Vol.2020-CE-154 No.1
2020/3/14

タートルグラフィックスにおけるプログラム理解支援手法の提案

柿島 遥[1,a)]　西田 誠幸[1,b)]

概要：プログラミング教育において，タートルグラフィックスとそれを用いたフラクタル図形描画するプログラムが用いられることがある．この種のプログラムは再帰呼び出しを含むものであり，プログラムを理解するという観点では比較的難しい．そこで本稿では，タートルグラフィックスのうちで再帰呼び出しを含むプログラムの理解を支援する一手法として，コードと図の断片それぞれを対応づけて強調表示する手法を提案する．また，本手法を評価するために行なった実験とその結果を報告する．

キーワード：タートルグラフィックス，再帰呼び出し，プログラム理解

A Method to Support Program Comprehension in Turtle Graphics

Abstract: Fractal drawing programs in turtle graphics commonly involve recursive calls, which often make it difficult for students to understand programs. To support student's comprehension of fractal drawing programs, this paper proposes a way to show relationship of code snippets and their output of figure fragment. This paper also report development of an educational tool using our proposed method, and experimental evaluation.

Keywords: Turtle graphics, recursion, program comprehension.

1. はじめに

ベクターグラフィックスのひとつであるタートルグラフィックスは，画面上の仮想的な亀を模したペンに指示を出し，その軌跡をプログラムで描画するグラフィックス出力方式である．また，タートルグラフィックスはプログラムと実行結果の図を見比べることができるため，学習用のプログラミング言語として用いられることがある．

タートルグラフィックスを用いたプログラミング教材の1つにフラクタル図形の描画がある．タートルグラフィックスにおけるフラクタル図形描画プログラムでは，図形の自己相似な断片の描画を，描画関数の再帰呼び出しによって記述する．再帰呼び出しを含むプログラムを読むことが苦手な学習者にとっては，プログラムの挙動や，プログラムの各部分が図形のどの部分に対応するのか，どうして再帰呼び出しによって記述するのかを理解するのは難しい．

そこで，本稿ではタートルグラフィックスにおいて，再帰呼び出しを含むフラクタル図形描画プログラムの理解支援の一手法を提案する．この手法は，プログラムコードの断片と，そのコードの実行によって描画される図形とを対応づけて学習者に提示するものである．また，提案手法をもとにした教育ツールの概要と，このツールを使用した実験的評価，実験結果を報告する．

以下2節では，タートルグラフィックスにおける再帰呼び出しを含むプログラムの理解支援手法を提案する．3節では，提案手法の評価実験とその結果を報告する．4節では，提案手法に関する関連研究を述べる．

