連載

Processingによるプログラミング教育

菊池　誠
（大阪大学サイバーメディアセンター）

✔ 授業の位置づけと変遷

大阪大学では，1年生を主な対象とする一般教育科目として，年度後半に「情報処理教育科目」というカテゴリーの科目がいくつか開講される．筆者が担当する「計算機シミュレーション入門」は，その中でプログラミングを学ぶ唯一の科目である．この授業では，プログラミングの初歩から始めて，グラフィクスを使ったアニメーション，そして乱数やセルオートマトンを使う簡単な計算機シミュレーションまでを扱う．例年，文系・理系を問わずさまざまな学部の学生が50名程度受講しており，そのほとんどはこの授業で初めてプログラミングを経験するという学生である．1セメスター（学期）分にしては少々欲張りな内容なので，プログラミングの初歩をどれだけ効率的に教えるかが工夫のしどころとなる．

3年前まで，この授業ではC言語を用い，グラフィック・ライブラリとしてEGGXを利用していた．山内千里氏によるこのライブラリは，初心者にも使いやすく，機能も十分で，これのおかげで授業が構成できたといっても過言ではない．

そうはいっても，C言語をきちんと教えるとなると，それだけでかなりの時間がかかる．そこで，「スーパー・サバイバルC」と銘打ち，シミュレーションに必要な最低限のC言語だけに内容を絞った．プログラミングの専門家には怒られそうだが，関数の作り方・ポインタ・構造体などは教えない．これらの概念は初心者には理解しづらい上，知らなくてもセルオートマトンのプログラムくらいは書けるからである．しかし，それだけ内容を絞っても，限られた授業時間の中で，まずC言語の基礎，次にグラフィクスという流れでは急ぎ足にならざるを得ない．もっと簡単にできないかと考えていたとき，Processingに出会った．

✔ プログラミング言語としてのProcessing

Processingは言語であると同時にJavaで書かれた開発・実行環境でもある（各OS用の実行ファイルが http://processing.org/ から入手できる）．言語仕様はJavaの簡易版となっており，オブジェクト指向である．もともとメディア・アート向けに作られた言語であるため，グラフィクスが簡単に操作できることが最大の特徴となっている．ダブルバッファリングを使うアニメーションもバッファを意識せずに実現できる．そして，後述するように，プログラミング入門用に適した言語であると筆者が考える理由もそこにある．

Processingのプログラムは実行時にJavaに翻訳されるので，Javaのプリプロセッサとも言えるが，実際にはJavaであることを意識する必要はほとんどなく，独立した言語と考えておけばよい．ただし，必要であればJavaのライブラリも使え，また，できあがったプログラムをJavaアプレットに変換する機能も用意されている．

実際の使用イメージを**図-1**に示す．開発環境は実際にはエディタだけと言ってよく，デバッガなどの機能はないが，言語そのものが簡単なので，困ることはあまりないだろう．エディタでプログラムを書き，実行ボタンを押せば，グラフィクス用のキャンバスが開いて実行が開始される．

言語としてのProcessingは，2Dと3Dのグラフィクスを記述するための豊富な関数群とアニメーシ

ョンのフレームワークが用意されていることを除くと，簡単であるという以上には特徴があるわけではない．しかし，サウンドやビデオなど，機能を拡張するさまざまなライブラリが作られていて，画像と音を組み合わせたインタラクティブなプログラムも手軽に書くことができる．また，ArduinoやGainerなどのフィジカルプログラミング用ライブラリもあり，実際，筆者がProcessingを知ったのは，Gainerのためのプログラミング・ツールとしてだった．

✔ Processingによるプログラミング入門

プログラミング学習の初歩は往々にして抽象的になりがちである．わけも分からずやみくもにサンプル・プログラムを入力してみると，謎の数字が1個だけ出力される，というのが典型的な姿だろうか．この授業では，その最初の段階を「図形」という具体的な対象の操作に対応づけることで，理解しやすくしようと考えた．たとえば，かのK&R以来，最初のプログラムは "Hello, world" と相場が決まっているが，この授業での最初のプログラムは

```
line(0,0,100,100);
```

である．この1行のプログラムでウィンドウが開き，キャンバスに斜線が描かれる．ヘッダーやmain()などの準備は不要で，いきなり実行文を書けばよい．

続いて，簡単な図形を描くプログラムを作る．たとえば

```
fill(100,0,0);
rect(0,0,50,50);
fill(0,200,100);
rect(30,30,50,50);
```

という4行のプログラムは，それぞれ(1)色を決める(2)正方形を描く(3)色を変える(4)別の位置に正方形を描く，という4つの操作と1対1に対応している．#include<stdio.h>やmain()といった「おまじない」を必要としないのが，初心者に優しい点である．さらに，図形を組み合わせて，各自，好きな絵を描いてみる．座標をすべて数値で表すのは大変に思えるが，結果が即座に絵として確認できるので，学生

