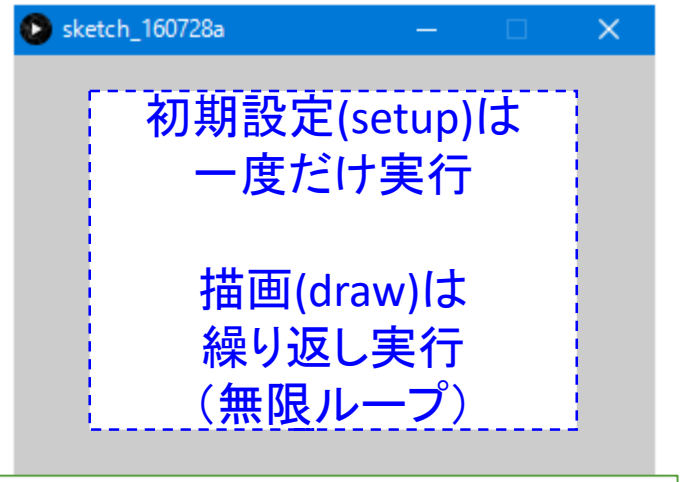
①初期設定

ウインドウサイズ
320 x 240

②描画処理

{と}でプログラムの記述範囲を表す





■Processingの基本

- 命令の後には「;」(セミコロン)をつける。
- {...} は命令の記述範囲を表す。今回利用するのは、setup(){...}とdraw(){...}
- 「//」の後ろはコメント部分となる。

今回、プログラムの構造は以下のように3つの部分から成ると考える。

メモリ(変数)の定義

```
int x, y;        ←(例) 座標(x,y)というメモリを定義
```

```
void setup() {
      :          ←メモリの最初の値などをここに記述(一度だけ実行される)
}
```

```
void draw() {
      :          ←描画命令はここに記述(繰り返し実行される)
}
```

【演習】320x240の大きさのウインドウを生

■Processing入門2: 描画命令の利用①

```
void  setup()
{
   size(320, 240);
}
void  draw()
{
  ellipse(160,200, 40,40);
}
```

