DEIM2020 I1-4(day1 d13)

プログラミング学習における演習課題の過程評価支援システムの開発

三嶋　哲也† 　前川　絵吏† 　谷口　哲朗† 　清光　英成† 　西田　健志†

† 神戸大学大学院 国際文化学研究科　〒 657-8501 兵庫県神戸市灘区六甲台町 1 − 1
E-mail: †{tetsuya.mishima,eri.maekawa,tetsuro.taniguchi}@mulabo.org, ††kiyomitu@kobe-u.ac.jp,
†††tnishida@people.kobe-u.ac.jp

あらまし　プログラミング演習は受講者がプログラムを実際に書くことができるという点で優れている．しかし受講者の人数が多ければ, 講師による評価において個人の課題作成の過程を評価することは困難である．そこで本稿では，JavaScript のビジュアルコーディングライブラリである p5.js を用いたプログラミング演習において，学生が演習中に編集したプログラムを一定間隔でスクレイピングすることで，課題評価時に編集履歴と編集回数，実際に描画されたキャンバスを確認することのできるシステムを開発した．本システムによって課題の完成度と過程を同時に評価することが可能となる．

キーワード　プログラミング教育，プログラミング演習，評価支援システム

1 はじめに

近年の高度情報化社会において，プログラミング教育の重要性はさらなる高まりを見せている．その一例として，文部科学省は 2020 年度より学習指導要領にプログラミング教育を必修化することを決定した [1]. 大学をはじめとする教育機関でも，プログラミングをテーマとした授業が増加しており，実際にコードを書くような演習形式のものも多い．

プログラミング演習は講義と比較して受講者が実際にプログラムを書くことができるという点で優れている．一方で，受講者の人数が多ければ講師による課題の評価において，成果物がどのような過程で制作されたのかを個々人について把握することが困難であるという性質もある．しかし演習課題の評価において，講師が受講者の課題作成の過程を知ることは適切なフィードバックを行うためや今後の演習の方針（課題の難易度や講義のスピードなど）を決定するために重要な要素の 1 つであると考えられる．そのためプログラミング演習では講師がより効率的かつ正確に，受講者の課題がどのような過程で作成されたのかを把握することが求められる．

そこで本稿ではプログラミング学習，特に演習において，講師が評価を行う際の課題の過程の把握を支援するシステムを開発する．

本論文は以下のような構成になっている．第 2 章では，関連研究を紹介し，従来手法と本研究との関連について述べる．第 3 章では，本システムに関して開発環境や使用したライブラリ，システムの構成や画面の設計について述べる．第 4 章では，本稿のまとめと今後の課題や展望について述べる．

2 関連研究

プログラミング学習における演習課題の過程評価の支援を目的とした研究は，演習中にリアルタイムに過程を把握するものと演習後に次回演習に向けて過程を把握するものの 2 種類が存在する．

リアルタイムに過程を把握するものとして，藤原ら [2] は講師がコーディング過程の各種履歴を組み合わせたパターンを定義しておくことで，過度に進捗が遅れている等の事前に決められたパターンに合致する受講生を検出できるシステムを作成している．また井垣ら [3] はプログラミング演習中に停滞や遅延が発生している受講生の状況把握を目的に，総 LOC(Lines Of Code)，課題ごとのコーディング時間，単位時間あたりのエディタ操作数，課題ごとのエラー継続時間の 4 種類のメトリクスをリアルタイムに計測し，ランキング等の可視化処理を行い，講師に提示するシステムを作成している．彦坂ら [4] はコンパイル時等のエラーの継続時間とその種類を簡易的に識別したものを講師に掲示することで，的確かつ早期に受講者を支援することができるようなシステムを作成している．これらはリアルタイムに受講者のコーディング過程を把握し，アドバイスなどの支援に役立てることを目的としている．そのため座席表と連携させる，なるべく早期に受講者のつまずきを検知するなどの機能が搭載されたものが多い．

