JSiSE Research Report
vol.34,no.3(2019-9)

オブジェクト指向プログラミング学習のための グラフィックスを題材とした Web 演習システムの実装

朝野有也[*1]，香川考司[*2]
*1 香川大学大学院工学研究科
*2 香川大学創造工学部

Implementation of a Web Exercise System with Graphics Programming for Learning Object Oriented Programming

Yuya ASANO[*1], Koji KAGAWA[*2]
*1 Graduate School of Engineering, Kagawa University
*2 Faculty of Engineering and Design, Kagawa University

あらまし：オブジェクト指向プログラミング（OOP）言語を学習する際に，学習者は言語の基本構文に加えて，構文に対応するオブジェクト指向を構成する概念を理解する必要がある．学習者が実例を通して概念自体を理解したとしても，その概念の有用性や概念に対応する構文の理解までは至らない場合がある．また，OOP の設計技法を理解するには OOP の概念の理解が不可欠である．本研究では，前述した概念の有用性や構文，設計技法をグラフィックスプログラミングを通して理解を促す Web ベースの学習支援システムの開発を行った．

キーワード: プログラミング学習，オブジェクト指向，Java 言語，問題出題，Web ベース

1. はじめに

情報系の大学では，応用的なプログラミング学習として，しばしばオブジェクト指向プログラミング(OOP)が題材とされる．継承やカプセル化などの概念があり，それらは講義や演習などの短期間では初学者にとって理解が難しい．そのような概念を視覚的に学習できる開発ツールとして，BlueJ(1)や Greenfoot(2)がある．しかし，これらは静的なオブジェクトの関係を視覚化することで学習者の理解を促すことに重点を置いているため，ツール単体で OOP の技法学習まで網羅できない．また，OOP の設計技能は経験によって培われるものであり，体系的な設計技法はデザインパターンとしてまとめられている．この設計技法は規模の大きなプログラムであるほど効果を発揮するが，上記の可視化に重点を置いている既存ツールでは扱うのが難しいといえる．

本研究では，OOP 初学者に対して，OOP を構成する概念や設計技法を習得できる演習問題を提供する Web ベースの学習システムの開発を目的とする．手続き型のようなモジュール化とは異なるオブジェクト指向の再利用性という利点を体験するために，規模の大きなプログラムであり視覚的に動的なオブジェクトの状態を理解できるグラフィックスプログラミングを題材とする．

2. 既存の学習支援ツール

BlueJ(1)は Java を使用言語としたオブジェクト指向の概念をデータ構造の可視化という手法で学習するツールである．学習者はクラス構造を図示したエディタを操作してプログラムを作成する．クラスのテンプレートの提供やディレクトリ構図の構築をツールが代行することで，言語仕様による躓きの可能性を最小限に抑える仕組みがなされている．ただし，プログラム構築にクラス構造の図式を利用するため，大きなプログ

ラムの作成に向いていないことや OOP の設計技法を学習するには教育者側から適切な課題を提供する必要がある．

Greenfoot[2]はクラス構造の図式のエディタを利用して，2 次元のグラフィカルなゲームの開発環境を学習者に提供する．グラフィカルな題材とすることで，学習者の学習意欲向上を期待している．ゲーム開発を容易とする API，実際の物理世界に沿った空間をベースとしたゲーム開発という学習者が理解しやすいシナリオが用意されている．しかし，BlueJ と同様に設計技法に関しては，教育者側から適切な課題を用意する必要がある．

CodinGame[3]は複数のプログラミング言語に対応したオンラインプログラミング学習プラットフォームである．学習者は提供されるプログラミング問題を解き，実行結果をゲームの画面のような動画で確認することができる．解答には言語ごとの標準出力機能を利用して，条件を満たす文字列を出力することで正誤の判定を行っている．競技プログラミングとしての側面が強く，OOP の学習には適していない．

3. システム概要

本システムでは，OOP 言語である Java の基本構文を学習した OOP 初学者を対象としている．前述した学習ツールの問題点である設計技法や演習課題の提供の問題を解決し，且つ利点であったオブジェクトの可視化を踏破することを到達目標とする．また，学習者の事前準備の負担を軽減するために Web ベースのシステムとして開発する．

概念や設計技法などのオブジェクト指向の特徴を活かすにはデータ構造が複雑であり，オブジェクト間のメッセージの送受信が多いほうが良い．本システムでは，規模が大きく，且つデータ構造が複雑化することで生じる動的データ構造の把握の難化を緩和するためにグラフィックスを演習課題の題材として提供する．演習課題の流れは，図 1 に示すように予めシステムが用意したライブラリを利用し，課題毎に提示された条件を満たすオブジェクトを生成または操作するコードを学習者が作成する．そのコードをシステムに提出し，課題の全ての条件を満たすまで，試行錯誤的に編集と提出を繰り返す．全ての条件を達成するコードを提出することで演習の完了とする．

また，グラフィックスに関する処理はシステムから提供されるライブラリで予め実装されており，これにより OOP 学習に直接関係ないグラフィックスに関する処理を学習者に意識させない．実行結果は図 2 に示すような Web ページ上でアニメーションとして提示される．アニメーションと同期して，条件達成の正誤情報が更新され，一覧として学習者に提示される．

