原著論文

初学者教育用Javaプログラミングライブラリ「Basic Graphics」

マッキン ケネス ジェームス*

要旨：Logo言語で提案されたタートルグラフィックスは，グラフィックスがプログラミング教育に非常に効果的であることを示した．以降，多くの教育向けプログラミング言語では様々な簡易的グラフィックス機能が提案されてきた．Java言語においても，タートルグラフィックスライブラリや，Processing言語のようなグラフィックス機能の拡張が提案されている．しかし，これらはオブジェクト指向の知識が必要または，専用のグラフィックス機能が高度すぎるという問題がある．本研究では，Java言語初学者の学習意欲とアルゴリズムの理解度向上を主眼に，Java言語用簡易グラフィックス機能BasicGraphicsを提案する．提案手法では，オブジェクト指向，スレッド処理，イベント駆動などをユーザに意識させずに，グラフィックス機能を提供する．提案ライブラリを用いることで，オブジェクト指向未学習者に簡易的なグラフィックスを用いて学習意欲を高めながら，その後のオブジェクト指向グラフィカルプログラミングへスムーズに移行させることが可能となる．

キーワード：Java言語，非イベント駆動，グラフィカルユーザインタフェース，アニメーション

Basic Graphics: Java programming library for beginner learners

Kenneth James MACKIN*

Abstract: Turtle graphics in the Logo programming language successfully showed that graphics has positive effects on programming education. Many educational use programming languages has followed suit and implemented various graphics features. Turtle graphics for the Java programming language has been previously proposed, and the Processing language also aims at providing graphics features to Java. These previous methods either require object-oriented programming knowledge, or knowledge of a large set of completely different graphics methods. In this research, a simple graphics library for Java in the early learning stage is proposed. The proposed library does not require understanding of object-oriented programming, thread operations and event handling. The proposed graphics library enables beginner learners to easily use graphics in order to assist the understanding of basic programming algorithms and improve learning motivation.

Keywords: Java language, non-event driven, graphical user interface, animation

* 東京情報大学　総合情報学部
Faculty of Informatics, Tokyo University of Information Sciences

2021年10月27日受付
2022年２月２日受理

1．はじめに

近年，コンピュータの応用範囲の拡大とその需要の高まりにより，あらゆる業種でコンピュータが利用されるようになった．大学におけるコンピュータ教育も，情報系の学部・学科に限らず，ほぼ全ての学問分野においてもコンピュータ利用の教育が行われるようになった．それに伴い，コンピュータソフトウェアのプログラミング教育の重要性が増している．

社会的なニーズの変化に合わせ，新学習指導要領では，高校でのプログラミング教育が必修化される．それに従い，従来のように大学などの専門的教育の一部としてのプログラミング教育と比べ，より幅の広い層に対するプログラミング教育の方法論が必要になると予想される．プログラミングは，特に初学者に取って，初期段階の学習が難しい点が多くある．

プログラミング教育は実践型教育が効果的であり，一般的なプログラミング講座には講義やテキストに加えて，実際にプログラムを組む実習と組み合わせて実施される．これは，知識としての理解だけではなく，実際に何度も繰り返しプログラムを組むことで，プログラミングの共通パターンや基本的な構造の理解を促すためである．またプログラムを実際に組み，実行することにより，本人の理解が正しかったかが本人が確認できるのも実習の利点である．

しかし，プログラミング教育，特に初期段階の学習の問題点としては，プログラムの挙動やアルゴリズムの仕組みが理解できないために足踏み状態になり，それ以降のプログラミングの内容についてこられなくなることがある．また，理解が進まないことが学習者のストレスになり，学習意欲そのものが落ちてしまうことも問題となる．

プログラミング教育では繰り返し学習が重要であるが，学習意欲を失った学習者は繰り返し学習を怠るため，理解が進まず，結果的にプログラミングが身に付かない，という悪循環に陥ることになる．

また，多くの大学の情報系学科では，ソフトウェア業界で中心的に利用されているC++/C#言語やJava言語などのオブジェクト指向型の強力なプログラミング言語の教育を行っている．このようなオブジェクト指向型プログラミングでは，難易度の高いオブジェクト指向の考え方を教育する必要があるが，その前提として先に述べたプログラミングの基本的な考え方が身に付いている必要がある．