演習後に次回演習に向けて過程を把握するものとして，諏訪ら [5] は，コンパイルの履歴を取得することでコードの差分やエラーメッセージなどの情報をまとめて，講師が参照できるようなシステムを開発し，全体的な受講者の学習状況の把握を試みている．田口ら [6] は，協調フィルタリングを用いることで最も学習効果が高まると思われる演習課題を個別に選出することで，学習意欲を失わせることなく学習を継続させることに成功している．また加藤ら [7] [8] は，個別学生の学習状況を把握するという課題を，作業開始時刻の外れ値を分析し，作業進度の遅れ検出により解決している．蜂巣ら [9] は，ソースコードを演算子，特徴式，特徴文，特徴文の構造の 4 つの観点から分析

を行い，学習者の誤りなどのコーディング状況を把握可能にしている．藤原ら [10] は，受講者のソースコードをトークン化し，模範解答との距離を求めることで行き詰まり箇所や原因を特定し，個人へのフィードバックを最適化することを試みている．

これらのシステムはどれも個別またはクラス全体の学習状況を把握するために編集履歴の内容やエラーログ等を取得し，様々な解析を行った上で生徒や講師に次回演習に向けたアドバイスや情報を提供するための機能を搭載している．

これまでの関連研究では様々なアプローチでプログラミング課題の過程の評価・把握を可能にしているが，本稿ではより簡易的なプログラミング演習課題の過程評価支援システムの開発を目指す．本システムを使用する対象として想定されている演習では，言語として JavaScript を用い，特に p5.js というビジュアルコーディングライブラリで図形やアニメーションを描画する課題が課される．講師はあらかじめファイルの雛形を課題ごとに用意し，HTML ファイルに JavaScript へのパスを読み込ませておくことで，受講者は JavaScript のソースコードを書くことに集中できる環境が用意されていた．これらの HTML ファイル等を受講者の学籍番号に紐づいた学内ネットワークのパブリックなファイルシステム上にダウンロードさせることで，著者らは特別なエディタやシステムの開発を必要とせずに，簡単なスクレイピングを行うスクリプトのみによって受講者らのソースコードの編集履歴を取得することができる．より詳細な編集履歴の取得方法や表示方法などについては次節のシステム概要において説明を述べる．

3　演習課題の過程評価支援システム

3.1　システム概要

Section名
section1-1
section2-1
section2-2
section2-3
section2-q1
section2-q2
section2-q3
section2-q4
section3-1
section3-2
section3-3
section3-q1
section3-q2
section4-1
section4-2
section4-3
section4-4

図 1　課題選択画面

講師がコーディングの過程を把握するために用いる要素として，本システムではソースコードの編集履歴，編集が行われた回数，編集が行われたタイムスタンプ，そして描画の結果が出力されたキャンバスを採用している．講師は図 1 の課題選択画面から確認したい課題の番号を選ぶ．課題を選択すると，受講者ごとに図 2 のような課題の過程を確認することのできる画面が表示される．