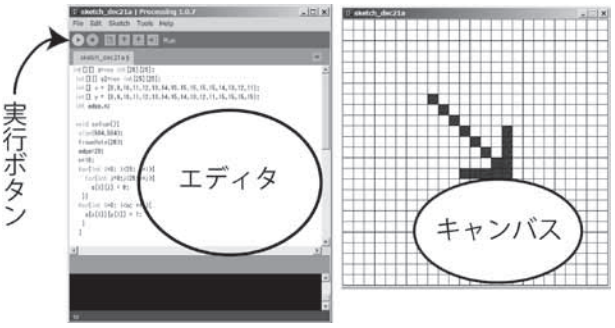

図-1

も楽しんでやれるようだ．

では，変数や配列などの抽象的な概念やループ・分岐といった実行制御はどう教えるか．これらも最初はすべて図形やアニメーションの操作と関連させる．たとえば，同じ図形をさまざまな位置に描こうとすれば変数を使うのが便利だし，格子を描いたり同じ図形を規則的に並べたりするためにはループを使うという具合である．

配列は，まずは碁盤のような升目を表現する方法として導入する．升目に色をつけて絵や文字を描き，それを1ステップにつき1マス分だけずらしてゆくアニメーションを作れば，電光掲示板もどきが作れる．これを2次元配列の操作で実現するわけである(図-1および**付録**参照)．また，これはその後に学ぶセル・オートマトンの簡単な例にもなっている．

このようにして，まずは目に見える対象の操作として，プログラムのさまざまな概念を教え，それから数値の操作など目に見えない対象の操作へと進めてゆくというのが，この授業のポイントである．なお，授業ではシミュレーションに必要な最低限のプログラミングしか扱わないので省略しているが，オブジェクトやクラスも図形という具体的なオブジェクトと関連づけて導入すれば，初学者が概念を把握しやすいだろう．

✔ 授業の効果

毎回の授業では，最初に15分ほど解説をし，あとは各自に実習してもらう．もちろん，プログラミング経験も適性もばらばらな学生たちなので，全員が同じ進度では進まない．解説は標準的な進度

の目安を与える程度のものと位置づけ，実際にはe-learningシステムで与えられる教材に基づいて，各自が自分なりの進度で学習を進めるようにしている．それを筆者と3名のTAとで指導してゆく．

Processingの学習がひととおり終わると，次は乱数やセル･オートマトンを使ったシミュレーション･プログラムに移るのだが，詳細は割愛する．

では，1セメスターでどの程度のプログラミングができるようになるのか．学生には，最低限の到達目標として，アニメーションのプログラムが書けるようになることを求めている．残念ながら，セルオートマトンまで進めるのは全体の半分程度だが，乱数を用いて偶然性をアニメーションに導入する（たとえば，花びらがひらひらと舞う様子を乱数で表現するなど）程度までは，ほぼ全員が到達できる（TAの努力のおかげで，ついていけずに諦める学生は稀である）．学期の最後には自由制作を提出してもらう．課題は，各自の到達度に応じて，アニメーションまたはシミュレーションのプログラムとしており，毎年独創的な力作が揃う．簡単なものでは数10行程度で，100行を超えるプログラムも少なくない．スタート時点ではプログラム未経験者がほとんどだったことを思えば，悪くないのではないだろうか．

興味のあるかたは，筆者のWebサイトに講義資料を公開しているので，ご覧いただきたい (http://www.cp.cmc.osaka-u.ac.jp/~kikuchi/kougi).

（平成22年8月31日受付）

菊池　誠　kikuchi@cmc.osaka-u.ac.jp
1986年東北大学大学院理学研究科物理学専攻修了（理学博士）．大阪大学理学部助手，助教授を経て，2000年より大阪大学サイバーメディアセンター教授．専門は計算物理学・統計物理学・生物物理学．

付録

参考までに，実行例として図-1に示したプログラムを掲載する．これは，2次元配列の練習のために用意した「電光掲示板もどき」である．

```
int[][] s=new int[25][25];
int[][] s2=new int[25][25];
int[] x = {8,9,10,11,12,13,14,15,15,15,15,
15,14,13,12,11};
int[] y = {8,9,10,11,12,13,14,15,14,13,12,
11,15,15,15,15};
int edge=20,n=16;

void setup(){
 size(504,504);
 stroke(0);
 frameRate(20);
 for(int i=0; i<25; ++i){
   for(int j=0;j<25;++j){
     s[i][j] = 0;
  }}
 for(int i=0; i<n; ++i){
   s[x[i]][y[i]] = 1;
  }
 }

void draw(){
  for(int i=0; i<25; ++i){
    s2[i][0] = s[i][24];
    for(int j=1;j<25;++j){
      s2[i][j] = s[i][j-1];
   }}
  for(int i=0; i<25; ++i){
    for(int j=0;j<25;++j){
      s[i][j] = s2[i][j];
   }}
  for(int i=0; i<25; ++i){
    for(int j=0;j<25;++j){
      if(s[i][j] == 0){
        fill(255);}
      else{
        fill(0,0,255);}
      rect(i*edge,j*edge,edge,edge);
   }}
}
```