このように，プログラミング教育の重要性がより一層高まる中，本論文では，プログラミング教育の初期学習の段階において，学習者の興味と学習意欲を高めることを主目的とし，Java言語の教育に用いるための簡易グラフィックスライブラリを提案する．提案する簡易グラフィックスライブラリの特徴や仕組みについて説明し，授業においての実践結果について報告する．

2．プログラミング教育におけるグラフィックスの利用

コンピュータグラフィックス機能をプログラミング教育に利用することは，古くから提案されている．

1967年に開発されたプログラミング教育言語Logoに含まれる，線画をわかりやすく生成するためのグラフィックス機能タートルグラフィックス（Turtle Graphics）[1]の登場はその後にも大きな影響を与えた．タートルグラフィクスの開発したシーモア・パパート（Seymour Papert）は，これを身体同調的推論（body-syntonic reasoning）と呼び，画面上に描かれるタートル（亀）のキャラクターに学習者が自身を投影しながら，プログラムによるタートルの挙動を予想・理解することでプログラミング教育を目指した．

タートルグラフィックスは，MITメディアラボが2003年から開発を続けるScratch言語など，現在でも多くの教育用プログラミング言語にも採用されている．

他にも，初学者用プログラミング言語として1964年に開発されたBASIC言語も，対応するハードウェアの広がりと共に，出力装置がラインプリンタからCRTディスプレイへと進化した1970年代には，文字の表示位置を指定する機能や，線画を描画する機能が追加され，グラフィックス機能を得た．初学者向けの単純な文法と簡易的なグラフィックス機能により，BASIC言語は1970年代から1980年代のホームコンピューターでの圧倒的なシェアを誇った．

機械式計算機とは違い，電子コンピュータでは，プログラム実行時の計算過程はメモリとCPUの間のデータ交換で実現されているため，計算過程は視覚的には何も表れない．特に初期のプログラミング教育で必要となる制御構造や基本アルゴリズムを理

解するための課題においては，出力結果を文字出力として表示するだけのプログラムが多い傾向にあり，学習者の理解度や学習意欲を高めるのには不利に働いている．

タートルグラフィックスを筆頭に，初期のプログラミング教育からグラフィックス機能を活用し視覚的表現を用いて，学習者の理解度や学習意欲を高めることが報告されている[2][3][4]．しかしタートルグラフィックスを用いたグラフィックスプログラミングは，幾何学の学習には効果があることが報告[5]されているが，タートルグラフィックスを一種の言語サブセットあるいはミニ言語[6]として追加で学ぶ必要があり，またアルゴリズムの視覚化において汎用的に利用することは難しい．

また，2013年にCode.orgにより始められたHour of Codeは，初等教育をターゲットにしたブロックプログラミングを用いて，タートルグラフィックスのタートルと同じく，キャラクターを画面上に移動させることで，プログラムの動作を視覚的に表現している．しかし，Hour of Codeにおいては，学習者の学習意欲向上には効果があるが，プログラミング言語の機能が限定されすぎているために，プログラミングスキル向上には効果が少ないとの研究報告[7]もある．

つまり，理想的なプログラミング教育向けのグラフィックス機能は，プログラミング教育の初期段階である学習者にとってプログラミングの学習意欲を高めると同時に，アルゴリズムの理解などのプログラミングスキルを向上させることにつながることが望まれる．

3．簡易グラフィックスライブラリ BasicGraphics

3．1　概要

本論文では，Javaプログラミング教育向けの簡易グラフィックスライブラリを提案する．本提案では，機能性をあえて絞った簡易グラフィックス機能により，使いやすさと，従来のテキストベースのプログラミング教育との親和性を最優先とした．

本論文で提案するJava言語用簡易グラフィックスライブラリBasicGraphicsは，既存の教育用グラフィックス機能と比較して，以下の特徴がある．

1）BasicGraphicsをJava言語用のライブラリとして実装しているため，現在システム開発で広く利用されているプログラミング言語であるJava言語の教育の導入にそのまま利用できる．

2）インタプリタ型BASIC言語のグラフィックス文法を参考に実装しており，文字の描画や線画，キー入力などを非同期イベントハンドリングやウィンドウオブジェクト操作をまったく意識せず，非常に簡単に扱うことができる．