fill(255,0,0);
の追加で赤色に

中心(160,200)、大きさ40x40の円を描画

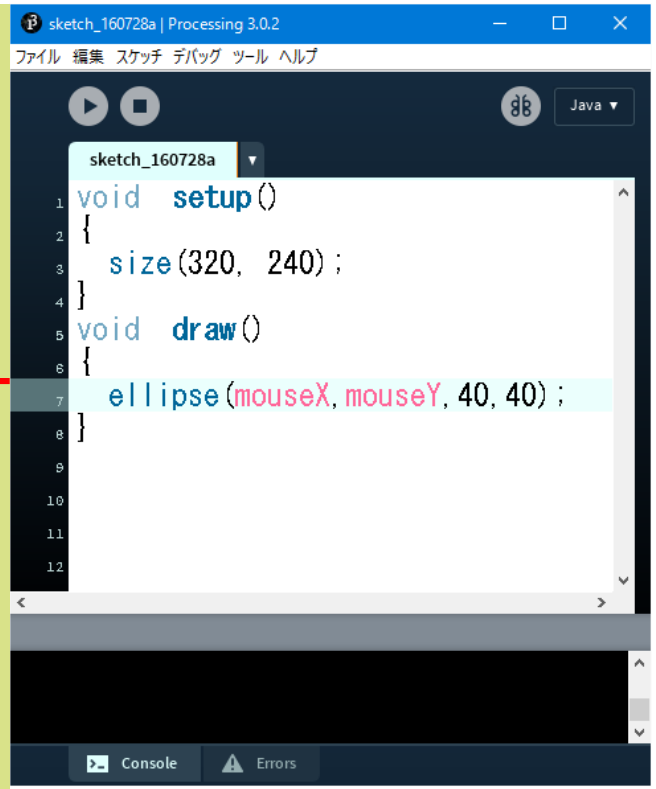
【演習】円(、線、矩形、塗りつぶし円)を描く。

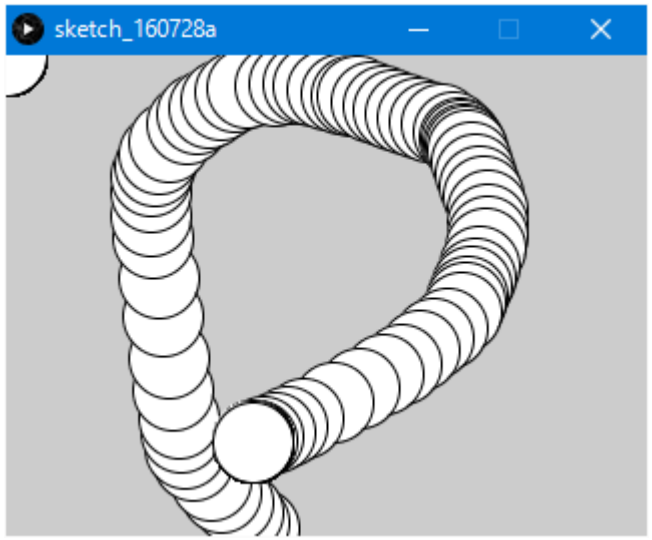
line(160,200, 200,100);　　rect(160,200, 80, 40);
fill(255,0,0);　←塗るときは直前に色を指定((255,0,0)は赤)

■Processing入門3: マウス・キーボードの利用

```
void  setup()
{
   size(320, 240);
}
void  draw()
{


  ellipse(mouseX, mouseY,40,40);
}
```

マウスの座標は
(mouseX, mouseY)





【演習】円をマウスで動かす。

キャンバスをクリアするには単色で塗る（background命令）

■Processing入門4: 変数(メモリ)の利用

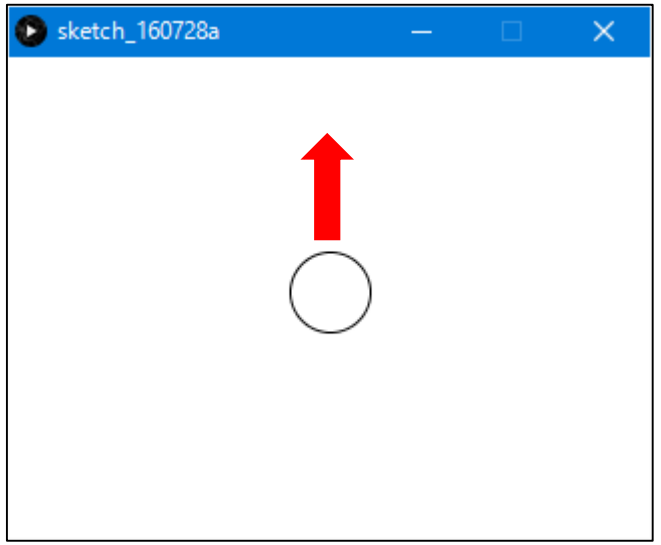
```
int  x, y;
void  setup()
{
   size(320, 240);
   x = 160;  y = 200;
}
void  draw()
{
  background(255,255,255);
  ellipse(x, y, 40,40);
  y = y - 1;
}
```

