内容梗概

プログラミングが重要視されている昨今では，プログラミングの授業が小学校に導入されるなど，プログラミングの一般教養化が推し進められている．しかしプログラミングを楽しむためにはある程度の習熟を必要とし，学習中に挫折してしまうプログラミング初学者も少なくない．初学者でもプログラミングの楽しさを感じられるように学習コンテンツをデザインできれば，初学者の学習モチベーションを損ねることなく，プログラミング学習を継続させることができると考えられる．

そこで本研究では初学者の継続的なプログラミング学習を促すため，2 つのアプリケーションを開発した．1 つ目はエンタテインメントの要素を取り入れることによりコードリーディングを促進するソースコード閲覧システムである．このシステムではクイズ，占いといったエンタテインメントを交えて GitHub にあるソースコードを表示し，ユーザに読解させることにより，楽しみながらコードリーディングを促進することを目指した．実装したアプリケーションを使用し運用を行い，その後アンケート調査・ディスカッションを行った結果，クイズに関しては楽しかったという回答が得られたが，ややプログラミング初学者にとっては難解であるという意見が得られた．また日常的な使用を促すために更なる工夫が必要であり，これらの問題点を今後改善していく予定である．2 つ目はライブコーディングの要素を取り入れた対人形式のプログラミングゲームである．このシステムでは習熟度の高いプログラマ 2 人がプログラミングスキルによって力量を競って勝負することができ，プログラミング初学者でも観戦して楽しむことができるゲームを開発し，初学者のプログラミングに対する興味関心を高めることを目指した．評価実験では実際に 2 人のプログラマの対戦を初学者に観戦させ，プレイヤであるプログラマと観戦した初学者の双方にアンケート調査を行った．実験結果としてシステムのプレイ・観戦は楽しく，プログラミングに対する興味が高まったなどシステムに対する肯定的な意見が多く得られたが，ゲームシステム・デザインの改善点に関するコメントも多く得られた．またプログラマがプレイ時に記述したプログラムを分析した結果，戦略の多様化・より可読性の高いプログラムの表示などいくつかの改善点が見つかった．この結果を受け，今後は提案システムを改善すると共に，より多くのプログラミング初学者を含むプログラマにシステムを使用させ，より良いシステムの構築を目指す．

目 次

1　はじめに

高速に IT 化の進む近年では，スマートフォンの普及を皮切りに，IoT，AI, クラウドコンピューティングといった単語がメディアで散見され，コンピュータが着実に人々の生活を満たしつつあると言える．コンピュータが遍在する現代において，人とコンピュータの接点は圧倒的に多くなり，コンピュータを専門的に扱う人でなくともそれに触れて暮らすことが当たり前となり，コンピュータとの親和性を高めることが必要である．

近年ではその高速な社会の IT 化に伴い，プログラミングが重要性を増している．以前は「プログラミングはエンジニアや理系の学生がやるもの」というような，専門家だけの技能であるという認識が強かったが，プログラミングを英語のように一般教養とし，プログラミングに関わる仕事に就かない人もある程度理解していることが望ましいという風潮がある．国際的にもロシアでは 2009 年からプログラミング教育が導入され，英国では 2014 年から「Computing」というコンピュータサイエンス，情報技術，デジタルリテラシーの 3 分野からなる科目が導入されており [1]，日本においても今年度から小学校でのプログラミング教育が必修化されている [2].

プログラミングという行為は機能的であるだけでなく，自身のアイディアをコンピュータを介して表現する手段でもあり，プログラミングスキルを向上させることは自己表現の可能性を広げることにつながる．近年では Processing や openFrameworks，Arduino などの登場により，プログラミングにより創造的な表現を生み出す「クリエイティブ・コーディング」やメイカームーブメントが流行し，コンピュータ・サイエンス等の知識がなくともプログラミングによってものづくりをし，発表できる土壌が整いつつある．

しかしプログラミング教育の現場やその学習教材では，プログラミングの機能にのみ焦点が当たりがちで，プログラミングの「楽しさ」が疎かにされていることが少なくない．プログラミングをする上で楽しさを感じることは，学習の観点からも重要である．松本らによる C 言語プログラミングの授業では、学習の楽しさや、プログラミングへの興味が高いほど授業の学習率が高いことが示されている [3].

しかし最初から理解が容易で楽しさを感じやすい読み書きや運動とは異なり，プログラミングは楽しさを感じるまでにある程度の習熟を要する．よく書かれた小説などの文章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．

プログラミング初学者にプログラミングの楽しさを実感させるために設計された学習コンテンツは多数存在し，Scratch や Viscuit などのビジュアルプログラミング言語 (VPL) がその代表的な例である．これらは実際に学校教育で導入され，小・中学生の

プログラミング教育に貢献している．しかしそれらは小・中学生など若い年代をターゲットに設計されているため，高校生や大学生，それ以上の年代にとっては楽しさを感じにくい場合がある．また実践的なソフトウェア開発においてはテキストプログラミング言語 (TPL) を用いる場合がほとんどであるため，VPL と TPL の間に乖離があることも問題である．

本研究では従来のシステムがターゲットとしている世代よりも上の世代のプログラミング初学者を対象に，エンタテインメントの要素を用いることでプログラミングに対する抵抗感を減らし，プログラミングを楽しんでもらうためのシステムの開発を目指した．具体的には初学者のコードリーディングを促進するアプリケーション，プログラミングゲームを用いてプログラミングに対する興味喚起を行うアプリケーションの 2 つを開発し，それぞれ初学者を対象に運用を行うことでその効果を検証した．

本論文では以降，2 章で研究指針について述べた後，3 章でゲーミフィケーションを用いたコードリーディング支援システムについて述べ，4 章でプログラミング初学者の興味喚起を目的としたプログラミングゲームについて述べる．5 章で本論文をまとめる．

2　研究指針

本研究の目的は，エンタテインメントの要素を用いて初学者のプログラミングに対する抵抗感を無くし，プログラミングを楽しんでもらうことである．

初学者がプログラミング学習を挫折してしまう要因はいくつか考えられるが，本研究ではそのうち3つの要素に焦点を当てた．1つは「辛さ」である．プログラミング初学者向け教材では，記述されているプログラムを書き写す，いわゆる「写経」を行い，サンプルアプリケーションの作成などをする．これは学びの多い作業ではあるが，単調な作業になり，初学者にとっては苦痛を伴う場合がある．

2つ目は「煩わしさ」である．初学者が挫折する原因として多いのが環境構築が上手くいかず，学習を諦めてしまうことである．また特にコンピュータを日常的に使用する習慣がない場合はプログラミングから遠ざかりがちである．

3つ目は「難しさ」である．プログラミング学習と言ってもその内容は多岐に渡り，あるプログラミング言語の文法だけでなくオブジェクト指向等の概念やライブラリ，エディタ，IDE，VCS等ツール群の用法など多くを学ばなければならない．初学者にとって難解な概念も多く，これらを並行して学ぶことはモチベーションを下げかねない．

上記の問題を解決するため，以下の指針を設けた．開発するシステムにはこれに基づいた設計指針を設ける．

1. **エンタテインメントの要素を取り入れる**

   コンピュータを日頃使わない者にとって，プログラミングに触れる機会は少ないため，初学者はプログラミングをすることあるいはプログラムを読むことに慣れておらず，少なからず抵抗感があると考えらえる．本研究ではエンタテインメントの要素を取り入れることで，プログラムに触れることへの抵抗感を減らすとともに能動的な利用を促し，プログラムに触れることを楽しんでもらうことを目指す．

2. **利用のハードルを下げる**

   初学者にとって環境構築や，普段使用しないソフトウェアの導入などの手間は，プログラミングに対するモチベーションを下げる要因となる．本研究ではシステムを利用する際の手間を極力小さくすることで，日常的な利用を促進し，初学者がプログラミングと接する機会を増やすことを目指す．

3. **不要な負荷をかけない**

   複雑な概念や理解が何回な要素は初学者の挫折の要因である．本研究では極力前提知識などを少なくし，初学者がプログラムあるいはプログラミング飲みに集中

できるように設計する.