3）テキストを用いた簡易グラフィックス機能を用いることで，様々なアルゴリズムの視覚化やアニメーション化が複雑な処理を用いずに容易に可能となる．

他にも，Java言語上で動作する簡易グラフィックスライブラリは報告[8][9]されているが，これらのライブラリは，あくまでもオブジェクト指向を教育するためのグラフィックスライブラリであり，プログラミングを初めて学ぶ初学者の教材にはそぐわない．

BasicGraphicsはオブジェクト指向をほぼ全て隠蔽し，簡単なBASIC文法を用いているため，タートルグラフィックスで可能な線画に加え，文字や画像の配置，キーやマウス入力，サウンド再生などもサポートしており，より汎用的なGUI機能を提供することにより，幾何学のみならず，アルゴリズムの学習などの視覚表現にも応用することができる．

同じような考えで，Processing言語[10]も，Java言語をベースにグラフィックス機能をより簡単に利用できるようにしている．しかし，Processingでは，従来のプログラミング教育で用いられるキーボードからの文字列入力を簡単に扱うことができない．また，Processingは独自のIDEを用いているため，標準出力はグラフィックスの表示領域とは別のせまいコンソール表示領域に表示するなど，従来のコンソールベースのテキストプログラミングとは相性が悪い．さらに，Processingは一部独自の文法や作法を用いているため，Processingを利用するための学習が必要であり，またJava言語への移行にもハードルがある．

BasicGraphicsは，Java言語のライブラリとして実装されているため，従来のJavaの学習過程にシームレスに導入でき，また非常に少ない命令で簡易グラフィックスを実装できるため，新たに学習を行う必要はない．

本章では，BasicGraphicsの基本設計，およびBasic

Graphicsを用いた以下の機能について説明する．

1）ウィンドウ表示およびテキスト描画
2）テキストアニメーション表示
3）キーボードからの文字列入力
4）バブルソートのアニメーション表示例

3.2　BasicGraphicsの基本設計

基本設計の詳細を説明する前に，BasicGraphicsを用いたHelloWorldプログラムのコード例を示す．

```
import jp.ac.tuis.basic.*;
public class HelloWorld{
  public static void main(String[] args){
    BasicGraphics g = new BasicGraphics();
    g.println("Hello, World!");
  }
}
```

上記プログラムのコンパイル・実行は，Windowsの場合は次のように行う．

```
javac -cp .;basic.jar HelloWorld.java
java -cp .;basic.jar HelloWorld
```

basic.jarはBasicGraphicsライブラリのjarファイル（Java用ライブラリファイル）である．Unix系OSの場合は，コンパイルおよび実行命令において，basic.jarの前のデリミタのセミコロン（;）がコロン（:）に変わる．

BasicGraphicsライブラリは，J2SE1.4（Java version 1.4.2）以降のバージョンのJDKあるいはJREの標準パッケージのみで実行することができる．

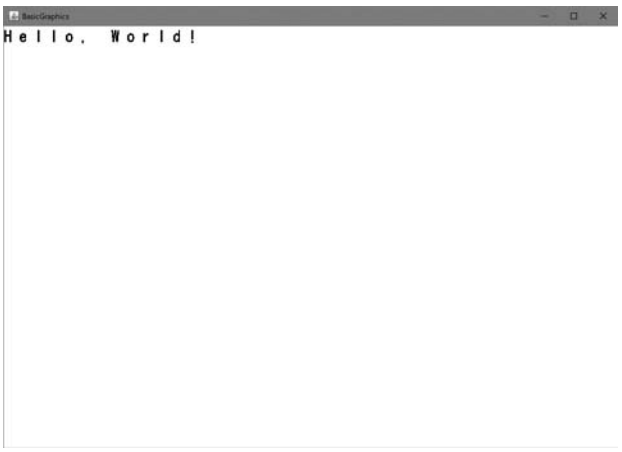

図1　プログラムHelloWorldの実行結果
Figure 1　Result of the HelloWorld program.