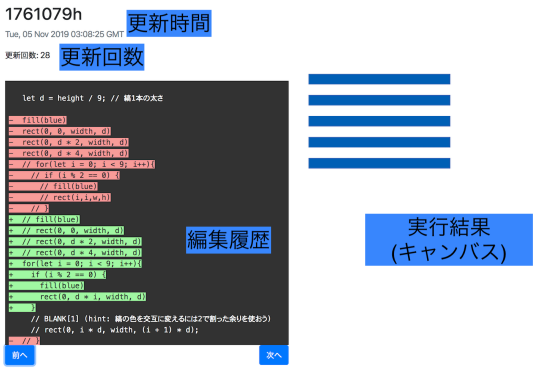

図 2　課題過程確認画面-1

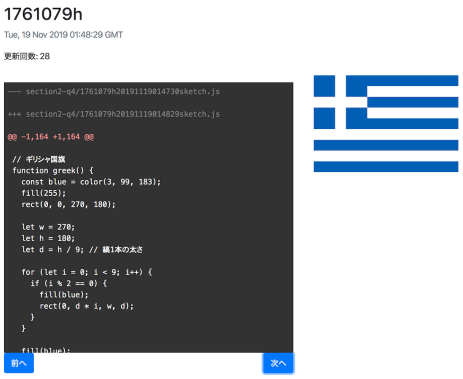

図 3　課題過程確認画面-2

3.2　課題ソースコードのスクレイピング

前章でも述べたように，本稿で対象にしたプログラミング演習では学籍番号に基づいた学内ネットワークのパブリックなファイルシステム上にあるファイルを受講者が編集していく形式であった．そのため，受講者ごとに一意な URL に編集するファイルが置かれているので，そのファイルをスクレイピングした．演習が行われている 90 分の間に一定間隔でファイルのの内容を確認し，前回のスクレイピングとの差分があれば編集があったとみなし，再びスクレイピングを行い，もし前回のスクレイピングとの差分がなければ，何もしないというのを繰り返す．このスクレイピングにおいて差分があったとみなされた回数を編集が行われた回数 (更新回数) とし，そのスクレイピングが行われた時刻を編集が行われたタイムスタンプ (更新時刻) とする．

3.3　編集履歴と実行結果

スクレイピングによって取得したソースコードをそのまま表示するだけでは，どこを編集したのかを目視で判別するのに時間がかかるため，差分を分かりやすく表示するような設計を心がけた．今回は UNIX 系の diff コマンドに「-u」オプションを付与することで取得することのできるユニファイド形式

(unified format) を用いた．編集履歴画面にはデフォルトでは受講者が編集を行う前のソースコードが表示されており，「次へ」ボタンをクリックすることで，編集履歴に基づいて挿入や削除がハイライトされた状態で表示される．

またそれぞれの編集履歴において，出力結果がどうなっているかも同時に確認することができるよう，編集履歴の横に実行結果 (キャンバス) を表示するような設計にした．これによって講師はコードの編集が妥当であったかを実行結果と照らし合わせて評価することができる．例えば図 2 と図 3 はギリシャの国旗を描画する課題の編集履歴確認画面の様子である．「次へ」ボタンをクリックするだけで，国旗がどの部分から描かれたのか，どのコードを追加してどの部分の描画が変化したのか，などを簡易的ではあるが十分に把握することができると考えられる．

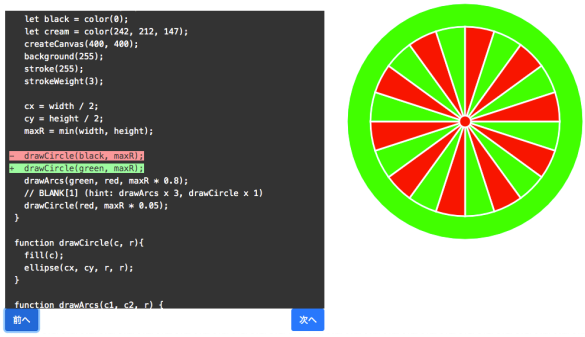

図 4　ダーツのボードを描画する課題-1

また別の例として図 4 はダーツのボードを描画する課題の編集履歴の確認画面である．受講者はおそらく「drawCircle」という関数の挙動を確認するため，引数の「black」を「green」に変更している．