3　コードリーディング支援システムに関する研究

プログラミングの重要性の高まりに伴い，プログラミングを学ぼうとする人が増えている．そういった人には書籍や学習用 web サイトを通して学ぶ場合と情報系の学校やプログラミングスクールなどに通ってプログラミングスキルを身につけようとするパタンがあるが，学習にかかる費用や時間が取れないなどの理由から前者を選ぶ人が多い．

初学者向けプログラミング学習コンテンツが普及し，プログラミング入門のハードルは下がってきているものの，それらではプログラムを「書く」ことに千年させるものが多い．例えばプログラミング学習サイトである Progate[4] では，主要なプログラミング言語の文法やライブラリの用法など,「どうプログラムを書くか」を重点的に教えている．

しかしプログラミング学習において重要とされているスキルは様々である．プログラムを書くスキルだけでなく，論理的思考力，コードリーディングスキル，情報収集スキルなど多くの技術を必要とし，これらを網羅的に学習教材を通して身につけるのは難しい．

本項で提案するシステムでは，その中でもコードリーディングスキルに焦点を当て，エンタテインメントの要素を組み込んでプログラムを閲覧させることにより，プログラミング初学者のコードリーディングを促進するシステムを実装した．

3.1　関連研究

3.1.1　GitHub を利用した研究

GitHub を活用した研究はいくつかある．Emitza らの研究 [7] では GitHub 上のオープンソースプロジェクトにおけるコミットコメントから感情分析をし，その結果とプログラミング言語やコミットの時間などとの相関を調査している．永野らの研究 [8] では GitHub と StackOverflow のデータを用いて双方を利用するユーザについて調査している．この研究ではユーザが GitHub で作成したリポジトリと StackOverflow への投稿のコンテンツの関連性について調べたもので，双方への投稿コンテンツに一定の相関があることを示している．また柴藤らは GitHub 上の断片データに関する情報を取得できるシステムを開発している [9]．このシステムではユーザが指定したソースコード中の一部の連続したコードに関するプルリクエストを取得できるというものであり，ソースコードを閲覧する際の利便性を高めている．

3.1.2 ゲーミフィケーションを活用した研究

またエンタテインメントの要素，特にゲーミフィケーションを活用した研究も盛んである．一ノ瀬らの研究 [10] ではソースコード上の技術的負債を可視化し，さらにゲーミフィケーションの要素を加えることで技術的負債の除去を促している．この研究ではソースコードのファイル構造を街のように可視化し，技術的負債が存在するファイルを目立たせ，さらに技術的負債を取り除いたユーザをランキング形式で表示することによって，ゲーミフィケーションの要素を元に生産的な行動を促している．三谷らの研究ではキャラクタをプログラムで制御するプログラミ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 [6].

3.1.3 コードリーディング支援に関する研究

なおコードリーディング支援に関してもいくつか研究がなされており，大村らや石尾らはソースコードを読解する際のコードリーディング支援を行うツールを開発している [12][13]．しかし，これらの研究ではある程度プログラミングに習熟したプログラマがソースコードを読むという状況を想定しているため，コードに触れる機会を増やすための本研究とは目的が異なる．

3.2 設計指針

システムを実装するにあたり，3 つの設計指針を設けた．

1. **遊び (エンタテインメント) の要素を取り入れる**

   コードリーディングはプログラミングにおいて重要なスキルの 1 つであるが，プログラムを読解した経験の少ないプログラミング初学者にとって，淡白なプログラムをただ読み込むことは難しい．よってエンタテインメントの要素を付与してプログラムを読解させることで，コードリーディングの心理的障壁を下げることを目指した．

2. **日常的に使用するツールに組み込む**

   アスリートが継続的なトレーニングを怠らないように，ある技能を高めるには日常的な鍛錬を要する．プログラミングにおけるコードリーディングでも同様であると考え，極力日常的な使用を促すためにシステムを普段から使用するデバイスやアプリケーションから使用できるように設計した．

3. **プログラムを読むことに専念させる**

   プログラムを「書く」ことを促進するならば対象のソースコードを 1 つの言語に絞るべきという考え方もある．プログラムを書くことが目的であれば，複数の言語を同時進行させることは混乱を招き，デメリットが大きい．しかし，プログラムを「読む」ことが目的であれば，様々なプログラミング言語に触れることで，コンピュータを取り巻くプログラミング言語とそのライブラリ，エコシステムなどに触れ，プログラミングに対する興味関心を高めることが可能であると考える．またプログラミング上級者との会話のきっかけができる，将来的に適材適所で言語を使い分けることにつながる，プログラムが多様なので飽きにくいなど複数の副次的なメリットがある．従って本システムでは，1 つの言語に読解を絞った機能だけでなく，多様な言語に触れられる機能も実装する．

3.3 提案システム

3.3.1 実装機能

提案システムに実装する，エンタテインメントの要素を用いた 2 つの機能について説明する．

- **クイズ機能**

  この機能はユーザが使用した際に GitHub からランダムに取得したプログラムを表示する．プログラムの記述言語は多様であり，ユーザは表示されたプログラムを読み，どのプログラミング言語で記述されたプログラムかを回答する．正解した場合は得点が得られ，取得した累計得点の多さによるランキングが表示される．設計指針 3 で述べた多様な言語に触れられる機能に相当する．

- **占い機能**

  この機能をユーザが使用すると GitHub からランダムに得られたプログラムが表示され，ユーザはそれを読解し自分なりの解釈をすることで，その日の運勢を占う．表示されたソースコードの意味を解釈することが必要であるため，必然的にコードの読解が必要となる．この機能においては表示されるプログラムを JavaScript で記述されたものに限定した．

3.4 プロトタイプシステム

初めに，提案システムをチームコミュニケーションツールである Slack の bot として実装した．システムを利用する際は，この bot を導入しているチャンネルでコマンド

を入力することで各機能を使用できる．利用可能なコマンドを表 1 に示す．また表示するプログラムは内容が 1 つのファイルにまとまっていることが望ましいため，GitHub Gist[5] からプログラムを取得した．またクイズ機能を使用している様子を図 1 に，占い機能を使用している様子を図 2 に示す．

表 1: コマンド一覧

コマンド	説明
fortune	1 日の運勢を占うソースコードを表示
quiz	記述言語を問うクイズを出題
hint	クイズに関するヒントを表示
answer	クイズの回答を表示
score	各ユーザの累計得点を表示

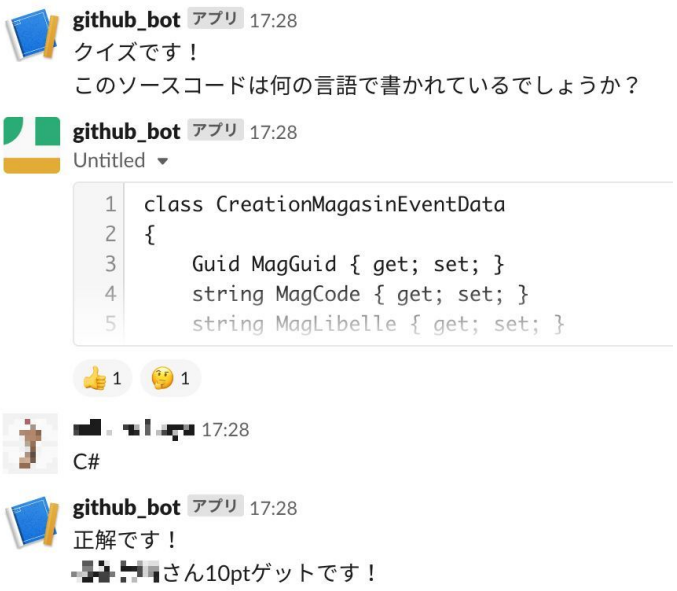

図 1: クイズ機能の様子

3.4.1 ケーススタディ

このシステムを筆者が所属する研究室で運用している Slack のチャンネルに導入し，研究室のメンバーを対象としたケーススタディを行った．研究室のメンバーは 3 名のプログラミング初学者であり，主に JavaScrtip や Google Apps Script を使用している．ケーススタディの内容としては，1 週間自由に機能を使用してもらい各コマンドの使用

図 2: 占い機能の様子

回数を調べた．またアンケートによって所感を調査した．アンケートの項目を以下に示す．

- どういう状況で機能を使用したか
- 占い機能について，楽しかったか (5 段階評価，1：楽しくなかった，5：楽しかった)
- 占い機能について，どういう点が楽しかった (楽しくなかった) か
- クイズ機能について，楽しかったか (5 段階評価，1：楽しくなかった，5：楽しかった)
- クイズ機能について，どういう点が楽しかった (楽しくなかった) か
- 機能を使用したことによって学びがあったか
- どういう学びがあったか
- コードを読むことへの抵抗は減ったと感じるか (5 段階評価，1：減らなかった，5：減った)
- 今後も機能を使いたいか
- 感想，意見, 改善点など

各コマンドの使用回数を表 2 に示す．

コマンドの使用回数に関しては，クイズ機能に関連した機能が多い結果となり，占い機能に関しては使用回数が少なかった．また期間中の最初の 3 日間ほどは頻繁にシステムが使用されていたが，期間の後半おいてはあまり活発に使用されていなかった．またどの機能も個別に使用するより，ユーザが実際に会って集まっている際にコミュニケーションを交えて使用されることが多かった．

表 2: 各コマンドの使用回数

コマンド	使用回数
fortune	9
quiz	43
hint	27
answer	3
score	33

表 3: アンケート結果

インタビュー内容	被験者 A	被験者 B	被験者 C
占い機能は楽しかったか	1	2	3
クイズ機能は楽しかったか	4	4	5
コードへの抵抗感は減ったか	3	5	5

次にアンケート結果を表 3 に示す.