上記プログラムをコンパイル・実行すると，ウィンドウが現れ，ウィンドウ上にHello, World! と表示される．このプログラムは，従来のコンソールへの出力を行う同等のJavaプログラムと比べ，importでBasicGraphicsをインポートする行とBasicGraphicsをnewで生成する行の合わせて2行の追加と，標準出力への表示のSystem.out.println() をg.println() に変更しているだけである．

BasicGraphicsライブラリは2つのクラス，BasicGraphicsクラスとBasicCanvasクラスからなる．

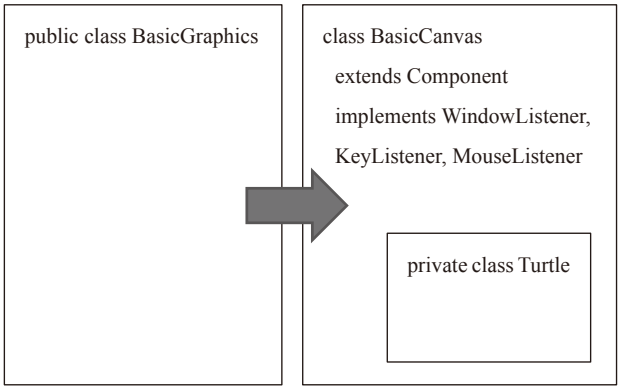

図2　BasicGraphicsライブラリのクラス構成図
Figure 2　Class structure of the BasicGraphics class library.

BasicGraphicsクラスは親クラスを持たないpublicなクラスであり，BasicGraphicsライブラリの機能は，全てBasicGraphicsのpublicなメソッドから提供される．これらpublicなメソッドから，非公開なクラスBasicCanvasが呼ばれ，実際の処理はほぼ全てBasicCanvasに実装される．このため，BasicGraphicsは一種のラッパークラスとしての役割を持つ．

BasicCanvasはJavaの基本GUIコンポーネントクラスであるjava.awt.Componentを親クラスに持つ．BasicCanvasはウィンドウイベント処理のためにWindowListener，キーボードイベント処理のためのKeyListener，マウスイベント処理のためのMouseListenerも実装する．また，タートルを複数同時管理するために，privateなインナークラスとしてTurtleクラスを持つ．

上記HelloWorldプログラムの例では，new BasicGraphics() からデフォルトサイズのウィンドウ（java.awt.Frame）が生成され，Frame上の主コンポーネントとしてBasicCanvasがウィンドウに追加（add）される．再描画イベントなどはBasicCanvasに送られ，

BasicCanvasが自動的に再描画を行う．BasicCanvasは，画面に表示されるテキストを，画面の表示位置と一致した配列でメモリに保持しており，再描画の度にメモリ上の配列から表示テキストを参照し，ウィンドウに表示する．

g.println(" Hello, World!"); では，まずHello, World!の文字列をBasicCanvasが配列として記憶し，すぐウィンドウに再描画処理を行い，再描画のタイミングで配列から文字列を参照してウィンドウにテキストを描画する．通常のJavaプログラムならば文字列を画面イメージに描画し，そのイメージを画像としてダブルバファリングしながら保持するのが普通だが，BasicGraphicsでは画面に描画したテキストを後から消したり，変更したり，表示されている文字を取得する機能を提供するために，テキストを一度配列に保存してから再描画の度に描画しなおしている．これは，BASICの基本的なテキストグラフィックス機能を実装するためで，この機能により，テキストを用いたアニメーションなども簡単に実装できる．

なお，BasicGraphicsクラスの公開メソッドは全てインスタンスメソッドとして実装されているため，基本的にg.println() のようにオブジェクト.メソッド名() のような形で呼び出す必要がある．この点は，インタプリタ型BASIC言語における予約語の呼び出しと違いがある．対して，Processing言語では，println() などは組み込み関数として呼び出せるため，さらにオブジェクト指向を隠す仕様になっている．

３.３ テキストによるアニメーション例

次に，テキストによるアニメーションのコード例を示し，テキスト描画機能の説明を行う．

```
import jp.ac.tuis.basic.*;
public class HelloWorldAnimation{
  public static void main(String[] args){
    BasicGraphics g = new BasicGraphics();
    int y=10;
    for(int x=0; x<27; x++){
      g.locate(x,y);
      g.print(" Hello, World!");
      g.sleep(200);
    }
  }
}
```

図３ プログラムHelloWorldAnimationの実行結果
Figure 3 Result of the HelloWorldAnimation program.