2. タートルグラフィックスにおけるプログラム理解支援手法

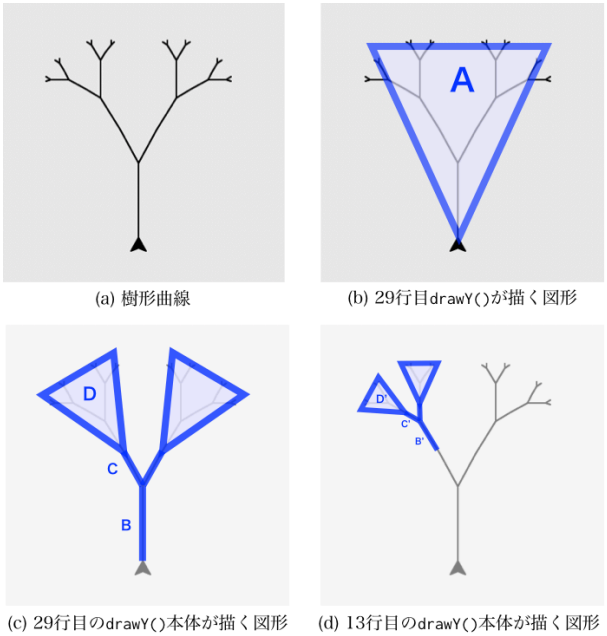
2.1 前提とするプログラム作成および実行環境

本稿では，プログラミング言語 Python とこれに付属す

[1] 拓殖大学工学部情報工学科，193–0985 八王子市館町 815-1
Department of Computer Science, Faculty of Engineering, Takushoku University, 815-1 Tatemachi Hachioji, Tokyo 193–0985, Japan
[a)] r68415@st.takushoku-u.ac.jp
[b)] snishita@cs.takushoku-u.ac.jp

```
1  # Y を描く
2  def drawY(length):
3      #再帰の終点
4      if length < 5:
5          return
6
7      # 幹を描く
8      fd(length)          # B
9
10     #左の枝を描く
11     lt(30)
12     fd(length / 2)      # C
13     drawY(length / 2)    # D
14     bk(length / 2)
15     rt(30)
16
17     #右の枝を描く
18     rt(30)
19     fd(length / 2)
20     drawY(length / 2)
21     bk(length / 2)
22     lt(30)
23
24     bk(length) #幹の根元に戻る
25 #初期配置
26 setheading(90)
27
28 #樹木曲線を描く
29 drawY(50)              # A
```

図 1 樹木曲線のプログラム



(a) 樹形曲線　(b) 29行目drawY()が描く図形
(c) 29行目のdrawY()本体が描く図形　(d) 13行目のdrawY()本体が描く図形

図 2 樹木曲線プログラムと図形の対応

る Turtle ライブラリを使ったプログラム作成および実行環境を前提とする．プログラムの一例を図 1 に示す．プログラム中の `fd`, `bk`, `lt`, `rt` は，それぞれ亀を前進，後退，左旋回，右旋回させる関数である．また，ペンの上げ下げなどタートルグラフィックスの標準的な関数を持つ．さらに，これら以外の代入文，分岐，繰り返し，ユーザ定義関数などの Python の標準的な構文が使用可能である．

2.2 フラクタル図形とタートルグラフィックス

フラクタル図形の 1 つ，樹木曲線を図 2(a) に示す．フラクタル図形の性質として，図形全体とその断片が相似であることが知られている．樹木曲線では 2 つに枝分かれした先が図形全体と相似である．樹木曲線全体を図 2(b) の三角形 A で抽象化して表すと，同図 (c) の三角形 D にあたる図形の断片は，三角形 A の図形全体と相似である．また，この三角形 D にあたる図形の断片とこれに含まれる図形（同図 (d) の三角形 D'）もまた相似関係にある．このようにフラクタル図形は図形（の断片）が自己相似な部分を持つという特徴がある．

次にタートルグラフィックスによるフラクタル図形を描画するプログラムの一例として樹木曲線描画プログラムを図 1 に示す．このプログラムはテキスト Turtle Cafe2 [8] の 14.3 節に掲載された，樹木曲線を描画するプログラムを本稿の構文にしたがって書き換えたものである．関数 `drawY()` は，樹木曲線における自己相似な断片を描画するため，13 行目と 20 行目において `drawY()` を再帰呼び出しする．この関数の引数は図形の大きさを意味する．再帰呼び出しの実引数として指定される `length/2` は関数の引数 `length` よりも小さい．再帰呼び出しが深くなるにつれて，引数にはより小さな値が指定され，いずれ再帰の終点（3 行目から 5 行目）の条件 `length < 5` を満たすこととなる．一般に，フラクタル図形を描画するプログラムは様々なものがあるが，本稿では図 1 のような再帰呼び出しを含むプログラムをプログラム理解支援の題材とする．

図 1 のプログラムは，フラクタル図形の自己相似の断片の描画を再帰呼び出しによって記述する．したがって，このプログラムの一文やコードブロックと，その実行によって描画する図形の断片との間の対応を取ることができる．樹木曲線描画プログラムの 29 行目の関数 `drawY()` の呼び出し文が描く図は，樹木曲線全体，すなわち図 2(b) の三角形 A である．また，29 行目の関数呼び出し中の 8 行目 `fd()` 文は図 2(c) の線分 B を描く．

2.3 再帰呼び出しプログラム理解の難しさと支援

再帰呼び出しを含むプログラムを理解する際に学習者が混乱する要因の一つに，実行トレースがある．図 1 のプログラムの実行にしたがって，亀は幹から，枝，葉に向かって描画し，逆に葉から幹に戻る．この様子を実行トレースによって推測することは可能ではある．しかし，トレースの過程で呼び出し元と呼び出し先のコンテキストが頻繁に切り替わることで，学習者が正しくトレースできないことがある．また，実行トレースの結果から，再帰呼び出しの