①座標がメモリ
一番上にかくとよい
int… 整数であることを表す

②初期値はここで

③円がメモリに連動

④メモリの値は変更可能



【演習】円の上方移動(座標の記憶と変更)。

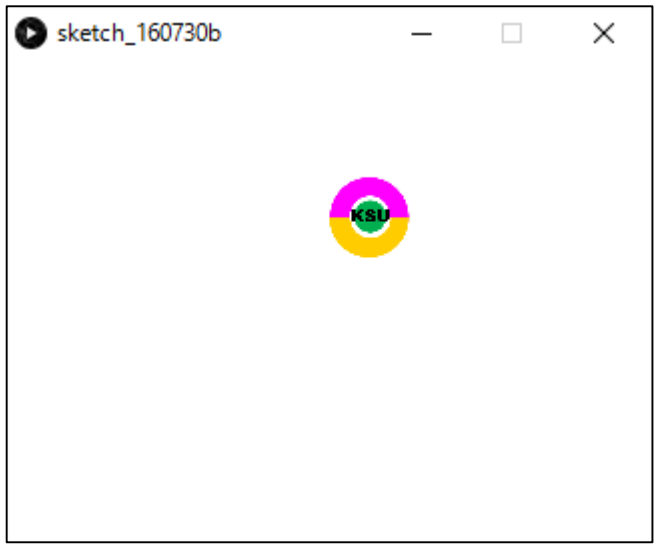
■Processing入門5: 描画命令の利用②

```
PImage  img;   // 画像メモリ名
int  x, y;
void  setup() {
    size(320, 240);
    x = 160;  y = 200;
    img = loadImage("ball.png");
}
void  draw()  {
   background(255,255,255);
   image(img, x, y, 40,40);
   y = y - 1;
}
```

③画像メモリへの読み込み
"http://～/ball.png"
とurlを指定してもよい

④画像の描画（円を差し替え）

画像ファイル（ball.png）



①まずプログラムを保存
（sketch_160824aというフォルダ名になる）
②そのフォルダに画像ファイル(ball.png)を置く
③画像メモリ名 img に画像データを読んでおく
④画像の描画はimage(img, x,y, 40,40);

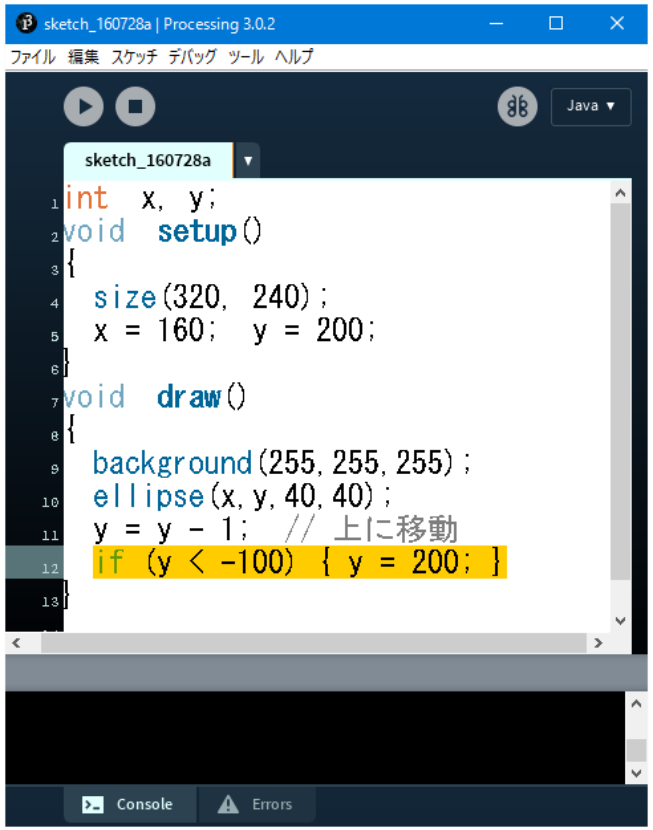
【演習】画像を読み込み円の代わりに描く。

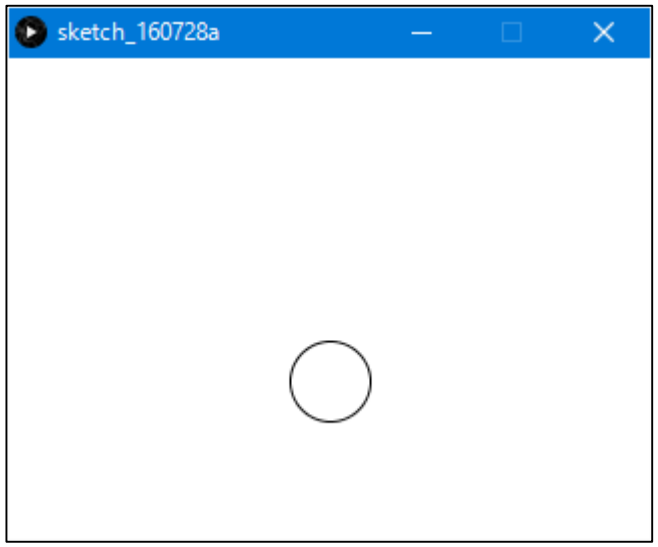
■Processing入門6: 条件文の利用

```
int  x, y;
void  setup() {
    size(320, 240);
    x = 160;  y = 200;
}
```

```
void  draw() {
   background(255,255,255);
   image(img, x, y, 40,40);
   y = y - 1;
   if (y < -100) {  y = 200;  }
}
```