上記プログラムは，左から右に向かって，文字列Hello, World! がアニメーションする．

g.locate(x,y) はそれ以降に呼ばれるprint() の左端の座標を指定する．デフォルトのウィンドウサイズは，横40文字縦20文字となる．ウィンドウサイズおよびフォントサイズは，BasicGraphics() のコンストラクタの引数で変更することができる．

g.print(" Hello, World!"); は前回の位置に右に一つずれて表示される．各文字の下にある前回の文字は消され，新しい文字が上書きされて表示される．文字列の最初に空白が入っているため，空白文字が前回のHの文字を消す効果がある．

g.sleep(200) は200ミリ秒処理を止める．内部的にはThread.sleep() を呼んで，処理を一時的にブロックしている．これにより，単純なforループで簡単にアニメーションが実装できる．なお，ウィンドウの再描画処理は，g.print() が呼ばれるタイミングで自動的に行われる．

上記例と同じアニメーションプログラムをJavaで作成した場合を以下に示す．

```
  import java.awt.*;
  import javax.swing.*;
  public class AnimationPanel extends JPanel implements
Runnable{
    int x=0; //インスタンス変数

    public static void main(String args[]){
      //ウィンドウ生成処理
      JFrame frame = new JFrame();
      frame.setSize(220,240);
      AnimationPanel panel = new AnimationPanel();
```

```
     frame.getContentPane().add(panel);
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CL
OSE);
     frame.validate();
     frame.setVisible(true);
     Thread thread = new Thread(panel); //スレッド生成
     thread.start(); //スレッド開始
   }

   /*描画コールバック paint()メソッド*/
   public void paint(Graphics g){
     super.paint(g);
     g.setColor(Color.BLACK);
     g.drawString("Hello, World!", x, 100);
   }

   /*Runnable インタフェース run() メソッド*/
   public void run(){
     while(true){
       try{
         Thread.sleep(200);
         x=x+10;
         if(x>200){
           return;
         }
         this.repaint(); //再描画依頼
       }catch(InterruptedException e){}
     }
   }
 }
```

上記Javaプログラム例でも見られるように，Java言語でテキストアニメーションを実現するには，オブジェクト指向の理解に加え，GUIコンポーネントの構成によるウィンドウの生成，描画イベントコールバックを用いた描画処理，スレッド生成，例外処理，子スレッドでの繰り返し描画依頼によるアニメーションの仕組みを理解しなければならない．

これに対してBasicGraphicsの場合，Java言語教育の初期段階で教える参照型（例えばStringクラスのnewを用いたメモリ生成など）の説明後なら，学習者は混乱せずにBasicGraphicsを利用することができ，コンソールベースのテキストプログラムとほぼ同じ処理で，テキストグラフィックスを用いたアニメーションが実現できる．

3.4　キーボードからの入力

キーボードからの入力のコード例を示す．

```
import jp.ac.tuis.basic.*;
public class Input{
  public static void main(String[] args){
    BasicGraphics g = new BasicGraphics();
    g.print("Enter name:");
    String name;
    name = g.input();
    g.println();
    g.print("Hello, " + name + "!");
  }
}
```

図4　プログラムInputの実行結果
Figure 4　Result of the Input program.

上記例のように，キーボードからの入力は，g.input()だけで済む．g.input()は，Enterキーが押されるまで，キーボードから文字列を読み込み，String型の文字列として返す．入力中の文字もBasicGraphicsウィンドウ内に表示されるため，コンソール上のテキストベースプログラムとまったく変わらない操作性を提供し，JavaのBufferedReader.readLine()と同じようにプログラムで利用することができる．

4．バブルソートのアニメーション表示

すでに説明したBasicGraphicsの機能を用いて，バブルソートのアニメーションを行う例を示し，従来のコンソールプログラムとの違いとその効果について説明する．

従来のプログラミング教育では，バブルソートの課題プログラムは，バブルソートの挙動を見せるため，図4のように配列の要素をステップ毎に表示することが考えられる．

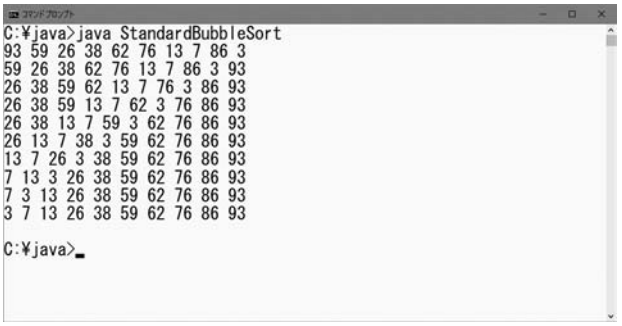

図5　従来のバブルソートプログラム実行結果例
Figure 5　Result of a standard bubble sort program.