まず占い機能に関してだが，使用する楽しさの評価は低い結果となった．ユーザの意見には「提示されたプログラムをどう解釈すればいいのか分からなかった」「占いという感触があまりなかった」などがあり，プログラミング初学者にとってプログラムを読解して独自の解釈を持たせることは難しく，面白みに欠けているようだった．またクイズ機能に関しては楽しさに関して高い評価が得られた．その理由として「スコアがあると、競争している感が出るところが楽しく感じた」「みんなで一緒にやっていて競争っぽくなるとき楽しかった」などの意見があり，スコアを用いたゲーミフィケーション，競争の要素がシステム使用のモチベーションを高めていたと考えられる．また「以前に出た言語の特徴に似ていると感じ、調べずに答えて正解したとき楽しかった」という意見が得られ，クイズを通したコードリーディングによってプログラミング言語の特徴を捉え，それに楽しみを感じていることが分かった．なお「機能を使用したことによって学びがあったか」という質問には全員が「あった」と回答し,「どういう学びがあったか」という質問には「言語ごとの文法の特徴が何となく分かってきた」「いろんな言語の存在を知ることができました」「今まで聞いたこともなかったプログラミング言語の名前を知り、親近感をもった」という回答があり，各プログラミング言語の特徴について理解するとともにプログラミングに対する親近感を抱かせることができていた．またプログラムへの抵抗感に関する質問でも肯定的な評価が得られ,「今後も機能を使いたいか」という質問に対して全員が「使いたい」と回答した．なおシステムの改善点に関して「プログラムのある GitHub のページへのリンクを表示して欲しい」「占いの際にラッキーアイテムなどを表示して欲しい」等の意見が得られたた

め，アンケート結果を元にプロトタイプを改善したシステムを開発した．

3.5 Web アプリケーション

プロトタイプシステムを用いたケーススタディで得られた結果を元に，プロトタイプシステムを改善し，Web アプリケーションとして再実装した．Web アプリケーションとして実装したのは Slack bot よりも開けたコミュニティでより多くのユーザの意見を取り入れるためである．このアプリケーションでは指定の URL にアクセスすることでクイズ・占いの機能を使用でき，ユーザ登録をしログインすることでクイズ正解時に得た得点の累計によるユーザのランキングを閲覧することができる．またクイズ機能においてはそのクイズに対するユーザの正答率，占い機能においては選ばれたプログラムに含まれているコメントの量，宣言された関数の数などから独自に算出した対人運，仕事運，コメントなどを表示するようにした．クイズ機能の様子を 3 に，占い機能の様子を 4 に示す．また実装したアプリケーションを Ubiquitous Wearable Workshop2019?? にて参加者に使用させ，議論を行った．占い・クイズ機能ともに使用され，表示されたプログラムを見せ合うなどシステムを介したコミュニケーションが見られたが,「クイズが難しすぎて逆にプログラムに対する抵抗感が生まれた」というコメントもあった．

図 3: クイズ機能の様子

図 4: 占い機能の様子

3.6 課題と考察

提案システムの実装と運用を通して，エンタテインメントの要素，特にクイズの要素を交えてプログラミング初学者のコードリーディングを促進することは，有効な方策であると考えられる．またその中で他者と競争するゲーミフィケーションはシステム使用のモチベーションとなり，これをプログラミング学習に取り込むことで有益な学習コンテンツを作成できると感じた．しかし，現状実装したアプリケーションには不十分な箇所が多く，より適切なゲームデザイン，ユーザがシステムを使い続けるための工夫，どのようなアルゴリズムでプログラムを選び，初学者に提示するかなど課題は多い．今後はこのシステムを通して得られた課題・観点を元に，より良いプログラミング学習コンテンツを作成していく．

4　競技性・観戦性を拡張したプログラミングゲームの開発

本項では，プログラミング初学者の興味喚起を目的とした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行った評価実験と見つかった課題，実験に関する考察について議論する．

プログラミングの重要性が高まり，プログラミング学習を始める人は増えているが，その中で挫折する者も多い．プログラミングの楽しさを感じるためには，ある程度プログラミングに習熟することが必要であることがその要因の 1 つであると考えられる．

プログラミングにおける難解な部分を隠蔽・抽象化することで，初学者でもプログラミングの楽しさを実感できることを目指したシステムはいくつかある．Scratch や Viscuit などがその代表的な例であり，これらは実際に小学校教育などに導入され，プログラミングに対する興味喚起に成功していると言える．しかしこれらは小・中学生など若い世代をターゲットとして作成されているため，高校生や大学生，あるいはそれ以上の年齢の層に対して，十分な興味喚起ができているとは言えない．また初学者向けにデザインされているため，Python や C 言語といった実践的なプログラミング言語との乖離が大きいという問題もある．

そこで本項で述べる研究では，従来の興味喚起を目的とした初学者向けプログラミング学習支援システムではカバーしきれていない層を対象としたシステムの開発を目指した．そのために

そのために本研究で提案するシステムでは，プログラミングゲームとライブコーディングの要素を取り入れた．

4.1　関連研究・関連システム

4.1.1　プログラミングを用いたエンタテインメントシステム

プログラミングとエンタテインメントを掛け合わせたコンテンツはいくつか存在する．TopCoder[15] などの競技プログラミング，コードゴルフ [16] や SECCON[17] などのハッキングコンテストが有名であり，これらはプログラマの間でも根強い人気がある．またプログラミングゲームとしては Robocode[18] が有名である．しかしこれらはある程度プログラミングに習熟したプログラマ向けのコンテンツであり，利用にはプログラミングスキルだけでなく数学やセキュリティ，コンピュータサイエンス等の知識を要するためプログラミング初学者が参加するにはハードルが高い．

4.1.2 初学者向けプログラミング学習支援システム

初学者向けプログラミング学習支援システムは数多く存在する．Scratch[21],Viscuit[22]などはその代表的な例であり，低年齢層をターゲットにした設計でブロックや絵を並べることでプログラミングでき，初学者にプログラミングの楽しさを伝えるためにデザインされている．塚本らの研究では，テキストベースのプログラミング言語による小学校でのプログラミング教育の可能性を示唆しているが，テキストベースのプログラミング言語とビジュアルプログラミング言語を用いた小学生向けの授業を比較し，ビジュアルプログラミング言語を用いた場合の方がモチベーションを向上させることができたと述べている．[19, 20] またドリトル [23] は中学校・高等学校での教育目的に使える環境を目指し，テキストベースでプログラミングさせつつも柔軟かつ小さい言語仕様により，初学者にプログラミングの楽しさを伝えることを目指している．しかし対象がやや若年層向けであり，プログラマに対する憧れを創出し，プログラミングへのモチベーションを高めるという本研究のアプローチとは異なる．

4.1.3 プログラミングゲームを用いた研究

プログラミングゲームはいくつか存在するが，その中でも有名なのが Robocode である．これは Java でロボットを制御するプログラムを記述し戦わせるゲームであり，long が Robocode コミュニティを対象に行った調査 [26] では，ユーザの多くが Robocode によりプログラミングスキルが向上したと回答し，Robocode の楽しい点としてアルゴリズムを見つけることとアーキテクチャのデザインが挙げられていた．またプログラミングゲームを盛り込むことでプログラミング学習を促進しようとした研究は多くある．Josha らはプログラミングゲームを用いて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を作成している [27]．Julian らはボクシング型の競争ゲームを用いてプログラミングスキルの向上を図っている [28]．また水口の研究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 [29]．なお増谷らの開発した VLogic[30] 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，プログラミングに対する興味喚起を行っている．これらはプログラミングの習熟度が高くなくても使用できるが，従来のプログラミングゲーム同様静的なゲーム展開であり，プログラマ同士のリアルタイムな駆け引きやアドリブといった観戦を楽しむ設計は成されていない．

4.2 設計指針

提案システムの実装にあたり，初学者の興味関心を高めるために 3 つの設計指針を設けた．

1. **観戦するにあたり，高度な専門知識を必要としない**

   初学者が提案システムでの対戦を観戦するにあたり高度な専門知識を必要としてしまっては，利用の心理的ハードルを上げかねない．極力前提知識なしに理解し，楽しめるようにデザインする必要がある．

2. **手間がかからない**

   初学者にとってプログラミング学習をする際に環境構築や普段使用しない独自ソフトウェアのインストールはモチベーションを下げかねない．今回は提案システムを JavaScript によって制御可能な Web アプリケーションとして実装し，初学者が実際にプログラミングを行わずとも見るだけで利用できるようにした．

3. **リアルタイムな駆け引き・アドリブを取り入れる**

   従来のプログラミング教育コンテンツとは異なり，スポーツなどの観戦する競技で見られるようなリアルタイムな駆け引き・アドリブの要素を提案システムに組み込むことで観戦して楽しめるようなデザインをする．

4.3 提案システム

本研究で提案するシステムでは，プログラミング初学者の興味喚起をするためにいくつかの工夫を施した．今回実装したシステムは，プログラマ同士がリアルタイムにプログラミングを行うことでキャラクタを制御し，対戦するという対人形式のプログラミングゲームである．この対戦の様子を初学者に観戦させることで，初学者のプログラミングに対する興味を高める．プログラミングゲームとして実装した理由としては，キャラクタがプログラムによって動作するというゲームの視覚的な出力が，初学者にとってプログラムの出力を理解する助けになると考えたためである．ゲームジャンルとしては，シューティングゲームの体裁をとった．これはシューティングゲームが「敵の攻撃を避けて，敵を攻撃する」というプリミティブなゲームシステムであり，見ていて展開を理解しやすいと考えたためである．またゲーム UI は LivecodeLab[24] や Hydra[25] などのビジュアルライブコーディング環境を踏襲し，ゲーム画面上にエディタを重畳することで，プログラムとその出力の双方を同時に見ることが可能なように設計した．またゲーム 3 つのフェイズに分かれており，以下で各フェイズについて説