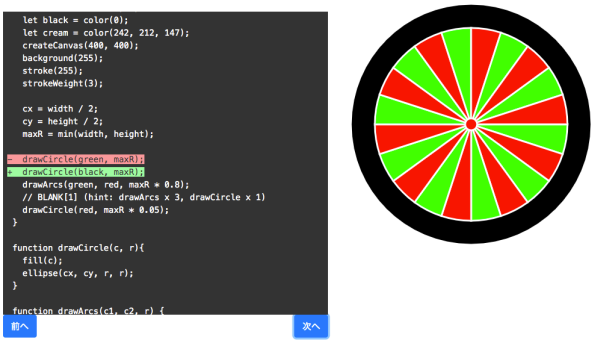

図 5　ダーツのボードを描画する課題-2

図 5 では「drawCircle」という関数が外側の円を描画するための関数であることに気付いたため再び引数を「black」に戻している．

図 6 では「drawArcs」という関数で内側の扇型を描画する処理を記述し，それがうまく動いていることが分かる．

最後に図 7 では真ん中の緑の円を付け足す処理を記述し，ダーツのボードを完成させていることが分かる．

また，図 8 は EU の国旗を描画する課題の編集履歴の確認画面である．

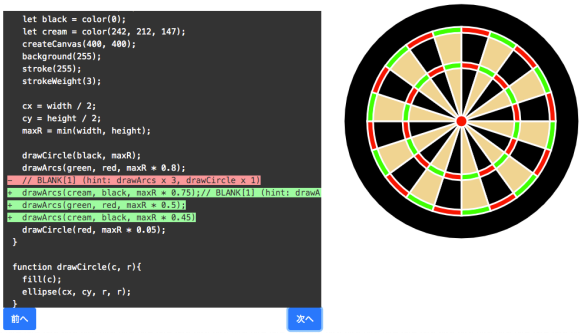

図 6　ダーツのボードを描画する課題-3

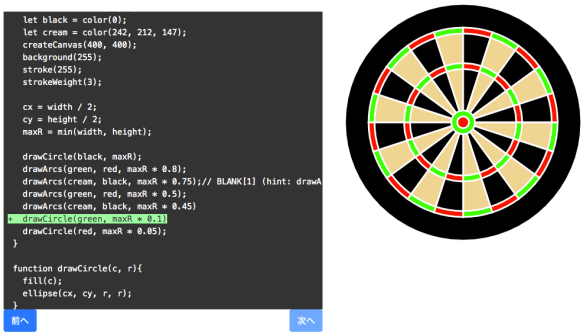

図 7　ダーツのボードを描画する課題-4

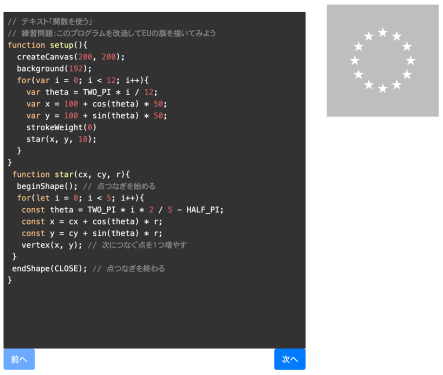

図 8　EU の国旗を描画する課題-1

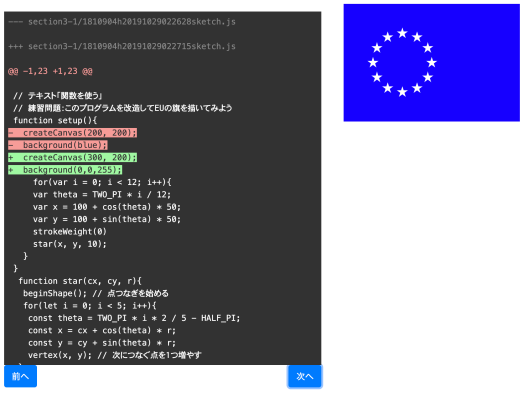

