プログラミング基礎

アニメーション（応用編）

＋

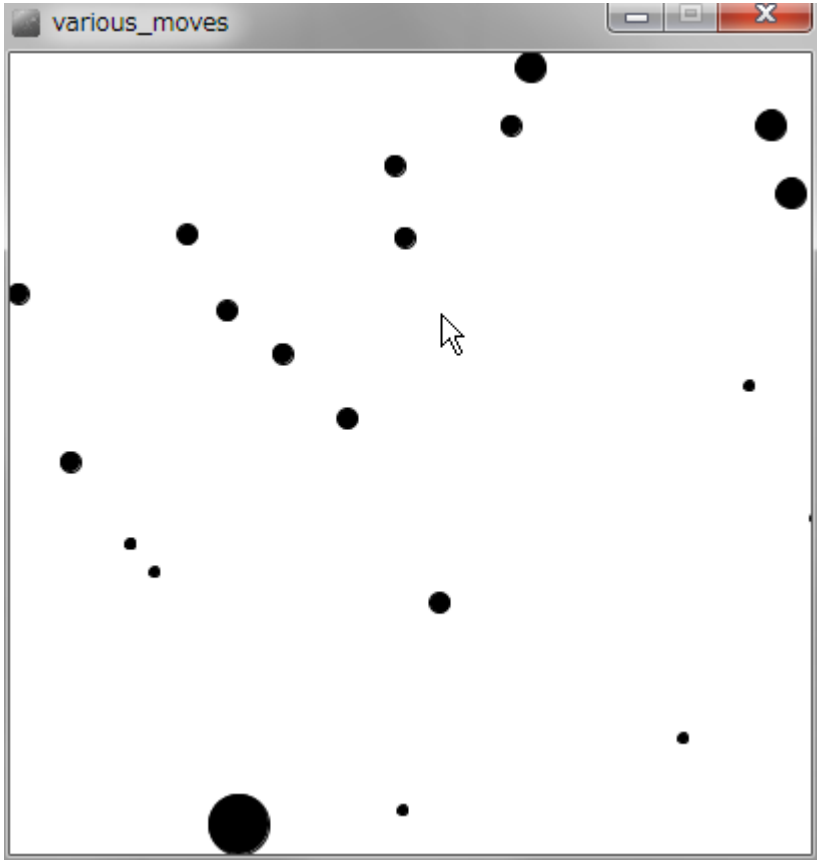
オブジェクト指向プログラミング（後編）

今回の内容

- 動くものをクラスで記述する
- いろいろな動きをクラスで記述する:
  クラスの継承
- 動くものを増やしたり減らしたりする
  - ArrayList

今回のサンプル:いろいろな動き

- 講義ページからダウンロード
  - 真ん中から湧いて出る
  - ウィンドウ下端を這う
  - クリックすると・・・
  - ドラッグすると・・・



動くものをクラスで記述

動くものをクラスで記述する

- 物体の動きを表現するためのデータ
  - 場所 (x, y)
  - 速度 (vx, vy)
    （表現したい動きによっては加速度なども必要）

- 複数のデータをまとめたい　→　クラス

動くものクラス (Mover)

```
class Mover {
  float x, y;   // 位置
  float vx, vy; // 速度
  int r; // 大きさ

  Mover(int x, int y, int r) {
    this.x = x;
    this.y = y;
    this.r = r;
  }

  void setVelocity(float vx, float vy) {
    this.vx = vx;
    this.vy = vy;
  }

  void setVelocityByAngle(float v, float angle){
    vx = v * cos(angle);
    vy = v * sin(angle);
  }

  void move() {
    x += vx;
    y += vy;
  }
```

```
void draw() {
  background(255);

  // 略

  for (int i = 0; i < ms.size(); i++) {
    Mover m = ms.get(i);
    m.move();
    m.draw();
    if(m.isFinished()) ms.remove(m);
  }
}
```