明する．大まかなフェイズ遷移の様子を図 5 に示す．また 2 人のプレイヤが対戦する「vsPlayer」モードと 1 人のプレイヤが CPU と対戦する「vsComputer」モードを実装した．

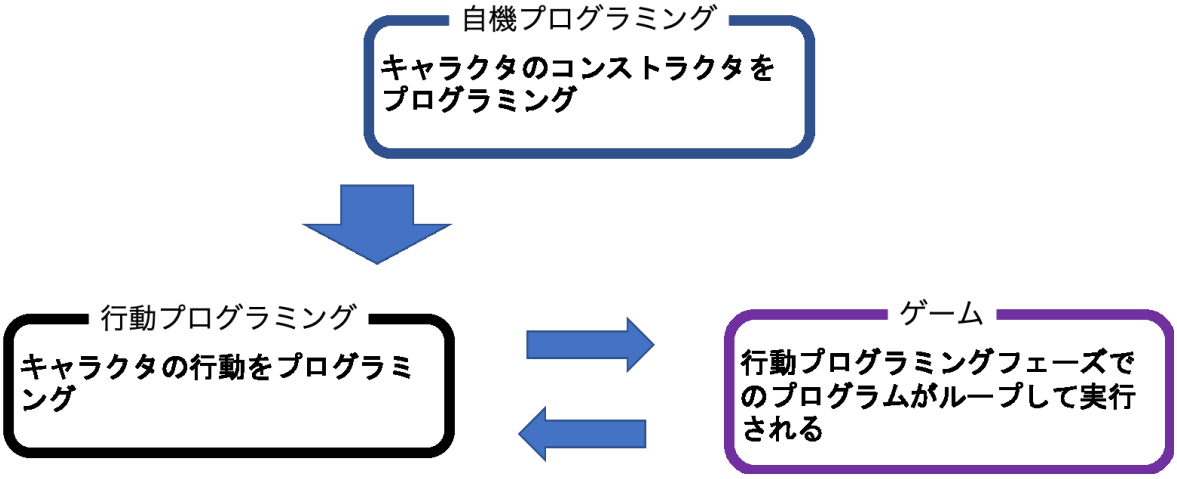

図 5: フェイズ遷移図

4.3.1 自機プログラミングフェイズ

プレイヤが「vsPlayer」モードを選んで相手プレイヤとマッチングする，または「vsComputer」モードを選ぶとこのフェイズに移行する．プレイヤには 1 か 2 の番号が振られ，番号に応じて画面の背景色や制御するキャラクタの番号が異なる．このフェイズの様子を図 6 に示す．ここでは予めエディタにプレイヤが操作するキャラクタのコンストラクタが記述されており，プレイヤはパラメータを書き換えることができる．具体的なパラメータには appearance,life,clock,power がある．appearance はキャラクタの外見であり，文字列を指定できるため，絵文字などを使ってプレイヤの好きな見た目を選ぶことが可能である．life はキャラクタの体力であり，いわゆる HP(ヒットポイント) を表している．非負の整数を指定でき，この値が 0 以下になるとプレイヤはゲームに敗北する．clock はキャラクタが行動できる回数の多さを表しており，非負の整数を指定できる．プレイヤは後述する行動プログラミングフェイズにおいて自分のキャラクタを制御するプログラムを記述し対戦するが，その際に記述したプログラムは 10 秒間ループして実行される．このループのインターバルを決めるのが clock であり，値が大きいほどインターバルは短くなる．power はキャラクタの攻撃力を表しており，これも非負の整数を指定できる．この値が大きいほど，自分が操作するキャラクタの攻撃が相手キャラクタに命中した際に削る life の値が大きくなる．双方のプレイヤが各パラメータを記述し終わると次の行動プログラミングフェイズに移行する．

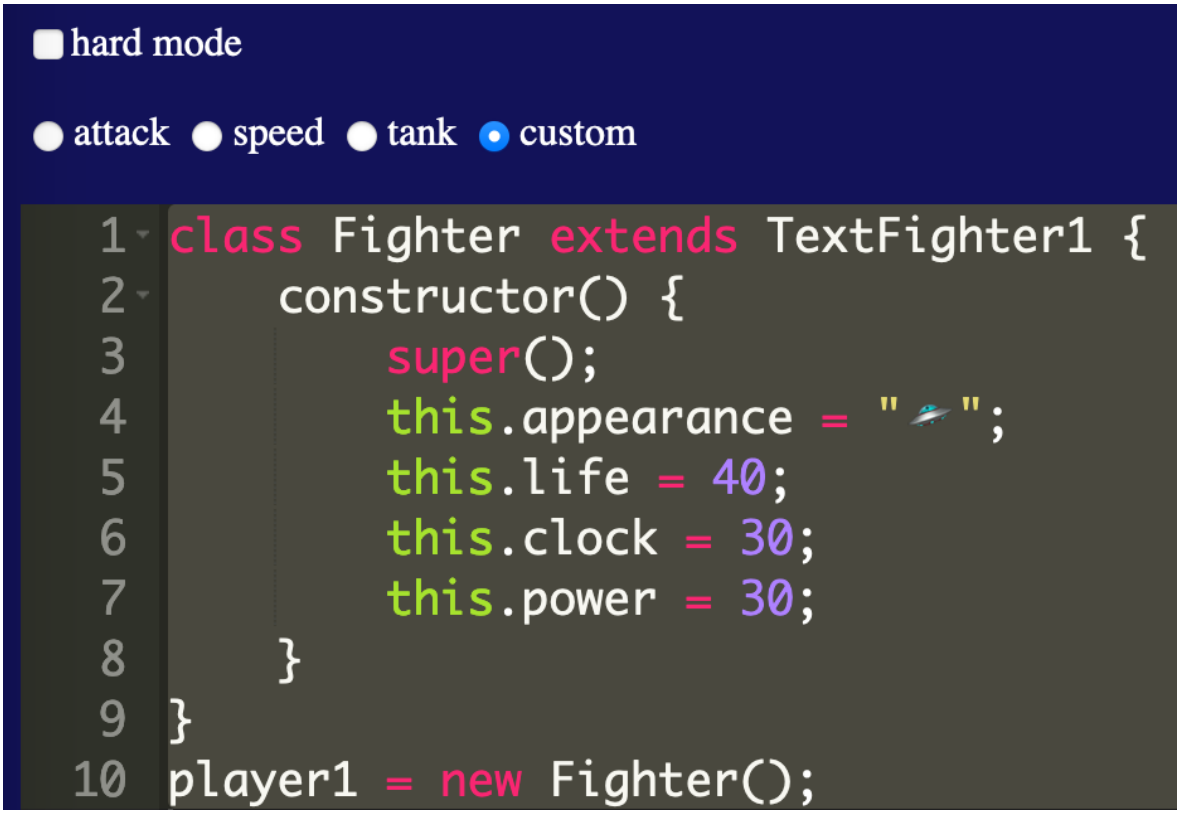

図 6: 自機プログラミングフェイズの様子

4.3.2 行動プログラミングフェイズ

このフェイズに進むと，自機プログラミングフェイズで記述したコンストラクタを元に両プレイヤが操作するキャラクタのインスタンスが作成され，ゲーム画面が表示される．このフェイズの様子を図 7 に示す．両プレイヤはプログラムをエディタに記述し，作成したキャラクタを操作する．エディタにはプレイヤに割り振られた番号に応じて「player1Loop」または「player2Loop」という名前の関数が予め用意されている．この関数が本フェイズ終了時にループして実行されることとなる．プレイヤは条件分岐や繰り返しなど従来の JavaScript の文法の他に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．用意したメソッドにはキャラクタを移動するメソッド (moveUp(), moveDown(),randomMove()) とキャラクタが攻撃を行うメソッド (shot()) などがある．またプログラム内で各キャラクタのパラメータを参照することもできる．両プレイヤがプログラムを記述し終わると次のゲームフェイズに移行する．なおこのフェイズでは相手プレイヤがどのようなプログラムを記述しているかを見ることはできない．

図 7: 行動プログラミングフェイズの様子

4.3.3 ゲームフェイズ

このフェイズでは行動プログラミングフェイ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．このフェイズの様子を図 8 に示す．このフェイズに移行した段階で，両プレイヤは相手プレイヤが記述したプログラムを閲覧することができる．このフェイズにおいて相手キャラクタを攻撃し，life の値を 0 以下にしたプレイヤの勝利となる．勝敗が決まらない場合はプログラム終了時の各パラメータを引き継いだまま行動プログラミングフェイズに戻り，再度プログラミングしゲームフェイズに移行するという過程を勝敗が決まるまで繰り返す．

