Media Programing 1 - Processing

- frame.03 - 2010.05.11 hrr

## 今回のテーマ

キーボード、マウス復習
オブジェクト指向プログラミング
衝突判定、跳ね返り
ソーティング（並び替え）アルゴリズム

## キーボード＆マウスによる操作

キーボード入力イベントを扱うための関数（メソッド）として keyPressed() を使う。
マウスと同様にキーボードを押すと、この関数が自動的に呼ばれる。
どのキーが押されたかの情報は、引数によって得られる。
特殊なキーについては、OF_KEY_LEFT, OF_KEY_DOWN, OF_KEY_RIGHT,
OF_KEY_UP などのように特別な名前がつけれている。詳しくはドキュメントを参照。

```
--
   void testApp::keyPressed(int key){

      if(key == 'x'){
         ofDrawBitmapString("xx" , 100, 100);
      } else if (key == OF_KEY_UP){
         ofDrawBitmapString("UP", 200, 200);
      }
   }
--
```

**授業内課題1：**

ウインドウの下の方に、ラケットとなるような長方形を描画し、キーボードの
カーソルキーで移動出来るようにする。
Sikakuクラスのサブクラスとして、Itaクラスを作成して実現すると良い。
右に移動するメソッド、左に移動するメソッドの追加（Zukeiクラスに追加しても可）、
板を描画するpaint()のオーバーロード、コンストラクタの作成で、ほぼ完成する。
できれば、ウインドウをはみ出ないように移動を制限すること。

## クラスの配列

クラスの配列を用いてたくさんの図形を描画する。
まず前回のZukeiクラスとSikakuクラスを整理と機能追加する。

**ファイルの分割**

通常クラスを記述したファイルは別ファイルにする。
すべてのファイルを別ファイルにしても良いですが、まずはZukeiクラスや
Sikakuクラスを別ファイルにしてみよう。

ファイル名： Zukei.h

---

```
#include "ofMain.h"


class Zukei {

protected:
   float x;
   float y;
   ofColor c;
   float speed;

public:
   Zukei();
   Zukei(float x, float y);

   void setRandom();
   void ten();
   void ten(float x, float y);
   void moveH();

};

class Sikaku : public Zukei{

   float d;
   bool isFill;

public:
   Sikaku();

   void paint();
};
```

ファイル名：Zukei.cpp

---

```
#include "Zukei.h"
```

```
Zukei::Zukei(){
      speed = 1.0;
   x = ofRandom(0, 400);
   y = ofRandom(0, 400);
   c.r = ofRandom(0, 255);
   c.g = ofRandom(0, 255);
   c.b = ofRandom(0, 255);
   c.a = ofRandom(0, 255);
}

Zukei::Zukei(float x, float y){
   speed = 1.0;
   this->x = x;
   this->y = y;
   c.r = ofRandom(0, 255);
   c.g = ofRandom(0, 255);
   c.b = ofRandom(0, 255);
   c.a = ofRandom(0, 255);
}

void Zukei::setRandom(){
   x = ofRandom(0, 400);
   y = ofRandom(0, 400);
   c.r = ofRandom(0, 255);
   c.g = ofRandom(0, 255);
   c.b = ofRandom(0, 255);
   c.a = ofRandom(0, 255);
}

void Zukei::ten(){
   ofSetColor(c.r, c.g, c.b , c.a);
   ofRect(x, y, 10, 10);
}

void Zukei::ten(float x, float y){
   this->x = x;
   this->y = y;
   ofSetColor(c.r, c.g, c.b, c.a);
   ofRect(x, y, 1, 1);
}

void Zukei::moveH(){
   x = x + speed;
   if( x > ofGetWidth()){
      speed = speed * -1.0;
   }
```

```
}


Sikaku::Sikaku() {
   d = ofRandom(10, 100);
}

void Sikaku::paint(){
   ofSetColor(c.r, c.g, c.b, c.a);
   ofRect(x, y, d, d);
}
-----
```

この書き方は、既にtestApp.h やtestApp.cpp で見てきた物と同じ形式になる。
Zukei.h にはクラスのプロパティ（属性）となる変数とメソッド名が定義されている。
Zukei.cpp には、クラスのメソッドの中身が書かれている。
クラス名::メソッド名　{ 中身の定義}
という形式をとる。

testApp.cpp には、
#include "Zukei.h"
を加える事で、このクラスが使えるようになる。
なお、このクラスには、オブジェクトの座標値をランダムで設定するための
メソッド setRandomが追加されている。

**練習問題１：**

これでtestApp.cppがかなりすっきりした。
配列を使ってランダムに四角を１００個ほど描画するプログラムを完成させる。

**練習問題２：**

Sikakuクラスを拡張して、どのSikakuオブジェクト内にマウスポインタが
あるか判定できるようにする。
SikakuクラスにinsideメソッドをSikakuオブジェクトの色が変わるようにする。
マウスポインタのある座標値をすべてのSikakuオブジェクトに対して
insideメソッドで確認すれば、どのオブジェクトのエリア内にマウスポインタが
あるか判定出来る。
paint()メソッドにSetColorが入っている場合には、色を変更するメソッドを
Zukeiクラスに追加することで、色を変更できるようにする。