（本体のdraw関数）

1回drawするごとに
速度の分だけ移動

速度の表現

- vx　x軸上の速度
  - vx > 0　右方向
  - vx < 0　左方向
- vy　y軸上の速度
  - vy > 0　下方向
  - vy < 0　上方向

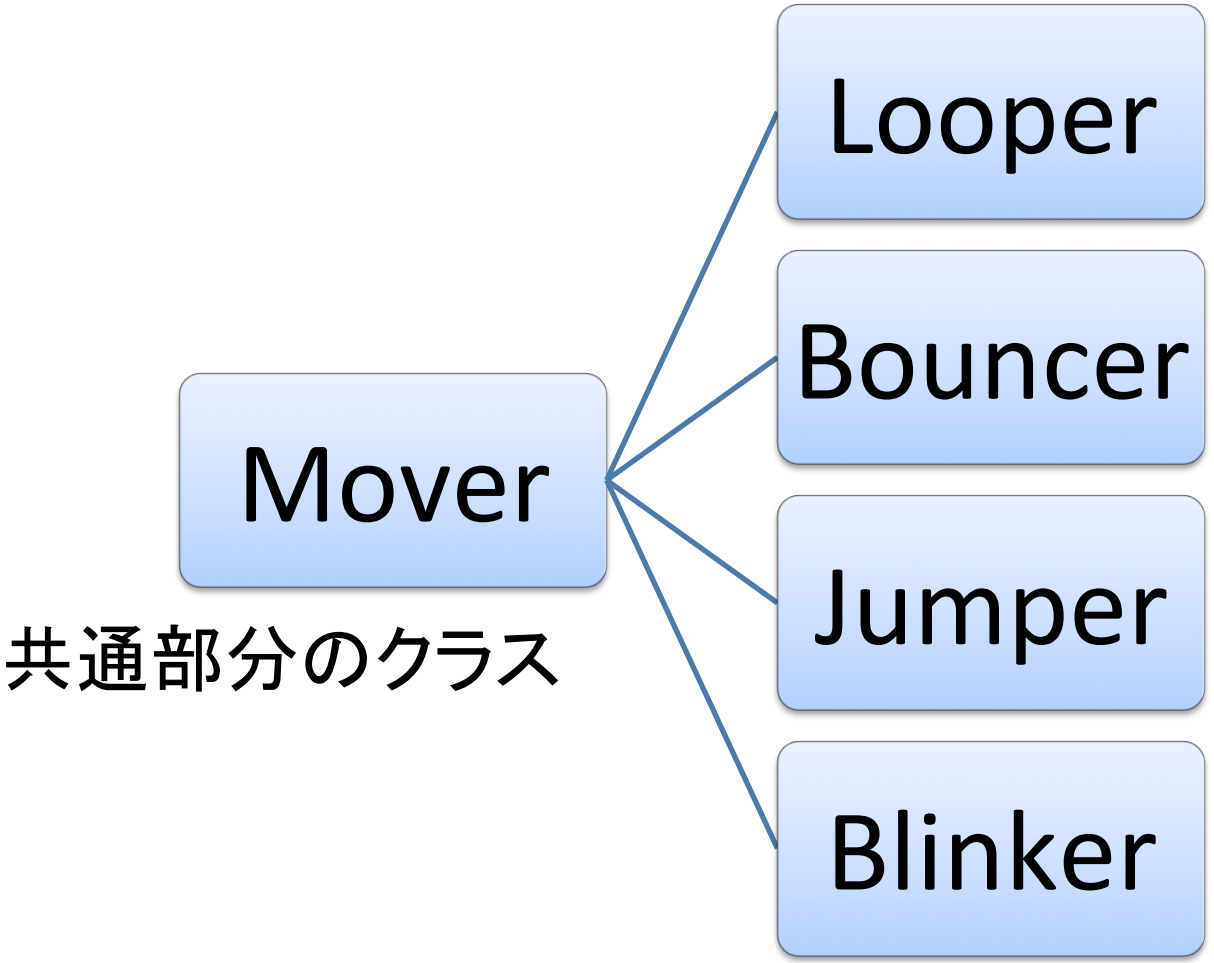
いろいろな動きをクラスで記述する

クラスの継承

- 動く物という意味では共通、動き方は色々

- クラスの継承を利用する
  1. 共通部分のクラスを作る（今回はMover）
  2. 作った共通部分のクラスを継承して、
     動きを変えた新しいクラスを作る