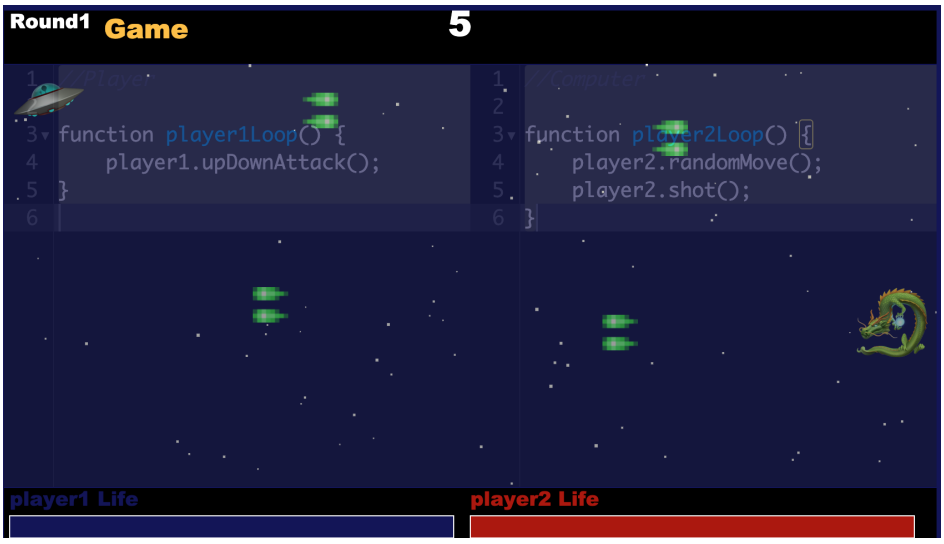

図 8: ゲームフェイズの様子

4.4 評価実験

提案システムの使用・観戦に関する感想や影響，その用法を調査するために評価実験を実施した．システムを用いて対戦するプログラマと対戦を観戦するプログラミング初学者を集め，システムでの対戦と観戦を実施し，アンケート調査と実際に対戦で使用されたプログラムのログを分析することでシステムを評価した．

4.4.1 実験参加者

実際にゲームをプレイするプログラマとしては，プログラミング (主にオブジェクト指向言語) の経験が 3 年以上ある大学院生 2 名 (男性) に声をかけた．2 名とも日常的にアクションやシューティング等のジャンルのゲームをプレイするため，システムをプレイする際にゲームに不慣れなためハンデが生まれることはないと思われる．また両者とも 3 年以上 JavaScript を使用した経験がある．またプレイを観戦するプログラミング初学者は，国際人間科学部にて開講されていた講義「プログラミング基礎演習 1」の受講者をとした．受講者のうち，実験に参加した者は 77 名であり，うち 37 名が男性，40 名が女性であった．またこの講義では JavaScript における変数，条件分岐，繰り返しなどの基礎的な文法を教えており，実験参加者は実験を行う時点でこれらを学習済であった．なお，うち 23 名は授業以前にプログラミングを学習した経験があったが，ソフトウェア開発を行ったことのある者はいなかった．またプログラマ含む実験参加者の全員が，競技プログラミングやプログラミングゲームなどのプログラミングを題材としたエンタテインメントシステムを使用した経験がなかった．

4.4.2 実験内容

初めにプログラマ 2 人にプログラミングやゲームの経験に関する簡単な事前アンケートを行った後，提案システムにある程度慣れ，用法を理解してもらう必要があるため，システムの練習をするための期間を設けた．システムの使用方法，ゲームシステム，独自に用意したメソッドなどについて説明した後，11/13 から 11/19 の 1 週間システムを自由に使用させた．また，ただシステムを使用させただけではゲームに対する理解が深まらない可能性があるため，期間中に 2 つのタスクをこなさせた．1 つは 1 人以上とシステムを使った対人戦を行うことであり，もう 1 つは相手がランダムな戦略を実行してくる対 CPU 戦において，勝率が高いと考えられるプログラムを作成することである．なお期間中はシステムに関する意見・疑問を逐次報告させ，システム使用における問題を改善した．練習期間が終わった翌日に，プログラマ同士の対戦を行った．対戦はシステムに関するプログラマ同士のコミュニケーション等の所作を観察するため，

感染症対策を徹底した環境で対面にて行った．両者の対戦時の画面を録画し，対戦後にシステムに関する事後アンケートを行った．なお練習期間中・対戦中にプログラマが記述した全てのコードのログを収集した．そして「プログラミング基礎演習 1」の最終講義で受講者に事前アンケートを行った後，初学者が観戦しやすいように対戦動画を編集したものを zoom を介して閲覧させた．またその後にシステムに関する事後アンケートを行った．

4.4.3 アンケート結構

対戦を観戦したプログラミング初学者に対する事後アンケートの内容及び結果を表 3 に示す．

表 4: 初学者に対する事後アンケート結果

Q	質問項目	評価分布					平均
		1	2	3	4	5	
Q1	ゲームを観戦するのは楽しかったか	3	6	33	27	5	3.34
Q2	ゲームの出力はプログラムを理解する助けになったか	0	9	25	30	10	3.55
Q3	観戦によってプログラミングに対する興味関心は高まったか	1	11	23	33	6	3.43
Q4	観戦によってプログラミングに関する理解が深まったか	1	23	29	17	4	3.00

まずゲーム観戦の楽しさに関して (Q1) であるが，実験参加者の多くが肯定的な回答を示した．またキャラクタが攻撃，移動するなどのゲームの出力がプログラム理解の助けになったかという質問 (Q2) に関しても 3 以上の評価が多かった．また観戦によりプログラミングに対する興味関心が高まったかという質問 (Q3) についても，3 以上の回答が多くなった．また観戦によってプログラミングに関する理解が深まったかという質問 (Q4) に関しては，やや低い評価が多い結果となった．またこれら意外にも以下の質問をした．

- Q5. このゲームを実際にプレイしたいか (プレイしたい/プレイしたくない)
- Q6. このゲームに他にどのような機能が欲しいか
- Q7. このゲームに関する意見

まずこのゲームを実際にプレイしたいかという質問 (Q5) に関しては 64.9%がプレイしたいと回答した．またこのゲームに他にどのような機能が欲しいかという質問 (Q6)

に関しては「色んな技が繰り出せるような機能」「通常攻撃以外にため技のようなものが欲しいと思った」「回復アイテム的なものや必殺技」「攻撃を受けたときの衝撃波」などの回答があり，キャラクタの行動，エフェクト，アイテムの追加などを求める意見が多く見られた．また「プログラム内容を解説する機能が欲しい」という意見も複数あり，Q4 の結果にも見られるように，プログラムの内容をより分かりやすくする工夫が必要だと考えられる．またこのゲームに関する意見 (Q7) としては,「自分のプログラミングのレベルを上げたいと思った」「面白い」「プログラミングを楽しみながら学習できると言う点でとても面白いと感じた」などゲームに対する肯定的な意見が多く得られた．しかし「プログラミング初心者には少し難しかった」「楽しめるようになるまでのハードルが高そうだった」「今持っている知識では自分では動かすことが出来ないと感じた」などゲームをプレイするのは難しそうだという反応も多く得られた．またシステムをプレイしたプログラマにも同様に事後アンケートを行った．その内容・結果を表 3 に示す．

表 5: プログラマに対する事後アンケート結果

Q	質問項目	プログラマ	
		A	B
Q1	システムをプレイするのは楽しかったか	4	5
Q2	ゲームの出力はプログラムを理解する助けになったか	4	5
Q3	プレイすることでプログラミングに対する興味関心は高まったか	3	5
Q4	プレイによってプログラミングに関する理解が深まったか	1	3

この結果についても初学者に対する事後アンケートと同様の傾向があり，Q1，Q2，Q3 については肯定的な評価が得られたが，Q4 に関してはやや評価が低い結果となった．またそれら以外にも以下の質問をした．

- Q5. 累計何時間程度システムを使用したか
- Q6. ゲームの難易度は自分にとって適切だったか
- Q7. どのような点が簡単 (難しい) と感じたか
- Q8. ゲームに他にどのような機能が欲しいか
- Q9. 今後もこのゲームを使用したいか
- Q10. ゲームに関する意見

累計何時間程度システムを使用したかという質問 (Q5) に関しては 1 名が 2 時間，1 名が 6 時間と回答した．またゲームの難易度は自分にとって適切だったかという質問 (Q6) に関しては 1 名が適切だった，1 名がやや難しかったと答えた．具体的には (Q7)「リファレンスが乏しくプログラムを書きづら買った」「現状のシステムではある程度システムを使い込んだ人をを想定した作戦を立てづらかった」と回答した．ゲームに他にどのような機能が欲しいかという質問 (Q8) に対しては,「オブジェクトが持っている値を確認できるリファレンスのようなものが欲しい」「横移動ができれば，被弾しやすいリスクと引き換えに，敵が来そうな位置で敵より先に弾を打ち込めるチャンスが増える」と回答していた．また今後もこのゲームを使用したいか (Q9) という質問には 2 名とも「使用したい」と回答した．なおゲームに関する意見は「相手のコードを読んで対処するという部分をもっと積極的にするべきだったかもしれない．ただし，相手の位置に近づくようにプログラムすれば確実に自分が先に被弾するので，自分は動かずに相手が近づくのを待つプログラム以外最良の手がないようにも思う.」というものが得られた．