従来の出力例では，アルゴリズムの直感的な理解にはつながらなく，またユーザの間違いによるプログラムの論理エラーを見つける補助にはならない．この例に対して，BasicGraphicsのテキストアニメーション機能を用いたバブルソートのコード例を示す．

```
import jp.ac.tuis.basic.*;
public class BubbleSort{
  public static void main(String[] args){
    BasicGraphics g = new BasicGraphics();

    int[] list = {93,59,26,38,62,76,13,7,86,3};

    for(int l=0; l<list.length; l++){
      g.locate(l*3,10);
      g.print(list[l]);
    }

    //バブルソート
    for(int i=0; i<list.length-1; i++){
      for(int j=0; j<list.length-(1+i); j++){

        g.color(g.BLUE);  //比較する要素を青く表示
        g.locate(j*3,10);
        g.print(list[j]);
        g.locate((j+1)*3,10);
        g.print(list[j+1]);
        g.sleep(1000);

        //入れ替え処理
        if(list[j+1]<list[j]){
          int swap=list[j];
          list[j]=list[j+1];
          list[j+1]=swap;

          g.color(g.RED); //入れ替える要素を赤く表示
          g.locate(j*3,10);
          g.print("  ");      //元の要素を消す
          g.locate(j*3,10);
          g.print(list[j]);   //新しい要素を表示
          g.locate((j+1)*3,10);
          g.print("  ");      //元の要素を消す
          g.locate((j+1)*3,10);
          g.print(list[j+1]); //新しい要素を表示
          g.beep();           //ビープ音で合図
          g.sleep(1000);
        }

        g.color(g.WHITE); //比較後元の表示色に戻す
        g.locate(j*3,10);
        g.print(list[j]);
        g.locate((j+1)*3,10);
        g.print(list[j+1]);
      }
    }
  }
}
```

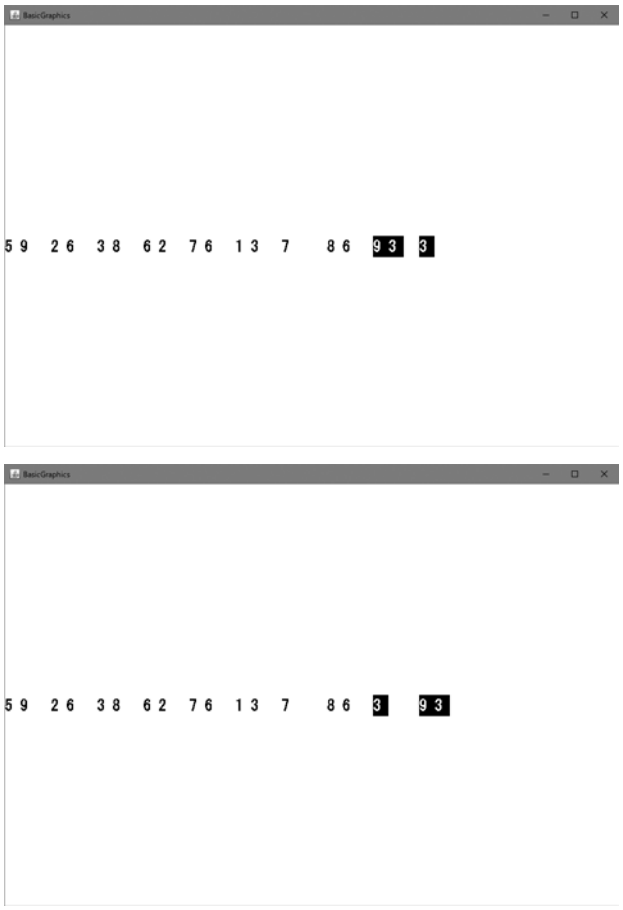

図6　バブルソートアニメーションのデータ入れ替え前後
Figure 6　Bubble sort animation before and after swapping data.