働き，すなわちフラクタル図形における自己相似を実現するための再帰呼び出しの働きを理解するのは難しい．

一方，再帰呼び出しによって得られる結果を先に仮定すると，プログラム全体の処理の概要を理解しやすくなることがある．例えば，樹木曲線描画プログラムにおいて，「`drawY()` の呼び出しによって樹木曲線が描かれる」ことを仮定すると，再帰呼び出し元をトレースすることなく，図 2(c) のように描画図形の概要を推測することが可能である．しかし，プログラムを理解するために，仮定をおいて考えることは学習者にとっては簡単ではない．

そこで，本稿では，図 2(c) のようなコードと図の対応を学習者に提示する方法をとる．この方法によって，上記の仮定が苦手な学習者や，再帰呼び出しを含むプログラムの実行トレースをよく間違う学習者に対して，図 2(c) のような対応関係の確認を支援すること，さらにフラクタル図形の自己相似な断片を描画するという再帰呼び出しの働きの理解を支援することを狙う．

2.4 コードと図の対応づけとその提示方法

コードと図の対応づけのため，次のルールを定める．

R0 プログラムの中の単文とコードブロックを対応づけの対象とする

R1 `fd` 文と `bk` 文は，それぞれの文の実行時に亀が描く線分に対応づける

R2 コードブロックは，このコードブロックの実行によって描かれる図形に対応づける

R3 重なる線分は区別して，それぞれのコードと対応づける

ルール R0 の補足説明として，コードブロックは，条件文の then/else 節，繰り返し文の本体，ユーザ定義関数の本体を指す．また，ルール R1 に関連して，亀のペンが上がっているときには，亀が移動するだけで線分が描かれないので，このときに `fd` 文に対応する図形はない．さらに，ルール R3 において，条件式が真のときには else 節が実行されず then 節だけが実行されるため，このときに else 節に対応する図形はなく then 節に対応する図形だけがある．

本稿では，コードと図の対応関係の提示のための教育ツールを作成する．このツールの画面上で対応関係を提示するために，次の方法をとる．

(1) コードの背景色と図の線色を同色に色付けして強調表示することによって対応関係を提示する
(2) 全ての対応関係を一度に提示するのではなく，注目するコードの断片とこれに対応する図の断片に絞って強調表示する
(3) 対応関係の提示のときにコンテキストを考慮する
(4) 最初はコード全体と図全体の対応関係を強調表示する．ユーザ操作によって，注目するコードと図を絞り込んでいくことができるようにする

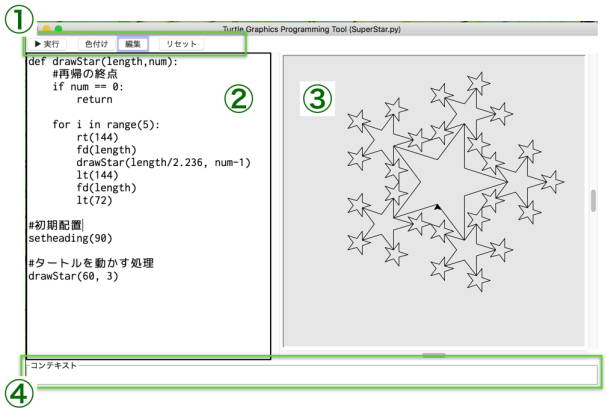


図 3 教育ツールの画面構成

(3) の補足説明をする．注目するコードの断片が繰り返し文の本体にある場合，対応する図の複数の断片に対応することがある．このとき，整数 i に対して「繰り返し処理の i 回目」に限定すれば，注目するコードと対応する図の断片を一意に決定できる．同様のことは，関数定義文の本体に対しても起こりうる．そこで，繰り返し本体の実行回数と関数呼び出しを「コンテキスト」と呼び，与えられたコンテキストのもとで注目するコードの断片に対応する図の断片を 1 つに特定して提示することとする．

2.5 教育ツールの概要

タートルグラフィックスにおけるコードと図の対応を提示する教育ツールを開発した．本ツールは，コードと図の対応を背景色と線色の色付けによって提示する機能を持つ．このツールの画面構成を図 3 に示す．なお，この図は，フラクタル図形「スーパースター」を描画するプログラムとその実行結果の図形を本ツールで表示しているところである．

ツール画面は 4 つのコンポーネントで構成される．

① 制御パネル
② プログラム編集エリア
③ 図形描画フィールド
④ コンテキストパネル

本ツールは，提案手法に従った色付けモードと，通常のプログラム編集モードの 2 つのモードをもっていて，ユーザは制御パネルの「色付け」と「編集」のボタンによってモード切り替えができる．プログラム編集モードでは，ユーザはプログラム編集エリアのプログラムを自由に編集することができる．