図 9　EU の国旗を描画する課題-2

図 9 では国旗の形を再現するために canvas の横幅を広げ，さらに背景色を変更していることが追加されたコードから分かる．

次に 10 では star という関数に渡す引数である x 座標の値を

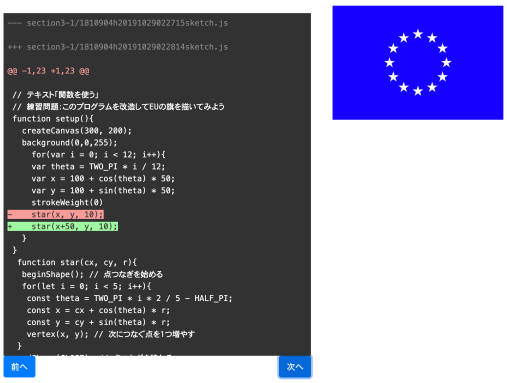

図 10　EU の国旗を描画する課題-3

調節し，星をキャンバスの中央に移動させている．つまり受講者が関数の使い方や引数の意味を理解できているということがこの画面から分かる．

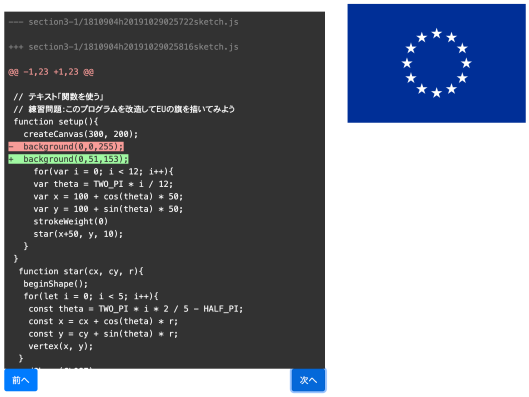

図 11　EU の国旗を描画する課題-4

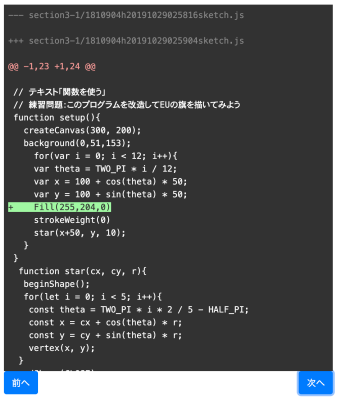

図 12　EU の国旗を描画する課題-5

その後，受講者は図 11 で背景の色を変更した．しかし，図 12 で追加した行によってキャンバスに何も表示されなくなってしまう．これは p5.js の組み込み関数である fill 関数を誤って「Fill」と記述してしまったため，実行エラーとなっているためである．