BasicGraphicsのバブルソートのコード例では，値を入れ替えるたびに色を変えてアニメーションを行い，さらにビープ音により，入れ替えアニメーションを際立たせる．アニメーションすることで，２重forループの処理が，具体的に何を行っているのか，視覚的に理解することが可能となり，またユーザのミスによる論理エラーがある場合も，その間違った挙動が視覚的に確認できる．

本例で示した通り，表示色はg.color()で既定の８色から選択でき以降のg.print()のテキスト描画色が変化される．またビープ音はg.beep()で鳴らせる．

５．授業での応用

BasicGraphicsは，東京情報大学総合情報学部情報システム学科の大学１年次前期で初めて受講するプログラミング科目である「ソフトウェア入門」において，応用を促すチャレンジ課題として活用してきた．この科目では，学生はLinux OS上でターミナ

ルからテキストエディタ（Emacs）を用いてプログラミングし，同じくターミナル上からjavacおよびjavaコマンドを用いてコンソールベースのJavaプログラミングを学ぶ．

BasicGraphicsは，ライブラリのダウンロード先と簡単な説明文書[11]と共に提示するだけで，学生が問題なく自習課題として利用できることは，チャレンジ課題の提出状況により確認できている．提出されたプログラムの内容から，まだオブジェクト指向の学習を始めていない学生が，授業範囲であるテキストベースの構造化プログラミングの知識の範疇で，創意工夫を凝らしたグラフィックスを用いたアプリケーションを作成できていることがわかる．また，チャレンジ課題を提出する学生の多くが，完成したプログラムを直接教員に見せにきてコメントを求める点において，学生の学習意欲に良い影響を与えていると考えられる．

また，アニメーション表示がアルゴリズムの理解に影響を与えるか，東京情報大学総合情報学部総合情報学科情報システム学系の３年生12名，４年生12名に対して，次のような簡単な実験を行った．実施環境は，ノートPC上のWindows10からメモ帳（notepad.exe）でソースコードを作成し，コマンドプロンプト（cmd.exe）からjavac.exeおよびjava.exeを実行した．JavaはノートPCにインストール済みのJava SE 11を用いた．ただし，対象の学生は全員，すでにJava言語を用いたオブジェクト指向プログラミング授業を受講済みであり，本来のBasicGraphicsの対象となるJava言語の初学者ではない．

学生を半分に分け，一方は，Java言語によるテキストベースのバブルソートプログラムを提示し，もう一方にはBasicGraphicsを用いたアニメーションを行うバブルソートプログラムを提示した．ソースコードを確認し，プログラムを実行する時間を15分与えた．その後，全員に対して，バブルソートのフローチャートを書くように指示した．その結果，最初のJavaのテキストベースプログラムのグループは4/12（33%）がフローチャートを正しく記述できたのに対して，BasicGraphicsプログラムのグループは，6/12（50%）の学生がフローチャートを正しく記述できた．この結果だけではBasicGraphicsの教育効果が高いと断定することはできないが，課題終了後に両方のグループに改めてBasicGraphicsの使い方を紹介したところ，授業時間外にも関わらず，両方のグループでBasicGraphicsを用いたプログラミングを自主的に開始した学生が多くいたことから，プログラミングに対する興味や学習意欲の向上については効果があると考えられる．

６．おわりに

本論文では，プログラミングの初学者用の学習意欲とアルゴリズム理解度の向上を目的とした，Java言語用簡易グラフィックスライブラリBasicGraphicsの提案と説明を行った．

Logo言語のタートルグラフィックスの登場以来，グラフィックスがプログラミング教育に高い効果があることが示されおり，多くの教育向けプログラミング言語は簡易的なグラフィックス機能を提供している．しかし一方で，これら教育向けプログラミング言語はビジネスや開発現場では用いられないため，大学を含めた多くの専門教育機関では，Java言語などの本格的なオブジェクト指向プログラミング言語で教育を行っている．

Java言語でグラフィックスを表示するためには，オブジェクト指向の知識とともに，Java AWT（Abstract Window Toolkit）のGUIアーキテクチャや，イベントハンドリング，スレッド処理を理解する必要があり，初学者にはハードルが高い．