色付けモードは，プログラムを実行して，図を描画し終わった時点で有効となる．

色付けモードへ切り替えると，メインブロックとそれに対応する図形全体を強調表示する（図 4）．プログラム編集エリアの `fd` 文，`bk` 文，繰り返し文，関数呼び出し文のいずれかにマウスカーソルが重なるとその文の背景色と対応す

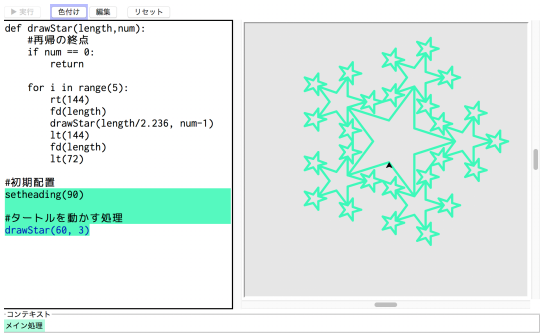


図 4　色付けモードの初期状態

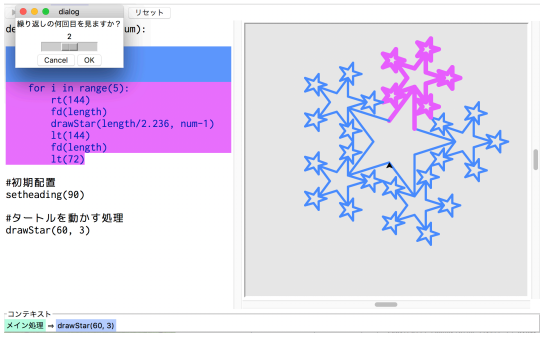


図 6　繰り返し文を選択した状態

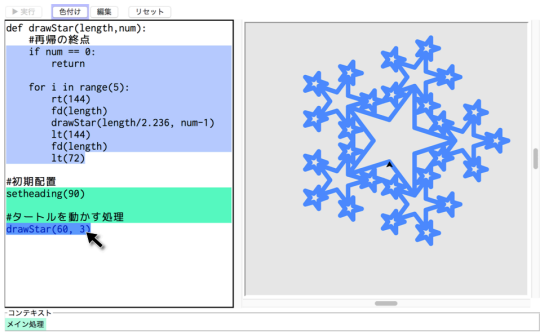


図 5　関数呼び出し文にマウスカーソルを重ねた状態

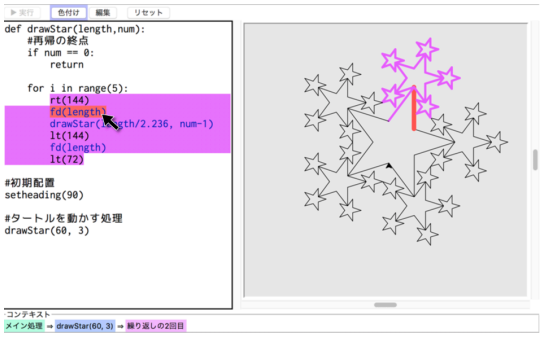


図 7　fd 文にマウスを重ねた状態

る図形の断片の線色を同色で塗り，さらに線幅を太くして強調表示する．図 5 では，メインブロックの `drawStar()` の呼び出し文にマウスカーソルを重ねたときの様子を示している．この呼び出し文で図形全体を描画するため，マウスカーソルを重ねると，図全体が強調表示される．

プログラム編集エリアにおいて，マウスカーソルが重なったのが繰り返し文，または関数呼び出し文の場合には，マウスクリックによって注目するコード領域を狭めることができる．このとき，コンテキストパネルには，繰り返し回数，あるいは関数呼び出し文を追加表示する．図 6 は，メインブロックの `drawStar()` 呼び出し文をクリックし，次に関数定義本体の繰り返し文をクリックした時の様子を示している．このとき，繰り返しの何回目をコンテキストとするかユーザに選択してもらうためのダイアログが表示される．ユーザがスライダを操作するのにつれて，繰り返し回数の指定が変わり，強調表示する図の断片が変化する．

繰り返し 2 回目を選択し，繰り返し文本体の `fd` 文にマウスカーソルを重ねたときの様子を図 7 に示す．この時点でのコンテキストは，「メイン処理」，「`drawStar(60, 3)`」，「繰り返し 2 回目」の 3 つである．コンテキストパネルでは，これらアイコンのいずれかをマウスクリックすることによって，注目するコード範囲を元に戻すことができる．