yが-100未満ならば200に戻す





【演習】ボールが見えなくなったら位置をリセット

■Processing入門7: ブラウザで実行するには

- HTML形式のファイルにそのままテキストとして記述することで実行可能
  （JavaScriptとして実行）
- 画像ファイルも利用可能（画像名の指定、画像ファイルの設置が必要）
- スマホ（iPhoneやAndroid）のブラウザでも動作可能

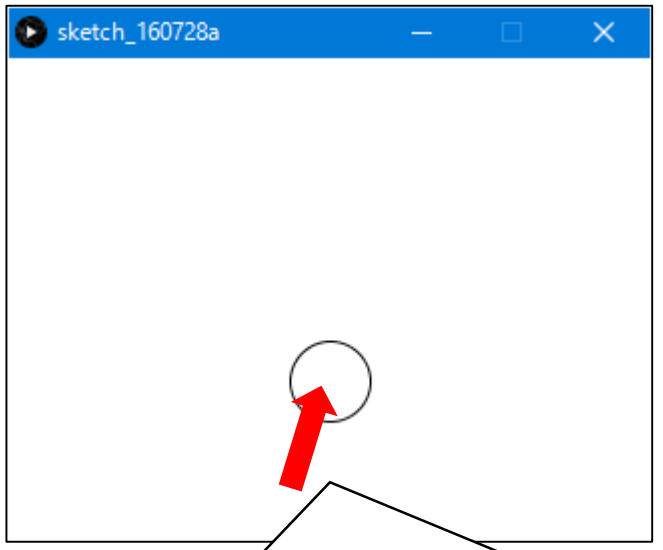
【演習】ブラウザ版で実行

①サンプルのキット（webtest.zipファイル）をダウンロードし
　デスクトップに解凍。index.htmが編集対象。
②該当箇所にProcessingのプログラムをコピー＆ペーストし保存
③index.htmをブラウザで実行（※IEは動作しない）

画像(ball.png)を用いている場合は
①画像ファイルをフォルダ内に置く
②/* @pjs preload="ball.png"; */
　を1行目に追加

■Processing入門8: モード管理

```
int  s;      // 状態s（1:オン  0:オフ）
int  x, y;
void  setup() {
  size(320, 240);
  x = 160;  y = 200;    s=0;
}
void  draw() {
  background(255,255,255);
  image(img, x, y, 40,40);
  if (s==1) { y = y - 1; }   // 上に移動
  if (y < -100) { y = 200;  s=0; }
  // 円内でクリック？
  if ( abs(x-mouseX)<20 &&  abs(y-mouseY)<20
       &&  s==0 &&  mousePressed) { s=1; }
}
```



円内でクリックすれば上に進む
||
|x-mouseX|<20 かつ |y-mouseY|<20 かつ
s=0 かつ クリックした ならば
s: 0 → 1 に変更

【演習】円内でクリックしたら上に移動

■まとめ

●プログラミング教育について

✓今後、情報の専門家以外にもプログラミング教育が必要な時代に

✓情報を教えることができる教師の不足

→情報教育ではできるだけ簡易な言語を利用することで補えないか？

たとえば、Processing

→高校～大学1年生までのカリキュラムを共通化（共通の教材）すれば対処できる？

●導入教育向きのプログラミング言語（Processing）について紹介

✓インストールなどの導入が簡単、おまじないが少ない

✓グラフィックスのプログラムは理解しやすいし興味も持たれやすい◎

✓少し高度なことをしようとすると、プログラミングの構文の理解が必須△

配列、繰り返し、処理の関数化、クラス・オブジェクトの概念等