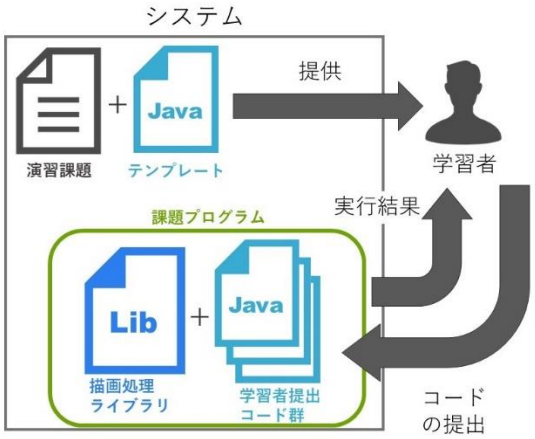

図 1 演習課題の構成

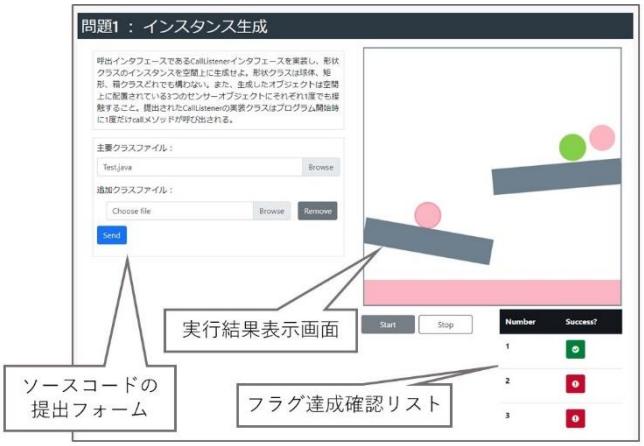

図 2 演習課題画面

4. システム実装

本システムでは，前述したようにプログラムの実行結果をアニメーションとして表示する必要がある．アニメーションの表示にはブラウザ上でプログラムを実行し，学習者に表示する手法を選択した．Java のプログラムをブラウザ上で実行するには，従来 Java アプレットが利用されていたが，2018 年 9 月にリリースされた Java11 で廃止された．GWT[4], DoppioJVM[5],

TeaVM[6]はブラウザ上での Java プログラム実行を可能にするツールである. GWT と TeaVM は Java プログラムのコードをブラウザ上で実行できる JavaScript コードに変換するツールであり，DoppioJVM はブラウザ上で動作可能な JavaScript 製の JVM である.

今回はコンパイル速度や代替 Java 言語に対応可能，マルチスレッドのサポートなどの観点から TeaVM を採用した. TeaVM は Maven や Gradle などの Java のビルドツールにより導入することができる. 本システム開発では Gradle により TeaVM の導入を行う. 本システムでは開発版の TeaVM を利用する. バージョンは 0.6.0-dev-674 である.

提出から JavaScript コードに変換するまでの詳細な流れを図 3 に示す. 学習者が提出したコードはまず Java バイトコードに変換され, 予め用意されていたグラフィックスライブラリとともに，TeaVM によって JavaScript コードに変換される. 変換されたコードはクライアント側に返信され, ブラウザ上で実行される. 実行結果は図 2 の演習課題画面上のキャンバスに描画される.

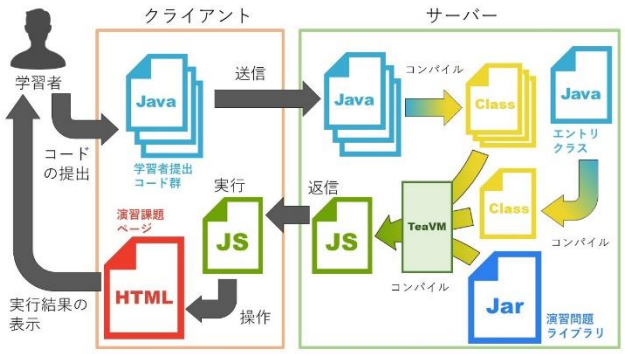

図 3 システムの構成

また，本システムで利用されるグラフィックスライブラリでは, 図 2 の実行結果表示画面で描画されるような空間と空間上に存在するオブジェクト群を提供する. このライブラリは JBox2D というサードライブラリを利用している. JBox2D とは，物理法則をシミュレートする 2D 物理エンジンである Box2D を Java で動作するように移植されたものである. 本システムでは，オブジェクトの落下やオブジェクト同士の衝突などを演習課題として取り扱う. 学習者には, JBox2D の API をラップしたクラスを利用させることで，JBox2D 特有の構文を意識させないようにする. ライブラリが提供するオブジェクトは図 4 に示すような矩形，円形，中抜きの矩形の 3 種類である. また，それぞれのオブジェクトは，実体が存在し，衝突時に跳ね返り処理と任意に設定した衝突イベントが実行される. また，実体がなく衝突イベントのみ実行されるセンサとしての役割を持たすこともできる. 前述した 3 種類のオブジェクトは，グラフィックライブラリ内ではそれぞれに対応する形状クラスが用意されている. この形状クラスには，オブジェクトを移動させるためのメソッドやオブジェクトが存在する座標を取得するメソッドなどが実装されおり，学習者はこれらのメソッドを利用して課題を攻略する. 学習者が利用する他のクラスとしてワールドクラスが存在する. このクラスはオブジェクト群が存在することになる空間そのものに対応する. 物理法則の管理や空間上への形状オブジェクト生成を行い，学習者はこのクラスを介して空間に任意のオブジェクトを生成する.

