競技性・観戦性を拡張したプログラミングゲームの提案

岡　大貴[1,a)]　西田 健志[1,b)]

概要：本研究では，メディアアートの文脈におけるライブコーディングの要素を取り入れた，プログラマ同士が戦う対人形式のプログラミングゲームを提案する．非プログラマでもプログラミングの観戦をわかりやすく楽しめるエンタテイメントを創出することで，プログラミングとプログラマにより多くの人に関心を持ってもらい，プログラミングをより一般的な行為にすることを目指す．

1. はじめに

自然言語での読み書きや健康のために行う運動のように，誰もがプログラミングやその考え方をある程度身につけていることが望ましいと考えられるようになっている．今年の 4 月からの小学校におけるプログラミング教育の必修化がその最たる例である．急速に情報化が進み，IoT デバイスやスマートスピーカーなどコンピュータと密接な関わりのある現代において，プログラミングとは何かを知らず，コンピュータが何を行っているか，その設計者が何を考えて実装を行ったかなどを完全にブラックボックス化してしまうのは問題である．

しかし，最初から楽しさを感じやすい読み書きや運動とは異なり，プログラミングそのものの楽しさは習熟が進むまで実感することが難しいという問題がある．よく書かれた小説やプロスポーツでのファインプレーには初心者でも心動かされるもので，そうした小説家やスポーツ選手への憧れが自ら書くことや体を動かすへの感心を高めるが，プログラミングの場合にはそのような憧れを持つ機会に乏しいのが現状である．プログラミング力を競うプログラミングゲームも数多く開発されているが [1], [2]，プログラミング初心者が観戦して楽しめるよう十分に配慮されているとは言い難い．

本研究では，高レベルのプログラマがそのプログラミング技術によって力量を競うことができ (=競技性)，プログラミングにまだ習熟していない人でも観戦して楽しむことができる (=観戦性) プログラミングゲームを開発する．競技性と観戦性を併せ持つゲームを通じて，レベルの高いプログラマに憧れを抱くようになり，自らプログラミングを行うことへの興味感心を高めることを目指す．

本研究で提案するプログラミングゲームの競技性と観戦性を高めるための設計指針は次の 3 点である。

(1) プログラミングを魅せるデザインをする
(2) 戦略に多様性を持たせる
(3) リアルタイムな駆け引き・アドリブを取り入れる

本稿では，提案するプログラミングゲームのプロトタイプについて紹介し，設計指針に基づくデザインの詳細や競技性・観戦性を評価する実験の計画について議論する．

2. 関連研究・関連システム

プログラミングゲームは数多く存在する．中でも本研究の提案システムと類似しているのは Robocode[1] や Fight Code[2] などのロボットバトルシミュレーション型プログラミングゲームである．これらは Java や JavaScript で二次元の盤面におけるロボット AI の思考ルーチンをプログラミングし，ロボット同士を戦わせるというものである．このようなシステムには根強い人気があるが，プログラムが長く複雑になり，「プログラムを観る」という観点では特に非プログラマにとってそれを読解することが困難であると考えられる．またあらかじめ記述したプログラムを戦わせ，ロボットの振る舞いを見守るという静的なゲーム展開であり，観戦には向かないと考えられる．なお日本語に対応したものが少なく，日本語が母国語であるプログラマにとって利用しづらいという問題点もある．

プログラミングゲームを活用した研究もいくつか行われており，Josha らの研究 [3] ではプログラミング初学者の問題解決能力を向上させるためのシステムにプログラミングゲームを用いている．また水口の研究 [4] ではプログラミングの講義における成績評価にロボットバトルシミュレーション型のプログラミングゲームを活用している．三谷らの研究 [5] ではプレイヤキャラクタを操作するプログラミ

1 神戸大学
a) hirokioka@stu.kobe-u.ac.jp
b) tnishida@people.kobe-u.ac.jp

ングゲームでプログラミングスキルの向上を図っている．また増谷らの開発した VLogic[6] では VR 空間上にブロックベースのプログラミングゲームを実装することで，手足を使ってプログラミングを体験することができ，非プログラマにプログラミングに関する興味を持たせている．これらはプログラミングスキルが高くなくても使用でき，ゲームとしても面白みがあるが，第三者が観るように設計されておらず，スポーツなどの観戦する競技で見られるリアルタイムな駆け引きやアドリブなどが存在しない．なお筆者らはプログラミング初学者のステップアップを目標に，クイズ・占いといったエンタテインメントの要素を取り入れたコードリーディング促進システムに関する研究を行ってきた [7]．本研究ではプログラミング初学者のみならず，より多様なプログラミングに携わる人々をターゲットに研究を行った．

