6.2 シミュレーションと結果の画像への出力

コンピューターシミュレーションとは、主に自然現象をモデル化し、これをコンピューターをつかって再現して見ることをいう。再現するために多大なコスト[1]がかかったり、再現すると危険であったり[2]、などというときにはコンピューター上で（安全に）再現し、知見を得るということがよくやられる。いわば、コンピューターのなかで実験をしてみるわけだ。

本節では、ランダムウォークを体験してもらう。実際に簡単なシミュレーションプログラムを作成し、先回作成したexport.c、imgutil.c、word.cを使ってこれを可視化しよう。

6.2.1 ランダムウォーク

ランダムウォークとは、日本語で「酔歩」ともいう。酔っぱらいのおじさんがあっちにゆらゆら、こっちにゆらゆら歩く様子と似ているからである。酔っぱらいのおじさんには2次元平面を歩いてもらおう。2次元平面でのおじさんの位置の時刻tでのx座標、y座標をそれぞれ$x(t), y(t)$とすると、おじさんの動きはこれらの列

$$x(0), x(1), x(2), \cdots, x(t), x(t+1), \cdots \tag{6.1}$$
$$y(0), y(1), y(2), \cdots, y(t), y(t+1), \cdots \tag{6.2}$$

をみればわかる。実は、このランダムウォークはBrown運動などの物理系や株価の動きなどの経済分野にもみられるものである。詳細はこの授業の範囲を超えるので省略するが、興味がある学生は調べてみるとよい。

なお、今回はsrand関数とrand関数を使う。例えば、

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(void){
      int i;
      srand((unsigned)time(NULL));
      for(i=0; i<10;i++)
```

[1] いわゆる、ヒト・カネ・モノ・時間を指す。
[2] 事故究明のために原因の再現をするなど。

```
        printf("%d\n",rand()%4);
}
```

とすれば、0から3がランダムに並んだ列が生成される。srand関数は乱数の初期化のための関数であり、引数は乱数を生成させるもとになる数値（seed)である。この例では実行されたタイミングによって異なる乱数を生成するようになっている。詳細は省くが、そのまま使って頂いてかまわない。また、rand関数は整数値の乱数を生成する関数である。いまは%4としているので4で割った余りである0から3までの値が出力されるのである。

以下はこれを参考にせよ。

3人のおじさんのランダムウォーク（家に帰る気がないバージョン）

3人のよっぱらいのおじさんがとにかくその場でふらふらしている場合を試してみよう。このとき、$x(t)$と$y(t)$の更新則は以下のとおりであるとする。rand()%3の値を$r0$とし、再度実行したrand()%3の値を今度は$r1$とすると、

- $r0$の値が0のとき、

$$x(t+1) = x(t) + 1 \tag{6.3}$$

- $r0$の値が1のとき、

$$x(t+1) = x(t) - 1 \tag{6.4}$$

- $r0$の値が2のときは

$$x(t+1) = x(t) \tag{6.5}$$

- $r1$の値が0のとき、

$$y(t+1) = y(t) + 1 \tag{6.6}$$

- $r1$の値が1のとき、

$$y(t+1) = y(t) - 1 \tag{6.7}$$

- $r1$の値が2のとき、

$$y(t+1) = y(t) \tag{6.8}$$

ただし、`x(t+1)`または`y(t+1)`の値が負になったり、上限を超えた場合は時刻tのときの座標から変えないものとする。

以上を踏まえて、これをプログラムとして実装しよう。$x(t)$および$y(t)$はint型であるとする。おじさんの位置はx座標とy座標の組で指定できるから、次の構造体POINTを定義する。この構造体ではおじさんに割り当てる色(RGB)も保持できるようになっている。

```
struct point
{
  int x;
  int y;
  unsigned char r;
  unsigned char g;
  unsigned char b;
};

typedef struct point POINT;
```

関数としては以下を実装する。以下、`pointArray`は`POINT`型を要素に持つ配列、`totalPointNum`はおじさんの人数、`initX`、`initY`はおじさんの最初の位置のx座標、y座標をそれぞれ表す。

- void init(POINT *pointArray, int totalPointNum, int initX, int initY);

  POINT型の要素をtotalPointNum個もつ配列としてpointArrayを`malloc`関数を用いて初期化されていることを前提とする。次にpointArrayのすべての要素の`x`と`y`の値にそれぞれ`initX`、`initY`をセットする。`r`、`g`、`b`の値は、i番目のおじさんに対して、$i\%3$の値が0のとき赤、1のとき青、2のとき緑であるとする。`srand`関数を用いて乱数の初期化もここで行うこと。

- void move(POINT *pointArray, int i,int w, int h);

  配列pointArrayのi番目の要素(i番目のおじさん)に対し、上記の更新則による移動操作を行え。`w`と`h`は画像の幅と高さを与える。

- void drawPoints(POINT *pointArray, int w, int h, int totalPointNum, int turns);

  まず、以下の様なIMAGE構造体[3]を型にもつポインタ変数 imgを用意する(ここでのw, hは引数にあるもの)。

```
IMAGE *img=(IMAGE *)malloc(sizeof(IMAGE));
img->width=w;
img->height=h;
img->depth=24;
img->pixels=(PIXEL *)malloc(img->width*img->height*sizeof(PIXEL));
```

  まず、一度、全ての画素（ピクセル）を白色[4]にする。

  おじさん（pointArrayの各要素）ひとりひとりに以下の操作を行う。すなわち、あるおじさんについて、まず上で作ったmove関数で移動させ、おじさんの位置に対応するピクセル[5]に対し、それぞれおじさん（要素）のr、g、bの値をセットせよ。これが終わったら次のおじさんに同じ処理を行う。この操作を、おじさん全員分（全要素分）に対し行う。

  以上の操作をturns回繰り返せ。

  この結果をrandomWalk.bmpという名前のビットマップ画像ファイルに書き出せ。書き出しの際には、先回作成したsaveImage関数を利用すること。

[3]先週作成した。
[4]RGBでは赤(r)、緑(g)、青(b)の全てが255であるとき白色になる。
[5]先回作成したgetLabel関数を活用せよ。

これらの関数の実装をファイルrandomWalk.cに記述しよう。また、これらの宣言を次に示すヘッダファイルrandomWalk.hで行う。

```
#ifndef RANDOMWALK_H_
#define RANDOMWALK_H_

#include <time.h>

struct point
{
  int x;
  int y;
  unsigned char r;
  unsigned char g;
  unsigned char b;
};

typedef struct point POINT;

void init(POINT *pointArray, int totalPointNum, int initX, int initY);
void move(POINT *pointArray, int i, int w, int h);
void drawPoints(POINT *pointArray, int w, int h, int totalPointNum, int turns);
#endif /* RANDOMWALK_H_ */
```

また、実行すべきmain関数は以下のとおりであるとする。このファイルをsim.cとし、このファイルを、先回作成したexport.c、imgutil.c、word.c、今回作成したrandomWalk.cとともにコンパイル・リンクし、実行ファイルsimを作成し、実行してみよ。

```
#include "randomWalk.h"
#include "imgutil.h"
#include "export.h"
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv){

  int totalPointNum=3;
  POINT *pointArray=(POINT *)malloc(totalPointNum*sizeof(POINT));
  int width=500;
  int height=500;
  int initX=width/2;
  int initY=height/2;
  int turns=100000;

  init(pointArray, totalPointNum, initX, initY);
  drawPoints(pointArray, width, height, totalPointNum, turns);
}
```