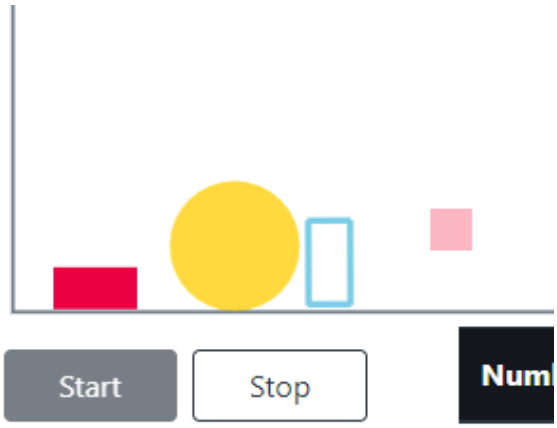

図 4 オブジェクトの種類

このライブラリは，学習者が提出コード内で利用するだけでなく，システム管理者が新たに演習課題の対象であるグラフィックスプログラムを実装する際にも利用される.

5. 試用実験と評価

開発したシステムに対して, 学習効果を見込めるか, 操作性に問題がないかを確かめるために実験評価を行った. 実験評価では，香川研究室の院生 2 名，学部生 2 名にシステムに関する説明を行った後，2 つのサンプル問題に取り組んでもらい，その後インタビューを行った. インタビューでは「既存の演習系 Web アプリケーションと比較して良い箇所・悪い箇所」,「操作性

で良い箇所・悪い箇所」,「その他の良い箇所・悪い箇所」の 3 項目について質問した.

インタビューに取り組んでもらった 2 つのサンプル問題は，共にオブジェクトを指定のセンサオブジェクトに接触させるというものであった．1 つ目の問題はオブジェクトを生成することで条件を達成させ，2 つ目の問題は既存のオブジェクトを移動させることで条件を達成させるという制約をつけた.

インタビューの結果を分析したところ，既存の演習系の Web アプリケーションとの比較に関しては, 実行結果のオブジェクトの可視化がプログラムの挙動の理解に役立っているという意見を多く得ることができた. 悪い箇所として，実行までに必要な動作が多いという意見が目立った．操作性の良い箇所として，応答速度の速さが挙げられた. TeaVM によるコンパイル速度は学習者にとって不自由となっていないといえる．しかし，扱うプログラムの規模によって処理速度が変わるため，規模と処理時間の関係を調査する必要がある．また，その他の良い箇所として，少ないコードの記述量でグラフィックスプログラムを扱えることから達成感があるという意見があった．グラフィックスプログラムを題材とすることで学習意欲向上に役立っているといえる．その他の悪い箇所としては，座標軸などオブジェクト操作のために必要な情報が少ない点が挙げられた.

6. おわりに

本研究では，グラフィックスプログラムを題材とした OOP 初学者向けの演習型システムの開発を行った. 本システムでは, OOP の概念や設計技法を学習するために，規模の大きな題材としてグラフィックスプログラムを題材として扱う．試用実験では，サンプル問題を実装し，システムの効果や操作性などの評価を行った.

試用実験から得られたフィードバックから今後の課題を検討した. 本システムで扱う演習課題を検討する. 演習課題は OOP の概念や設計技法の有用性を実感でき，学習者が実装する部分において把握できる規模のものを選択する．それに伴い，実現できる演習課題を増やすためにグラフィックスプログラムのライブラリの拡張を行う必要がある．また，試用実験の評価を受け，実行までの手順簡略化，プログラム実行中に提供する情報の拡張などの UI の改善を行う.

謝辞

本研究は JSPS 科研費 15K01075 の助成を受けたものです.

参 考 文 献

(1) M. Kölling, B. Quig, A. Patterson and J. Rosenberg, “The BlueJ system and its pedagogy”, Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, Vol 13, No 4, pp. 249-268, Dec. 2003.

(2) M. Kölling, “The Greenfoot programming environment”, ACM Transactions on Computing Education, Vol 10, Issue 4, Article No 14, Nov. 2010.

(3) CodinGame, “CodinGame”, https://www.codingame.com, Retrieved August 2019

(4) R. Dewsbury, “Google Web Toolkit applications”, Prentice Hall, 2017.

(5) J. Vilk and E. D. Berger, “Doppio: breaking the browser language barrier”, Proc. of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '14). pp. 508-518, 2014.

(6) A. Andreev et al., “TeaVM”, http://teavm.org, Retrieved August 2019.