参考情報：

新しいプロジェクトにSikakuクラスを書いたファイルを追加するには、
XCodeに左側のフレームにあるsrcフォルダを右クリックして、
追加/既存のファイル　を選択し、Sikaku.h Sikaku.cpp を選ぶ。
これで追加できた。あとはtestApp.cppでincludeなどを追加する。
なお、この状態で、Sikaku.h Sikaku.cpp を編集すると元のファイルも変更される。
また、Finder でファイルを新しいプロジェクトのフォルダにコピーして、
追加しても良い。この場合には、元のファイルはそのままでコピーしたファイルが
編集される。

Sikaku.h

```
    class Sikaku{
        bool inside(float x, float y);
}
```

Sikaku.cpp

```
    bool Sikaku::inside{float x, float y){
            // 描画範囲内にあったら、true
            // そうじゃなかったら、falseを返す
    }
```

## 衝突判定と反射

ウインドウの枠に到達したら、跳ね返るようにする
既に前回、moveH, moveVメソッドの作成でウインドウの枠に達した事の判定を
行っているが、さらに斜め方向にぶつかった場合も考慮した跳ね返りを実現する。
ウインドウのサイズは、width, height 変数内に自動的に入っている。

**練習問題３：**

ランダムな方向に一定速度で四角が動くようなアニメーションを作成する。
方向を決定するのは、進行方向の角度をランダムで決定し、その値から
sin, cosを使って、x方向、y方向の変化量を決定する。
移動速度は、適当な変数、speedなど、を使って調整できるようにしておくと良い。

なお、Sikakuクラスを使うって作成すること。
Sikakuクラスを拡張して、Maru というサブクラスを作って円形を描画できるようにしてもよい。

**授業内課題２：**

練習問題３を改良し、ウインドウの枠に達したら、跳ね返るようにする。
跳ね返るときにどのような変化が起きるか？
普通は、進行方向を示す変数も必要となる。
図に描いて確認してみるとわかりやすい。

## ソーティング

頭の体操として、並べ替えの方法を考えてみる。
１０個の数値があるときに小さい順に並べ替える。
色々な方法がありますが、今回は以下のような方法を用いるとする。

1. １０個のランダムな数値が適当な順番に並んでいるとする。
   配列で考えれば、配列の０番目から９番目にランダムな数値が入っていると考えることができる。
2. 数値の０番目の値と１番目の値を比較して、０番目の値の方が大きければ、０番目の値と１番目の値を入れ替える。
3. 同様にして、１番目と２番目の比較、２番目と３番目の比較と続け、８番目と９番目まで実行する。（比較して大きければ入れ替え）
4. ここまでで、9番目に一番大きい値が入っている。
5. 同様に０番目と１番目の比較から始め、７番目と８番目まで比較していく。
   ２回目なので、比較の回数が一回減ることに注意。
6. これで、８番目に次に大きい値が入る。
7. これを、繰り返すと毎回比較の回数が減りながら、
   最後まで行くと並べ替えが終了する。

**０１２３４５６７８９**
□□□□□□□□□□

まずは、紙にでも書いて仕組みを理解しましょう。

**練習問題４：**

１０個の大きさを持つint型の配列を用意し、ランダムな値と入れる。
初期状態を出力したあと、上のアルゴリズムを使って並べ替えを行い、

結果を出力する。

---
```
int ia[10];
string str, str1;

void testApp::setup(){
    str = "before: ";
    str1 = "sorted: ";

    for(int i = 0; i < 10 ; i++){
        ia[i] = ofRandom(0,1000);
        str = str + ofToString(ia[i], 0) + " ";
    }
    // 並べ替えを実施
    // 二重ループを使う

    for(int i = 0; i < 10; i++){
        str1 = str1 + ofToString(ia[i], 0) + " ";
    }
  }

}
```
---

**授業内課題３：**

Sikakuオブジェクトを１０個使って、ソーティングの様子をアニメーションにする。
Sikakuオブジェクトの色をソートするための数値として利用して、
練習問題４と同様なソーティングをdraw()内に記述する事で、
色が並べ替える様子が見えるようにする。
ただし、この場合はループで処理しているループをはずして、draw()がフレーム毎に繰り返し呼ばれることをループの代わりに使う。
色のうち、特定の要素（RGBならRだけ、あるいは明るさ）に注目して
並べ変えれば、変化がわかりやすい。
フレームレートを低くしてゆっくりと見せれば、徐々に並べ替えられていく様子が見えるでしょう。

## 課題：

次回は、ネットワークを使って、プログラム同士でやり取りができるようにします。
これまでに習ったことをベースに、ネットワーク越しに対戦が出来るゲーム、
あるいは、ネットワークを使う事でインタラクティブに変化していく表現を考えて
次回までにメールで送る事。
また、次回以降に画像処理（リアルタイムのキャプチャなど）やグラフィックスの
描画（の基礎だけ）、音声処理（の基本だけ）などもやるので、それらを考慮した
内容としてもよい。

あまりに内容がかぶるようなら再考してもらう場合もあります。
条件としては、
・オブジェクト指向プログラミングによって実現する事。
　Zukeiクラス、またはそれを拡張したものを利用する事。
　Sikakuクラスのサブクラスを作成して、円にしたり、必要に応じて機能を
　加えていくのは自由です。
・ネットワーク越しに２つ以上のプログラムが無いと実現出来ない内容であること。
　２つ以上のプログラムは同一のものが複数であってもよいし、サーバ的なものと
　クライアントの違うプログラムが連携するものでもよい。
・キーボードやマウスなどによって変化するインタラクティブなものであること。