プログラミングを用いたエンタテインメントとしては，Topcoder[8] などの競技プログラミングやハッキングコンテスト [10]，コードゴルフ [9] などがある．これらはプログラマの間で根強い人気があり，プログラマ個人に焦点を当てるという観点で優れたシステムといえる．しかしながら，参加するには数学やセキュリティ，コンピュータサイエンスの知識を要するため，非プログラマにとってはそれを観て理解するのは難しいと考えられる．

またリアルタイムにプログラムから視覚的な出力を行うビジュアル・ライブコーディング環境としては，Livecode-Lab[11]，Hydra[12] などが挙げられる．これらはライブコーディング環境であり，本研究で提案するシステムとは目的・用途が異なるが，プログラムから即興的に視覚的出力を行うシステムという上では類似している．これらのシステムはプログラムと出力の両方を見せるエディタを備えているため，「プログラムを魅せる」デザインを設計する上で参考にした部分が多い．

3. 設計指針

本システムではより多くの人にプログラミングに関心を持ってもらい，プログラマ個人に焦点を当てるため，以下の設計指針を設けた．

3.1 プログラミングを魅せるデザインをする

本システムで行うのはあくまでもプログラミングであり，その観戦性を高めたい．よってプログラマがプログラミングしている様子やプログラムそのものをライブコーディングにおける「show us your screens」のように見えるようにすることが望ましい．

3.2 戦略に多様性を持たせる

プレイヤの戦略が一辺倒になってしまっては，観戦していても面白みがない．そのため，多様な戦略・戦法を可能にする．

3.3 リアルタイムな駆け引き・アドリブを取り入れる

競技において，プレイヤ同士のリアルタイムな駆け引きや，予測できなアドリブは，観戦者を盛り上げる重要な要素である．非プログラマにプログラミングを観戦させるため，プログラマ同士が駆け引きし，アドリブでプログラミング可能な設計を行う．

4. 提案システム

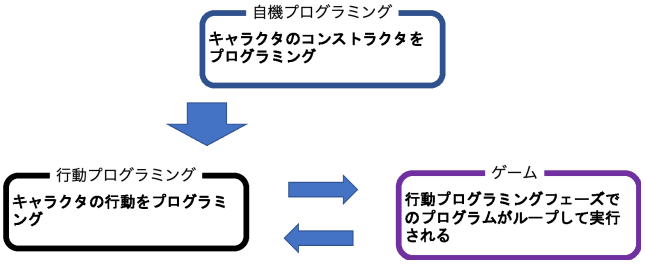

図 1 フェーズの遷移

本研究では非プログラマがゲーム展開やプログラムと出力の相関を理解しやすいように，「相手に攻撃を当て，自分は相手の攻撃を避けて，相手を倒す」というプリミティブなゲームであるシューティングゲームを採用した．本システムは web アプリケーションとして実装し，指定の URL にアクセスすることで使用できる．対 CPU モードと対プレイヤモードがあり，どちらかを選択できる．対 CPU モードの場合はコンピュータと対戦することができ，対プレイヤモードでは対戦を希望しているプレイヤ同士をマッチングし，対戦することができる．ゲームは以下の 3 つのフェーズに分けられる．なお本システムで使用できるプログラミング言語は現状，JavaScript のみである．