本ツールの実装について説明する．本ツールでは，プログラム実行の直前に，対応関係を記録するための追加コードを埋め込んでいる．プログラムの実行とともに，コードと図の対応関係の情報を取得し，実行終了後に，この情報を使ってコードと図の強調表示を行なっている．

3. 提案手法の評価実験

提案手法の実験的な評価を行なったので報告する．

3.1 実験方法

実験は授業時間外に，実験協力者を募る形式で実施した．実験協力者は拓殖大学の情報工学科 2 年生から 4 年生，合わせて 20 人である．彼らは Java による手続き型，オブジェクト指向プログラミングの経験を持つが，そのほとんどは Python とタートルグラフィックスによるプログラミングの経験がなかった．この 20 人を 10 人ずつからなるグループ A と B の 2 つに無作為に分けた．グループ A には提案手法を実装したツールを配付し，グループ B にはプログラム実行機能だけを持つツールを配付した．

実験の手順は以下の通りである．全体で 65 分の実験である．60 分前後の実験時間とした理由には，時間が比較的短い方が実験協力者の応募者数が見込めること，時間が短すぎると学習の時間が十分取れない恐れがあったことがある．

(1) アンケート用紙を除く，資料全てを配付（5 分）
(2) ツールの使用方法を学習（5 分）
(3) Python とタートルグラフィックスの学習（10 分）
(4) フラクタル図形の学習（30 分）
(5) 小テスト，アンケートに回答（15 分）

評価実験にあたり，資料「ツール使用方法」，資料「Python とタートルグラフィックスおよびフラクタル図形の描画方法」，アンケート用紙を用意した．このうち，資料「ツール

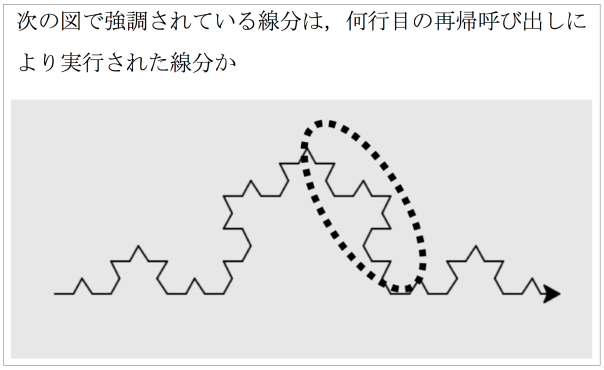


図 8 小テストのうちの 1 問

使用方法」では，ツールの導入と起動方法，実験全体の流れを説明している．なお，グループ A の実験協力者にのみ，色付け機能の使用方法の説明を資料に追加した．「Python とタートルグラフィックスおよびフラクタル図形の描画方法」の資料は，Turtle Cafe 2[8] の関連箇所を抜粋し，プログラムを Java から Python に書き換えて作成した．また，この教材の中で提示されるプログラムと，練習問題の解答例のプログラムファイルをあわせて実験協力者に配付した．

実験協力者には，配付したプログラムファイルをツールで開き，実行して学習してもらうこと，必要に応じて自身でプログラムに変更を加えて実行結果の変化を見比べて学習してもらうことを口頭でお願いした．なお，Python，タートルグラフィックスについての口頭での説明は行わなかった．また，インターネットを使用して学習することは禁止し，配付資料のみで実験を進めてもらうようお願いした．

(5) の小テストとアンケート調査は提案手法を用いたグループ A の協力者にのみ出題した．とくに小テストは，再帰呼び出しの理解度を測るためのもので，学習時には使わなかったコッホ曲線のプログラム（図 9 左のプログラム）を題材とした．小テストは全 3 問で，各 1 点の 3 点満点とした．小テストの 1 つを図 8 に示す．一方，アンケートはツールの使いやすさ，プログラム理解のしやすさ，ツールをもっと使ってみたいかを 5 段階評価で問うものである．また，アンケートには自由記入欄を設けた．なお，小テストとアンケートの回答中はツールの使用を禁止した．

3.2 実験結果

評価実験における小テストの平均点は，グループ A が 1.4 点，グループ B が 1.3 点で，2 つのグループに大きな差は見られず，小テストの結果からは提案手法の効果が認められなかった．