サンプルでのクラスの継承関係



クラスの継承

【書式】 class クラス名 extends 親クラス名

```
class Looper extends Mover {
  Looper(int x, int y, int r) {
    super(x, y, r);
  }

  void move() {
    super.move();
    x = x % width;
    if (x < 0) x = x + width;
    y = y % height;
    if (y < 0) y = y + height;
  }
}
```

親クラスのコンストラクタを利用

親クラスの
move 関数を利用

move 関数を
上書きして、
動き方を変更

画面端での動きが違う（Looper & Bouncer）

Looper

- 反対側から出てくる

```
x = x % width;
if (x < 0) x = x + width;
y = y % height;
if (y < 0) y = y + height;
```

Bouncer

- 跳ね返る

```
// Make sure that it is inside the window
x = constrain(x, 0, width);
y = constrain(y, 0, height);

// Bounce at window edge
if(x == 0 || x == width){ vx = -vx; }
if(y == 0 || y == height){ vy = -vy; }
```

重力・ジャンプ（Jumper）

1. 重力
   - 1回動くごとに vy に重力分を足す（重力加速度）
   - vy > 0 （下方向）になると落ちてくる
2. 着地＆ジャンプ
   - y 座標が地面（height）に到達したら vy < 0 （上方向）にする

Drawを上書き（Blinker）

```
class Blinker extends Looper{
  Blinker(int x, int y, int r){
    super(x, y, r);
  }

  void draw(){
    if(frameCount / 20 % 2 == 0){ fill(0); }
    else{ fill(255); }
    ellipse(x, y, r, r);
  }
}
```

動く物を増やしたり減らしたりする

ArrayList

- 配列は、増えたり減ったりするときには不便
  - 最初に個数を決めないといけないし・・・

- 代わりに ArrayList を利用する

ArrayList の使い方

```
ArrayList<Mover> ms;
```

```
ms = new ArrayList<Mover>();
```

個数を先に
決めなくていい

```
Mover[] ms;
ms = new Mover[5];
```

（参考：配列の場合）

```
ms.get(i);
```

```
ms.add(m);
```

```
ms.remove(m);
```

追加・削除するときには
何番目かを考えなくてもいい

動いている物を増やす

- ArrayList に add することで実現

- サンプルプログラムに3パターンの例
  - 最初に1回追加（setup関数）
  - 自動的に繰り返し追加（draw関数）
  - ユーザの操作に反応して追加

動いている物を消す

- ArrayList から remove することで実現

```
boolean isFinished(){
  return ! isInTheWindow(); // 画面外に行ったら消える
}
```

（Moverクラス）

```
void draw() {
  background(255);

  // 略

  for (int i = 0; i < ms.size(); i++) {
    Mover m = ms.get(i);
    m.move();
    m.draw();
    if(m.isFinished()) ms.remove(m);
  }
}
```

消えるときには true
消えないときには false
となるように作っておく

練習

- 好きな動きを作ってみましょう
- 思いつかない人向け
  - マウスを押しているとだんだん加速する
  - クリックされたやつは消える
  - ドラッグ後に離したときに、ドラッグした方向に1つだけ発射する
  - 【難】 動いている物同士がぶつかると消える
- 次スライドの「応用例」にも挑戦してみよう

応用例

- 自然現象系アニメーション
  - 雪、落ち葉 など
  - randomを使うとそれっぽい？
- シューティングゲーム
  - 直線的な動きばかりだとつまらない？

- もう少し真面目にやるとシミュレーションに