4.4.4 コードログ分析結果

練習期間中・対戦中にプログラマが記述したプログラムのログ，提出したプログラムを分析することで，システムがどのように利用されているか，どのようなプログラムが記述されているかを調査した．分析手法としては各プログラムをパースし，プログラム中のキーワードを抽出した．パーサには JavaScript パーサのデファクトスタンダードである Espria[31] を用いた．また各プログラムの可読性についても分析した．その指標として循環的複雑度と SLOC(source lines of code) を用いた．循環的複雑度とはソフトウェア開発におけるソフトウェア測定法であり，プログラム中の分岐の数からプログラムの複雑さを算出する．SLOC はソースコードの行数を意味し，今回は空行やコメント行などを除いた値である論理 LOC を用いた．またこれらのパラメータの算出には JavaScript 抽象構文木のソフトウェア複雑性分析ツールである escomplex[32] を使用した．

- 対 CPU 戦におけるプログラム分析

  まず vsComputer モードで使用されたプログラムの分析結果について述べる．このモードで使用されたコードは合計 327 件である．全てのプログラムに含まれていたキーワード (演算子を除く) の個数をグラフ化したものを図 9 に示す. もっとも多いキーワードは myself というものであるがこれは自キャラクタのインスタンスを代入するための変数として使われていた．プロトタイプシステムではプレイ

ヤが player1 になるか player2 になるかがランダムに決定されていたため，それによってプログラムの内容を書き換えなくて済むように初めに myself に割り当てられたインスタンスを代入していたと思われる．次に player1 というインスタンスを参照するキーワードが多く見られ，次いでインスタンスの座標を参照するキーワードである y，条件分岐に用いられる if のキーワードが多く見られた．プログラムの内容を観察するとキャラクタの座標を参照し，その値によって処理を分岐するプログラムが多く見られ，キーワードの分析結果と一致した．また「0x」で始まるキーワードがいくつか散見されるが，これはプレイヤが難読化ツールを用いてプログラムの難読化を試みた際に現れたキーワードである．

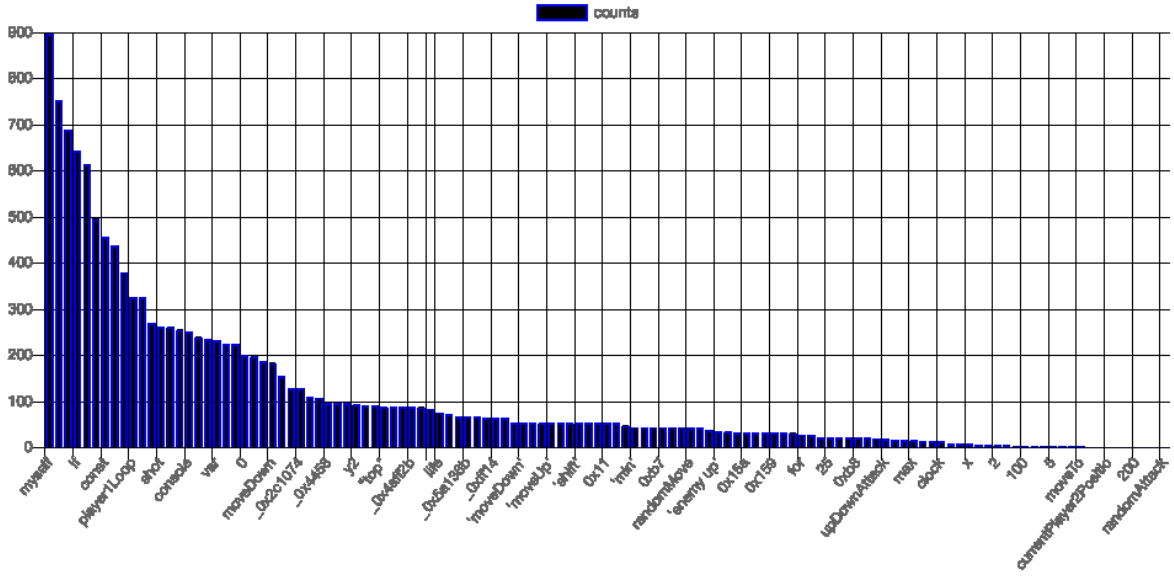

図 9: vsComputer におけるキーワード分析

次に vsComputer で投稿された各プログラムと SLOC・パラメータ数をグラフ化したものを図 10 表示する．全体として徐々に SLOC が増加しており，ゲームに習熟するほどコードの量が長くなっていったのだと考えられる．また SLOC が平行線をたどり，パラメータの数が増加している部分は難読化ツールを使用した部分である．

次に vsComputer における各プログラムの循環的複雑度を計測したグラフを図 11 示す．このグラフを観察すると循環的複雑度は 11 以下に収まっており，分岐によりプログラムの可読性を損なっていることはないと考えられる．

- 対人戦におけるプログラム分析

  次に vsPlayer において使用されたプログラムの分析結果について述べる．このモードで使用されたコードは合計 185 件である．全てのプログラムに含まれてい

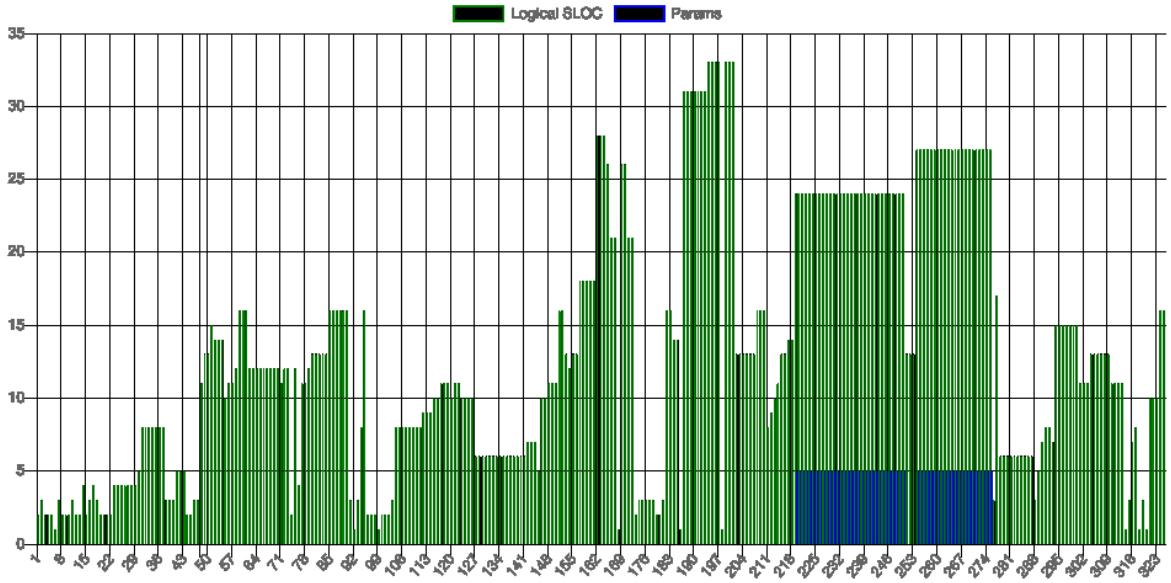

図 10: vsComputer における SLOC とパラメータ数

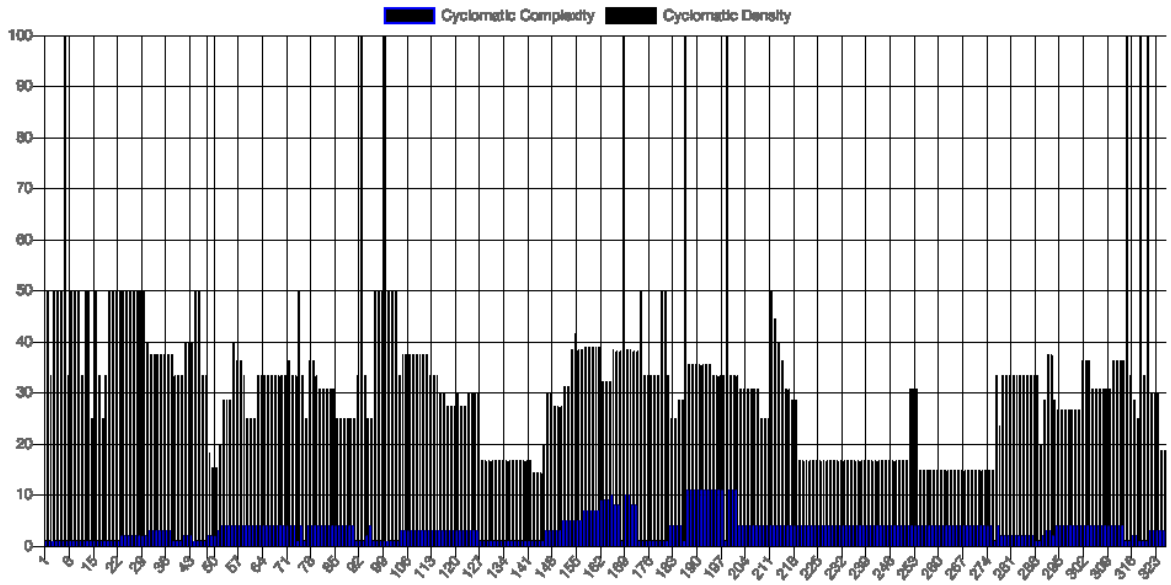

図 11: vsComputer における循環的複雑度とその密度

たキーワード (演算子を除く) の個数をグラフ化したものを図 12 に示す．vsPlayer において多かったキーワードは上から myself,if,y であり，vsComputer とプログラムの内容は大きく変わっていないと考えられる．