次に，アンケートのうち 5 段階評価の回答結果を表 1 に示す．本ツール使用者の評価がいずれも高かった．特に，小テストの平均点は振るわなかったものの，実験協力者自身の自己評価ではプログラム理解の支援を受けたという認識をもっていることが読み取れる．

表 1 ツールについてのアンケート

	5 段階評価の平均
ツールは使いやすかった	4.2
プログラムを理解しやすくなった	4.4
ツールをもっと使ってみたい	4.2

一方，アンケートの自由記入欄では，

- 学習と小テストの時間をもっと増やして欲しかった
- サンプルプログラムがもっと欲しかった

などの実験方法に対する意見があった．

3.3 実験結果の考察

小テストの結果として 2 つのグループで小テストの平均点に大きな差がなかったこと，平均点が 3 点の半分に満たないことから，実験的評価の結果として提案手法の効果が認められなかったと言える．

アンケートの自由記入欄では，実験協力者からは実験時間が短いという意見が多数よせられた．実験の過程で学生は Python の構文，タートルグラフィックスによるプログラミング，フラクタル図形描画プログラムと多くの学習項目を学ぶ必要がある．実験中の学生の様子を見たところ，時間に追われながら余裕なく取り組む姿がたびたび見られた．実験における各学習項目の理解の度合いは小テストの結果に影響する可能性があるため，この実験によって提案手法の評価を正確に行えなかった可能性がある．そこで，Python 言語を使用するのではなく，学生がすでに学んでいるプログラミング言語を使用して評価実験を改めて実施する方法が考えられる．評価実験における実験協力者の学習項目を減らすことができるため，提案手法のより正確な評価ができる可能性がある．

一方，5 段階評価のアンケートでは，ツールの使いやすさよりも，プログラムの理解のしやすさの方が平均点が高く，実験協力者の中には後者に意識的に高い点をつけた人がいた．彼らに対しては，提案手法の機能を持つ教育ツールがプログラムの理解に役立つものであるという認識を持たせることはできていたと考えられる．

3.4 関連研究

タートルグラフィックスにおけるプログラム理解支援に関する研究として，長が提案した「親亀・子亀方式」に基づくフラクタル図形の描画方法とこれを用いた再帰プログラミングの授業の実践 [1] がある．通常のタートルグラフィックスでは 1 つの亀が図形すべてを描画するのに対して，親亀・子亀方式では再帰呼び出しのときに，それまで使っていた亀を複製した子亀を作ることによって，呼び出し元と呼び出し先での処理を区別して学習者に提示する．この方式では，各亀が図形を描画する過程を見せるため

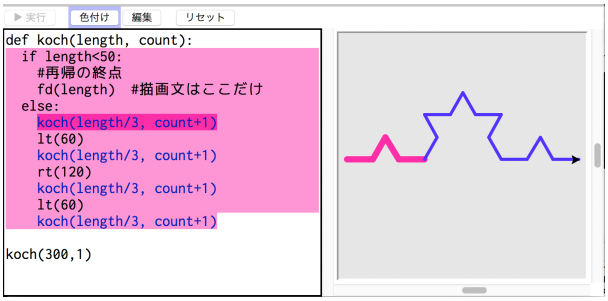


図 9 コッホ曲線描画プログラムと提案手法適用結果

に，プログラムをステップ実行するのに対して，本稿の提案手法はプログラムを実行し終わった後に，プログラムの各部分が描画した図形を提示するという点に違いがある．また，図 9 のコッホ曲線描画プログラムのように再帰の終点でのみ図形を描画するプログラムに対して，親亀・子亀方式ではもっとも小さな子亀のみが図形を描くこととなってしまうため，例外的にコッホ曲線の頂点を打つ描画方法を採用している．これに対して，本稿の提案手法はこのような例外が生じることはなく，他のフラクタル図形描画プログラムと同様に適用可能である．

本稿の提案手法は，プログラムの実行状態を可視化する技術の 1 つとも捉えられる．長の提案手法のようにプログラムをステップ実行しつつプログラムの実行状態を可視化するツールには，Python Tutor [2] など多数ある．これらのツールでは，プログラムリストとともに，実行過程で変数が保持する値や，コールスタック，ヒープの状態などを提示する．この際，プログラムリストの一部を強調表示することによって，実行中の文を示している．しかし，先行する研究のうちで，本稿の提案手法のようにコードと出力結果の対応を示す方法は見つけることができなかった．