本論文で提案するJava言語用簡易グラフィックスライブラリBasicGraphicsは，オブジェクト指向やJavaのGUIアーキテクチャをほぼ完全に隠蔽しており，オブジェクト指向を学んでいないJava言語の初学者が，それまで学んできたコンソールベースのテキスト入出力プログラムに，数行加えるだけで，簡単にグラフィックス表示を実現することができる．これにより，学習者がオブジェクト指向やJavaのGUIアーキテクチャおよびスレッド処理を学ぶまでの間，簡易グラフィックスライブラリを用いてプログラムの結果を視覚的に表現することが可能となる．グラフィックスは学習者の興味や学習意欲に効果があることがすでに多くの研究結果で示されており，さらにグラフィックスを用いることでアルゴリズムの理解度にも効果があることが期待される．

BasicGraphicsは非常に単純な記述で動作する簡易的なグラフィックス機能のため，その後のJava標準のグラフィックス機能の学習時において，学習者に

混乱を招く学び直しなどの必要がなく，スムーズに本来のオブジェクト指向グラフィックスプログラミングの学習に進むことができる．

BasicGraphicsには，本論文で説明した機能以外にも，線画などのペイント機能，タートルグラフィックス機能，画像ファイル表示，ソケット通信，サウンドファイル再生，MML（Music Macro Language）を用いた演奏，キー入力読み取り，マウス入力読み取り，アプレット表示などが行え，従来ならオブジェクト指向が必須となるこれら機能を，１行で実現できる．例えば，send(int) でintデータをソケット通信で送信することができ，receiveint() でそのデータを受け取ることができる．これにより，多くの学生を混乱させるプログラムの前処理を省き，重要なアルゴリズムの学習に専念させることができる．例えば，通信アルゴリズムであれば，どのような順番でどのようなデータを送るかという，アプリケーション層の通信プロトコルの仕組みを簡単に実践することができる．

BasicGraphicsの詳しいAPIについては，APIレファレンス[12]にて公開されている．

BasicGraphicsの今後の開発予定としては，ゲームコントローラーなどのマウスやキーボード以外のユーザインタフェースへの対応と，簡易的な３Ｄ表示の機能追加を予定している．

参考文献

［1］ Papert, S.. Mindstorms: Children, Computers and Powerful Ideas. Basic Books, New York. 1980

［2］ Glinert, Ephraim P.（ed）. Visual Programming Environments: Paradigms and Systems. IEEE Computer Society Press. 1990

［3］ Tisue, S. and Wilensky, U.. Netlogo: A simple environment for modeling complexity. International Conference on Complex Systems. 2004

［4］ Caspersen, M.E. and Christensen, H.B.. Here, there and everywhere- on the recurring use of turtle graphics in CS 1. Proceedings of the Australasian Conference on Computing Education. 2000, pp. 34-40

［5］ Abelson, Harold and diSessa, Andrea. Turtle Geometry. The MIT Press. 1981

［6］ Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A. and Miller, P.. Mini-languages: a way to learn programming principles. Education and Information Technologies, Springer. 1997, vol. 2, no. 1, pp. 65-83

［7］ Du, Jie, Wimmer, Hayden and Rada, Roy. “Hour of Code”: Can It Change Students’ Attitudes Toward Programming?. Journal of Information Technology Education: Innovations in Practice. 2016, vol. 15, pp. 53-73.

［8］ Roberts, Eric, Picard, Antoine and Fredricsson, Maria. Designing a Java graphics library for CS 1. Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education, 1998, pp. 213-218

［9］ Schaub, Stephen. Teaching Java with Graphics in CS1. ACM, SIGCSE Bulletin. 2000, vol. 32, no. 2, pp. 71-73

［10］ Reas, Casey and Fry, Ben. Processing: programming for the media arts. Ai & Society. 2006, vol. 20, no. 4, pp. 526-538.

［11］“プログラミング教育向けJavaライブラリBasic Graphics”．http://www.edu.tuis.ac.jp/~mackin/basic/,（参照 2021-10-01）.

［12］“BasicGraphics APIレファレンス”．http://www.edu.tuis.ac.jp/~mackin/basic/javadoc/,（参照 2021-10-01）.