また各プログラムと SLOC・パラメータ数をグラフ化したものを 13 表示する．

次に vsPlayer における各プログラムの循環的複雑度を計測したグラフを図 14 示す．vsPlayer においても循環的複雑度の最大値は 11 であり，複雑な分岐はプログ

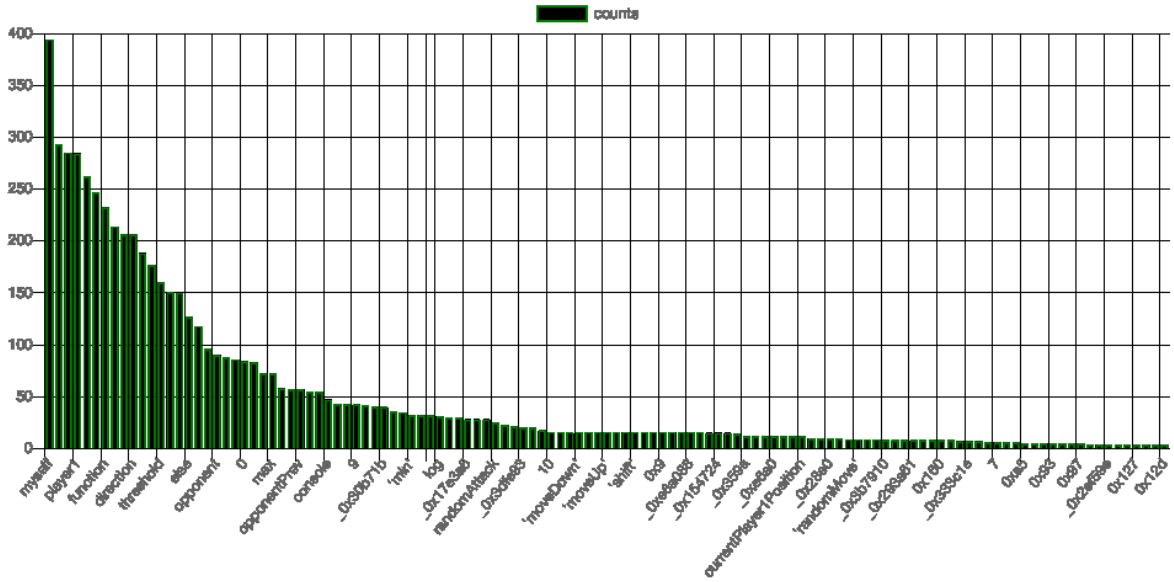

図 12: vsPlayer におけるキーワード分析

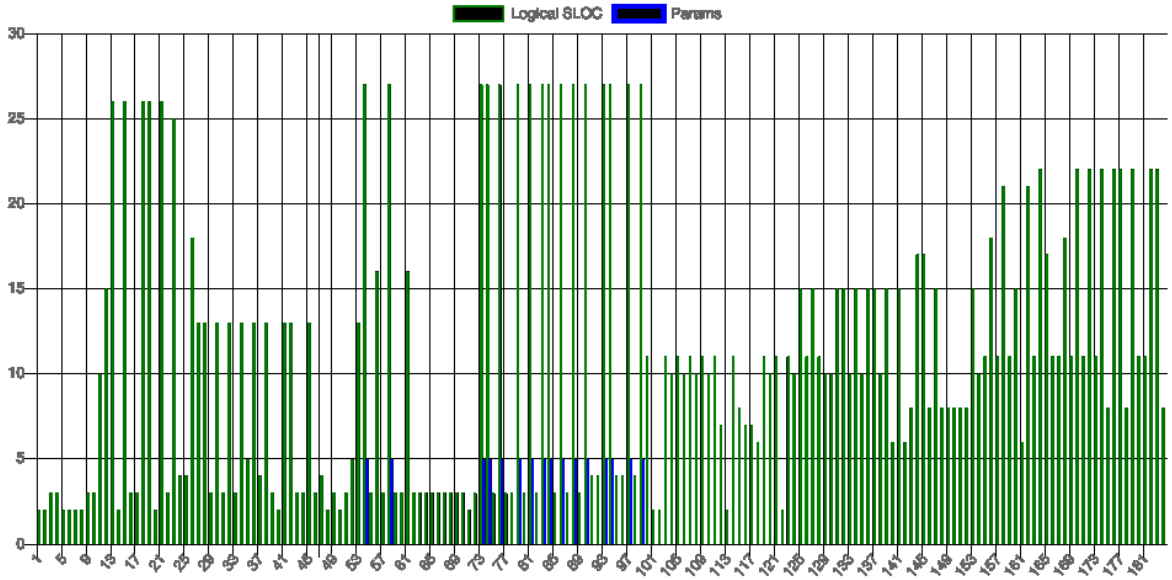

図 13: vsPlayer における SLOC とパラメータ数

ラム内に見られなかった.

- 提出プログラムの分析

  実験に参加したプログラマが提出したコードを 1,2 に示す.

ソースコード 1: プログラマ A

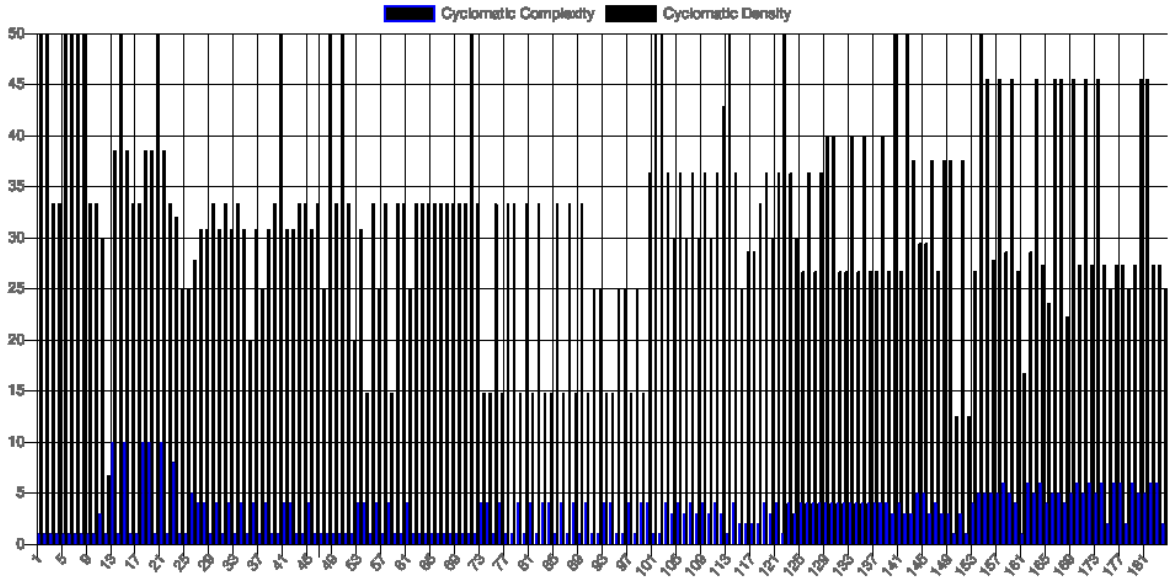

図 14: vsPlayer における循環的複雑度

```
1     const opponent = player1;
2     const me = player2;
3     let currentPlayer2Position = 240;
4     let currentMyPosition = 240;
5     function player1Loop() {
6         if (currentPlayer2Position === currentMyPosition){
7             currentMyPosition - 30 < 0 ? player1.moveDown() :
                  player1.moveUp();
8             player1.shot();
9         } else {
10            currentMyPosition < currentPlayer2Position ? player1.
                  moveDown() : player1.moveUp();
11            player1.shot();
12        }
13
14        currentPlayer2Position = player2.y;
15        currentMyPosition = player1.y;
16    }
```

プログラマ A の記述したプログラムは，相手キャラクタと自キャラクタの位置関係によって処理を分岐している．大まかな戦略としては，相手キャラクタと自キャラクタの座標を比較し，同じ位置にいれば上あるいは下に動いて攻撃し，違う位置にいれば相手キャラクタのいる方向へ移動し攻撃する，というものである．

ソースコード 2: プログラマ B

```
1     const threshold = { min: -17-25, max: 9+25 };
2     const myself = player1;
3     const enemy = player2;
4     function player1Loop() {
5         var diff = enemy.y - myself.y;
6         var direction = enemy.direction;
7         console.log(diff);
8         if (diff > threshold.min && diff <= threshold.max) {
9             if (direction === "top") {
10                myself.moveUp();
11            } else if (direction === "bottom") {
12                myself.moveDown();
13            }
14        } else {
15            if (diff < threshold.min) {
16                myself.moveUp();
17            } else if (diff > threshold.max) {
18                myself.moveDown();
19            }
20        }
21        myself.shot();
22    }
```

プログラマ B のプログラムもプログラマ A と同様，キャラクタの座標を用いた処理を行っているが，キャラクタの向きを表すパラメータである direction も使用している．自キャラクタと相手キャラクタの座標の差分が閾値の範囲内であれば，相手の direction を参照して相手と同じ進行方向に移動し攻撃，自キャラクタと相手キャラクタのが大きく異なれば，相手キャラクタから逃げて攻撃するというものである．