ステップ実行とは別に，プログラムの実行ログをとっておいて，プログラム実行終了後に実行状態を可視化する手法がある．Terada が提案したツール ETV [4] は，プログラムの実行トレースを可視化するものであり，プログラムの実行経路情報を記録し，プログラムを実行させたあとにその情報を視覚的に表示する．この点では，ETV は本稿の手法と類似する．しかし，ETV では，関数呼び出しが起こったときに，ソースコードウィンドウの複製を作って，関数の本体に制御が移ったことを提示する方式をとっている．本稿の提案手法では，1 つのコード表示ペーンを使う方式をとっていて，その代わりに学習者がマウスドラッグとクリックによって実行コンテキストを切り替えてもらう必要があり，この点で ETV と提案手法と異なる．

コードと出力結果を提示するという観点では，CUSEC における Victor の招待講演の動画 [6] がある．この動画の一部では，Web ページ上に図形描画する JavaScript が提示され，その一文にマウスカーソルを重ねると，その文が出力する図形を強調表示するというデモを行っており，本稿の提案手法と類似している．ただし，単文を対象としたデモを行っているのみで，提案手法のようにコードブロックに対しての強調表示のデモを行っていない．また，コンテキストの切り替え機能の有無もわからない．

また，Victor は Web ページ [7] において，プログラミング教育の一手法として，プログラムの出力結果とコードを対比して提示する方式を提案している．この手法は，プログラムの関数呼び出し文にマウスカーソルを合わせると，その文が出力する図形を強調表示する．また，関数呼び出しの実引数にマウスカーソルを合わせると実引数の意味（円の半径，図形の位置など）を伝える情報を出力結果に重ねて描画する．この手法との違いとして，本稿の提案手法は，可視化対象のプログラムを書き換えたり，一から書くことが可能なため，能動的な学習を支援できる点と，コンテキストを切り替えながらコードと図の対応を可視化する点がある．

最後に，本稿の先行研究として，Satoh ら [5] の可視化手法を紹介する．この手法は，タートルグラフィックスにおいてコードと図の対応を提示するという点では本稿の提案手法と同一である．ただし，Satoh らの手法は，ユーザ定義関数を対象から外している点と，コンテキストの切り替え機能を持たず，繰り返し本体のすべての回の実行結果を一度にまとめて色分けしつつ提示する点の 2 つで本稿の提案手法と異なる．

4. おわりに

本稿では，タートルグラフィックスにおいて再帰呼び出しを含むプログラムのうち，フラクタル図形の描画を行うプログラムの理解を支援する一手法を提案した．提案手法は，プログラムのコードの断片と，その断片の実行によって描かれる図形の断片との対応を提示するものである．提案手法の機能を搭載した教育ツールを作って 60 分程度の評価実験を行なったところ，提案手法によるプログラム理解支援の効果は認められなかった．今後の課題として，評価実験の再設計がある．特に，実験協力者が学習済みの言語を使った評価実験を行うことで，提案手法のより正確な評価が見込まれる．

参考文献

[1] 長　慎也：文科系大学生を対象とした再帰プログラミングの学習，研究報告コンピュータと教育，Vol. 2009-CE-99,2, No. 3, pp. 1–8 (2009).
[2] Guo, P. J.: Python Tutor — Visualize Python, Java, C, C++, JavaScript, TypeScript, and Ruby code execution, (online), available from ⟨http://pythontutor.com/⟩ (accessed 2019-06-18).
[3] Guo, P. J.: Online Python Tutor: Embeddable web-based program visualization for CS education, *In Proceedings of the 44th SIGCSE Technical Symposium on Computer Science Education* (2013).

[4] Terada, M.: ETV: a program trace player for students., *In Proceedings of ITiCSE 2005*, pp. 118–122 (2005).
[5] Satoh, M. and Nishita, S.: Correlating Program Code to Output for Supporting Program Understanding, *In Proceedings of IMECS 2019*, pp. 180–183 (2019).
[6] Victor, B.: Inventing on Principle, Invited Talk at CUSEC (online), available from ⟨https://vimeo.com/36579366⟩ (accessed 2019-06-18).
[7] Victor, B.: Learnable programming, (online), available from ⟨http://worrydream.com/LearnableProgramming⟩ (accessed 2019-06-18).
[8] Yoshiaki, M.: Turtle Cafe 2, (online), available from ⟨https://macc704.github.io/TurtleCafe/turtlecafe2/html4/index.html⟩ (accessed 2019-12-09).