4.1 自機プログラミングフェーズ

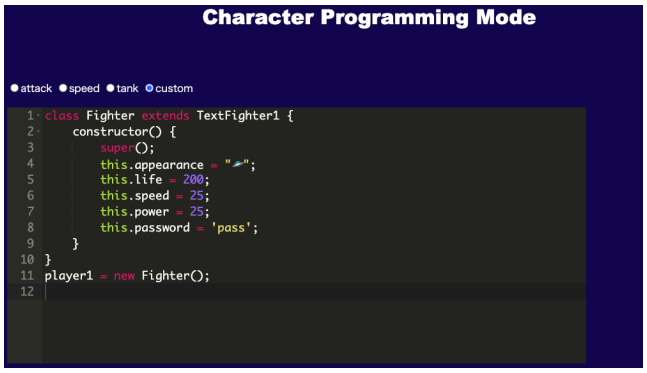

図 2 自機プログラミングフェーズの様子

ゲームが開始すると初めにこのフェーズに移行する．こ

こではプレイヤが操作するキャラクタのコンストラクタがエディタに表示されており，プレイヤはパラメータを自由に書き換えることができる．パラメータにはキャラクタの外見，体力，スピード，攻撃力，パスワードがある．外見はキャラクタの容姿であり，任意の文字列を指定することができ，絵文字も使用可能である．体力はいわゆる HP(ヒットポイント) を表しており，非負の整数を指定できる．これが 0 になるとゲームに敗北する．スピードはキャラクタが行動をできる回数である．これも非負の整数を指定でき，大きいほど多く行動できる．攻撃力はキャラクタが攻撃をする際にどれだけ相手キャラクタの体力を減らせるかを表した数値であり，非負の整数を指定できる．パスワードはのちに説明する「ハック」という要素の中で重要なパラメータであり，文字列を指定できる．この値が相手プレイヤに知られてしまうと，記述するプログラムが相手に丸見えになってしまう．各パラメータを記述し終わると行動プログラミングフェーズへ移行し，以後はゲームが終わるまで行動プログラミングフェーズとゲームフェーズを行き来する形となる．

4.2 行動プログラミングフェーズ

図 3 行動プログラミングフェーズの様子

このフェーズに移ると，自機プログラミングフェーズで記述したコンストラクタをもとにプレイヤキャラクタのインスタンスが生成される．プレイヤはプログラムをエディタに記述し，生成されたキャラクタを操作する．プレイヤは条件分岐や繰り返しなどの従来の文法の他，独自に用意されたプロトタイプメソッドを使うことができる．今回用意したメソッドはキャラクタが前方に攻撃を行う shot メソッド，キャラクタを移動する moveUp(キャラクタを上に移動)，moveDown(キャラクタを下に移動)，randomMove(上か下のどちらかにランダムに移動) などのメソッドである．また変数として自機プログラミングフェーズで設定したプロパティ以外にも，キャラクタの進行方向を示す「direction」やキャラクタの座標を示す「position」などのプロパティを使用することができる．プレイヤはこれらを用いてキャラクタの行動を制御するプログラムをエディタに記述する．エディタには文字数制限があり，長大なプログラムは記述することができない．キャラクタを壁に当たるまで上下に動かして攻撃するプログラムの例を以下に示す．

```
 if(player.direction === 'top'){
        player.moveUp();
 }else if(player.direction === 'bottom'){
        player.moveDown();
}

if(player.position <= 0){
        player.direction = 'bottom';
}else if(player.position >= height){
        player.direction = 'top';
}
player.shot();
```

またこの段階ではプレイヤは相手プレイヤがどのようなプログラムを記述したかを見ることはできない．記述したプログラムは次のゲームフェーズにおいてループして実行される．またループの回数は自機プログラミングフェーズで設定したスピードの大きさに比例する．

さらにここではプログラムを記述するだけでなく，「ハック」を相手プレイヤに仕掛けることができる．このゲームにおける「ハック」とは相手プレイヤの設定しているパスワードを推測し，相手プレイヤの記述したプログラムを開示させることである．プレイヤは画面下部のコマンドラインからコマンドを入力し，相手プレイヤに対してハックを行うことでゲームを有利に進めることができる．このハック機能はまだ試作段階であるが，予測したパスワードをもとにハックする guess コマンドと総当たり的に相手のパスワードを調べる bruteForce コマンドがある．プログラムを記述し終わった時点でゲームフェーズに移行する．

4.3 ゲームフェーズ

このフェーズでは行動プログラミングフェーズで記述したプログラムが 10 秒間ループして実行され，ゲームが進行する．各キャラクタはプログラムに基づき行動し，移動したり，他方のキャラクタを攻撃したりする．このフェーズで相手キャラクタに攻撃を当て，体力を 0 にした方のプレイヤの勝利となる．勝敗が決まるまで行動プログラミングフェーズとゲームフェーズがパラメータを引き継いで交互に繰り返される．従来のプログラミングゲームはあらかじめ作成したプログラムによる展開を見守る静的なものが多かったが，フェーズ制を導入することにより，相手の