受講者は図 13 のように追加した行を削除している．その後正しく fill 関数を呼び出し，星を白から黄色に変更することに

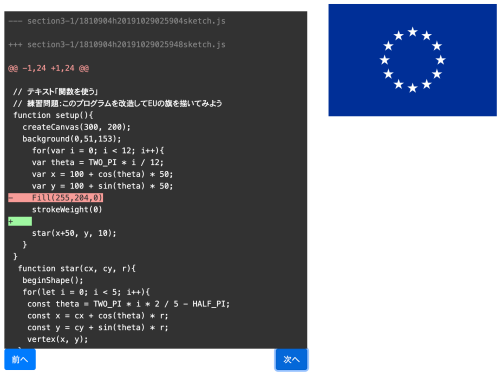

図 13　EU の国旗を描画する課題-6

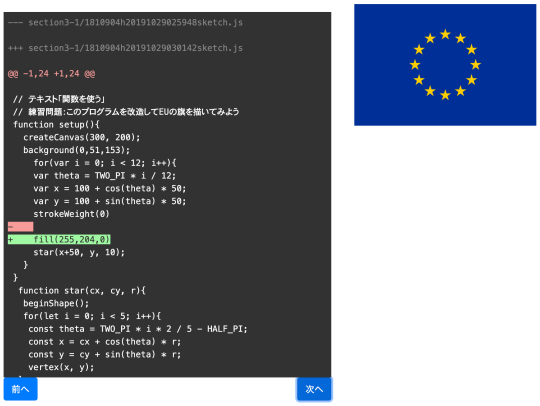

図 14　EU の国旗を描画する課題-7

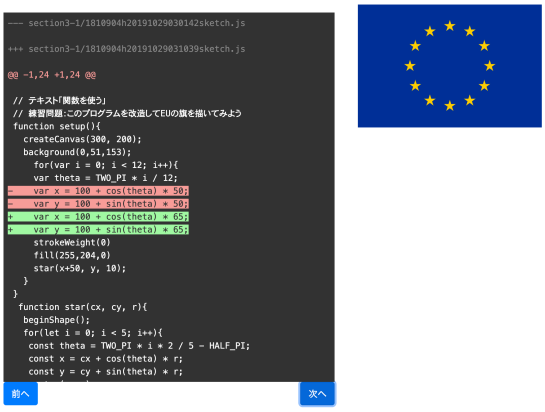

図 15　EU の国旗を描画する課題-8

成功していることが図 14 から分かる．

その後，図 15 において x と y という変数のどこを変更すれば星の円が広がるかどうかを理解できていることも確認できる．

このように，受講者がコードのどの部分を変更し，それによって出力がどのように変わったかということが直感的に理解できることで，受講者のコード変更の意図が明確となり，評価を助けるような設計になっている．

4　おわりに

本稿ではプログラミング学習，特に演習において，講師が評価を行う際の課題の過程の把握を支援することを目的とした

システムの開発を行った．これまでの関連研究では様々なアプローチでプログラミング課題の過程の評価・把握を可能にしているが，本稿ではより簡易的なプログラミング演習課題の過程評価支援システムの開発を目指した．講師に掲示する情報はソースコードの編集履歴・更新時間・更新回数・実行結果 (キャンバス) の 4 つのみを採用し，必要最低限の情報だけを表示し見やすいビューにしている．特殊な開発環境を用意する必要がなく受講者のソースコードのスクレイピングさえできれば本システムを利用できることが大きな特徴である．

今後の課題として，講師が課題を評価する際に何を求めているのかを明確にすべきという点が挙げられる．これを明確にするには，実際に今回作成したシステムを演習を行う講師に使用してもらいながらフィードバックを受ける必要があると考える．またこれまでの関連研究で実現されているリアルタイムに受講者の学習を助ける機能などと組み合わせて本システムを利用することも課題として挙げられる．

謝　　辞

本研究の一部は科研費 (19K03000) の支援による．ここに記して謝意を表す．

文　　献

[1] 小学校プログラミング教育の手引 (第二版), 2018 年 11 月, 文部科学省.
[2] 藤原理也, 田口浩, 島田幸廣, 高田秀志, & 島川博光. (2007). ストリームデータによる学習者のプログラミング状況把握. DEWS2007 D9-5.
[3] 井垣宏, 齊藤俊, 井上亮文, 中村亮太, & 楠本真二. (2013). プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案. 情報処理学会論文誌, 54(1), 330-339.
[4] 彦坂知行, & 北英彦. (2015). 多人数でのプログラミング演習における学習者のコンパイルエラー状況の把握システム. In 2015 PC Conference.
[5] 諏訪正則, 倉澤邦美, 鈴木惠介, 森本康彦, 横山節雄, 佐々木整, ... & 東京学芸大学. (2003). プログラミング教育における学習履歴取得システムの開発. 第 2 回情報科学技術フォーラム (FIT2003) 講演論文集, 583-584.
[6] 田口浩, 糸賀裕弥, 毛利公一, 山本哲男, & 島川博光. (2007). 個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援. 情報処理学会論文誌, 48(2), 958-968.
[7] 加藤利康, & 石川孝. (2013). プログラミング演習支援システムにおける学習状況把握機能の提案. 研究報告コンピュータと教育 (CE), 2013(2), 1-8.
[8] 加藤利康, & 石川孝. (2014). プログラミング演習のための授業支援システムにおける学習状況把握機能の実現. 情報処理学会論文誌, 55(8), 1918-1930.
[9] 蜂巣吉成, 吉田敦, & 阿草清滋. (2014). プログラミング演習におけるコーディング状況把握方法の考察. 研究報告コンピュータと教育 (CE), 2014(3), 1-8.
[10] 藤原賢二, 上村恭平, 井垣宏, 吉田則裕, 伏田享平, 玉田春昭, ... & 飯田元. (2018). スナップショットを用いたプログラミング演習における行き詰まり箇所の特定. コンピュータ ソフトウェア, 35(1), 1_3-1_13.