図 4 ゲームフェーズの様子

前のフェーズにおける戦法やキャラクタのパラメータを鑑みて次のフェーズでの行動を決めるという，駆け引き・アドリブの要素を生み出せると考える．またゲーム画面は LivecodeLab や Hydra などのライブコーディング環境を踏襲し，プログラムの出力を見せつつもその上にプログラムそのものが表示されるようにした．

5. 想定シナリオ

本システムは 1 人あるいは 2 人のプログラマにプレイされることを想定している．従来のゲームの形式のようにオンラインでプログラマ同士が対戦し，それをプレイヤでない人も観戦する，というのが想定する 1 つの利用シナリオである．また本システムはメディアアートにおけるライブコーディングのような利用も想定している．すなわち，あるスペースで 2 人のプログラマが対戦し，それを壁面やスクリーンにプロジェクタなどで投影し，複数の観客が観戦し楽しむというケースである．このようなケースが広がれば，よりプログラマが活躍できる機会を提供でき，また新たなラップトップパフォーマンスを開拓することにつながると考えらえる．

6. 実験計画

今後は提案システムが観戦可能であったか，ゲームとして奥行きがあったかを調査するため評価実験を行う予定である．プログラマにプレイしてもらうことで，今回設けたゲームシステムや制約が競技に活きていたかをインタビュー・アンケート調査する．またプレイを非プログラマに観戦してもらい，プログラムを理解できたか，プログラミングに興味が持てたかなどをインタビュー・アンケート調査し，フィードバックを得ることで，ゲームシステム，UI 等についても改善を加えていく．なおこれらを踏まえた上で，実際のライブコーディングのようにあるスペースで複数人の観客の前でシステムをプレイすることがどのような作用をプレイヤ・観戦者に引き起こすかを観察するケーススタディを行いたいと考えている．

7. 今後の展望

今回は JavaScript のみが使用可能であったが，Python や Java，Lisp など多様な言語でのプレイを可能とし，プログラマが自身の得意な言語で対戦したり，異なる言語での対戦が可能となるようにアップデートを行う予定である．またユーザ登録やランキング，観戦モードを導入し，よりゲームが盛り上がるような工夫を加える．なお今回はゲームをフェーズごとに区切った実装を行ったが，メディアアートにおけるライブコーディングのように即興でキャラクタの行動をプログラムし，対戦するというようなライブコーディングモードも追加したいと考えている．

8. まとめ

本研究では，より多くの人にプログラミングに関心を持ってもらい，プログラマ個人が注目される場を作るため，ライブコーディングの要素を取り入れたプログラミングゲームを提案した．今後は本システムを使用してもらい，そのフィードバックを取り入れてシステムを改善する．そして評価実験を行い，システムを用いることのメリットや課題について議論したい．

参考文献

[1] Robocode, https://robocode.sourceforge.io/.
[2] Fight Code, http://fightcodegame.com/.
[3] J.Shi et al: Pyrus: Designing A Collaborative Programming Game to Promote Problem Solving Behaviors, Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, No. 656, pp. 1–12(May.2019).
[4] 水口 充： 成績評価のためのプログラミングゲームの設計と実践, 研究報告エンタテインメントコンピューティング (EC), vol. 2016, pp. 1–7(July.2016).
[5] 三谷将大, 寺田 実: Web アプリケーションによるゲーミフィケーションを用いたプログラミング上達支援システム, 第 27 回インタラクティブシステムとソフトウェアに関するワークショップ, (Sept.2018).
[6] 増谷海人, 赤澤紀子： 仮想現実を用いた初学者向けプログラミング学習システムの提案, 2018 年度 情報処理学会関西支部 支部大会 講演論文集,vol. 2018, (Sept.2018).
[7] 岡 大貴, 西田健志: ゲーミフィケーションによるコードリーディングの促進, エンタテインメントコンピューティングシンポジウム 2019 論文集, vol. 2019, pp. 346-349 (Sept.2019).
[8] Topcoder, https://www.topcoder.com/.
[9] 浜地慎一郎: Code Golf, http://shinh.skr.jp/dat_dir/golf_prosym.pdf.
[10] SECCON, https://www.seccon.jp/.
[11] LivecodeLab, https://livecodelab.net/.
[12] Hydra, https://hydra.ojack.xyz/.