4.4.5 考察

評価実験に関する考察について述べる．プログラマ同士の対戦においては，行動プログラミングフェイズで設定した appearance について「かわいい」などとコメントしていた．また対戦後にお互いのプログラムの内容や今までのプログラミング経験に関するコミュニケーションをとっており，システムを使用することでプログラマ同士のコミュニケーションを促進できていたと考えられる．また対戦を観察した結果，プレイヤがどうプログラムを変更するか悩んでいる時間が多いため，リアルタイムな観戦においては観戦者が飽きないようにするための工夫が必要だと感じた．なおコードの内容・戦略がやや一辺倒になっていたため，1 つの強い戦略に収束してしまわないような工夫が必要である．各々のゲームに着目すると一方のプログラマの書いたプログラ

ムが圧倒的に他方よりも強い場合は 1 ターンで決着がついてしまう場合があった．初学者の対戦の観戦においては，ライブでプログラマ同士が対戦している状況を用意することが困難であったため今回は動画を閲覧するという状況を用意したが，動画を見ただけでは実際に人が対戦しているという感覚が希薄であり,「プログラマがプログラミングしている」場面を見せるためには更なる工夫が必要であると感じた．また今回の評価実験において，システムに対する肯定的な評価が得られたものの，どのような要素が初学者のプログラミングに対する興味に影響を与えていたのか，特に提案システムにおいて独特な要素であるリアルタイムな駆け引き・アドリブを誘発する要素の影響について調査する必要があると考えられる．

4.5 課題

提案システムの開発・評価実験で明確になった課題について議論する．大きく分けて「システムの改善（現状のプロトタイプをどう改善していくか)」と「今後の研究 (今後どのように調査を進めていくか)」について述べる．

1. **システムの改善**

   評価実験でのアンケート結果・コードログの分析結果を鑑みてより良いゲームデザインが必要であると感じた．まずパラメータバランスを調整し，1 ターンで終わるなどすぐにゲームが終わって終わないようにする．またプレイヤ同士の対戦において圧倒的に力の差がある場合でも逆転できるような要素を加えることで，プレイヤ・観戦者がより楽しめるようにする．さらにこのゲームにおける強さがプログラマのスキルに比例し，かつ観戦者にとっても可読性の高いプログラムを表示できるよう，最大コード長に制限を設けたり，循環的複雑度の最大値に制限を儲けるなどの工夫を加える．またプロトタイプシステムではエラーを画面にそのまま表示するだけだったが，プレイヤを積極的にデバッグに向かわせる工夫を加える．これらの工夫に伴い UI を設計し直すとともにキャラクタのエフェクトも観戦に適切なものを採用する．

2. **今後の研究**

   今後はシステムに改善を加えつつもより多くのユーザを対象とし，ワークショップとシステム利用による効果の調査を繰り返す．習熟したプログラマだけでなく，プログラミング初学者にも提案システムのゲームをプレイさせ，そのプログラミングへの影響を調査する．特に本システムのリアルタイム性・アドリブによる効果を重点的に調査する．

5　まとめ

本論文では，プログラミング初学者のプログラミング学習を促進するための 2 つのアプリケーションを提案した．

1 つはクイズ，占いといったエンタテインメントを交えて GitHub にあるソースコードを読解することでコードリーディングを促進するものであり，筆者の所属する研究室でケーススタディを行い，得られた問題点を改善したものを Web アプリケーションとして実装し直し，UWW2019 にて議論を行った．結果肯定的な反応が得られ，プログラミング言語への理解が深まったという意見が得られたり，プログラミングにまつわるコミュニケーションを促進できている様子が確認されたが，コードリーディングに対する抵抗感が強くなったという意見もあり，表示プログラムの選択アルゴリズムや継続的な利用を促す工夫など，改善点がいくつか見られた．

2 つ目はライブコーディングの要素を取り入れた対人形式のプログラミングゲームを初学者に観戦させることで初学者のプログラミングに対する興味関心を高めることを目指した Web アプリケーションであり，初学者を対象に評価実験を行った結果，肯定的な評価が得られたがゲームシステム・デザインに改善すべき点が見つかった．

今後はこの 2 つのシステムを通して得られた結果・意見をもとに，Scratch や Viscuit などを用いてもプログラミングに対する興味を持てない層，特に高校生・大学生のプログラミング初学者を対象にプログラミングに楽しく取り組める学習コンテンツを作成し，システムの評価を行う．さらにアンケート調査に止まらない興味推定手法を検討し，延いてはプログラミング学習コンテンツ作成・プログラミングに関する講義のデザインにおけるガイドラインの作成を目指す．

謝辞

本研究を行うにあたり，日頃より御指導，御激励を賜り，数々の御教示を頂きました西田健志准教授に深甚なる謝恩の意を表します．また神戸大学大学院国際文化学研究科に在学中，御教示，御激励頂いた神戸大学大学院国際文化学研究科の諸先生方に感謝すると共に，諸職員の方々に感謝いたします．日頃より数々の御助言を下さいました諸先輩方，快適な環境を作って頂いた研究室の皆様方，実験に協力頂いた実験参加者の方々に深く感謝いたします．特に研究活動に対する多くのアドバイスとサポートを頂いた工学研究科の清水友順氏，国際文化学研究科の三嶋哲也氏に深く感謝いたします．

参考文献

[1] 諸外国におけるプログラミング教育に関する調査研究, https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/__icsFiles/afieldfile/2018/08/10/programming_syogaikoku_houkokusyo.pdf.

[2] 小学校プログラミング教育の手引 (第三版), https://www.mext.go.jp/content/20200218-mxt_jogai02-100003171_002.pdf.

[3] 松本 絵里子, 佐藤良樹, 佐藤瑞帆ほか: C 言語の概念と実行過程を可視化するプログラミング学習用アプリケーションの開発, 専修ネットワーク＆インフォメーション, No.24, pp.15-26 (2016).

[4] Progate, https://prog-8.com/.

[5] GitHub Gist, https://gist.github.com/.

[6] 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).

[7] E.Guzman et al: Sentiment analysis of commit comments in GitHub: an empirical study, In Proceedings of the 11th Working Conference on Mining Software Repositories, 2014, pp. 352–355.

[8] 永野真知, 早瀬康裕, 駒水孝裕, 北川博之: GitHub と StackOverflow におけるユーザ行動の統一的な分析, 情報処理学会第 79 回全国大会, pp. 363–364 (Mar. 2017).

[9] 柴藤大介, 有薗拓也, 宮崎章太, 矢谷浩司: CodeGlass:GitHub のプルリクエストを活用したコード断片のインタラクティブな調査支援システム, 情報処理学会インタラクション, vol.2019, pp.159–16 (Mar. 2019).

[10] 一ノ瀬智浩, 畑秀明, 松本健一: ソースコード上の技術的負債除去を活性化させるゲーミフィケーション環境の開発, 情報処理学会関西支部支部大会講演論文集, vol.2016, (Sept. 2016).

[11] 樋川一幸, 松田滉平, 中村聡史: コミュニケーションチャネルに入り込む研究室実験 BOT の提案と運用, 情報処理学会研究報告グループウェアとネットワークサービス,vol.3, pp. 1–7(Mar. 2019).

[12] 大村裕, 渡部卓雄: プログラム理解のためのコードリーディング支援ツールの提案と実装, 日本ソフトウェア科学会講演論文集, vol.31, pp.443–446, (Sep. 2014).

[13] 石尾隆, 田中昌弘, 井上克郎: ソースコード上での情報タグ伝播によるコードリーディング支援, ウィンターワークショップ論文集, Vol.2008, pp.31–32, (Jan. 2008).

[14] Ubiquitous Wearable Workshop, http://cse.eedept.kobe-u.ac.jp/uww2019/.

[15] Topcoder, https://www.topcoder.com/.

[16] 浜地慎一郎: Code Golf, http://shinh.skr.jp/dat_dir/golf_prosym.pdf.

[17] SECCON, https://www.seccon.jp.

[18] Robocode, https://robocode.sourceforge.io/.

[19] H.Tsukamoto et al: Programming education for primary school children using a textual programming language, In 2015 IEEE Frontiers in Education Conference (FIE), pp. 1–7.

[20] H.Tsukamoto et al.: Textual vs. visual programming languages in programming education for primary schoolchildren, 2016 IEEE Frontiers in Education Conference (FIE), IEEE, 2016. p. 1–7.

[21] M.Resnick et al: Scratch: programming for all, Communications of the ACM, 52(11), pp. 60–67.

[22] 原田康徳: 子供向けビジュアル言語 Viscuit とそのインタフェース, 情報処理学会研究報告ヒューマンコンピュータインタラクション (HCI), 2005, pp.41–48.

[23] S.Kanemune et al: Dolittle: an object-oriented language for K12 education, EuroLogo, 2005, pp. 144–153.

[24] LivecodeLab, https://livecodelab.net/.

[25] Hydra, https://hydra.ojack.xyz/.

[26] L.Ju: Just For Fun: using programming games in software programming training and education, Journal of Information Technology Education: Research, 2007, pp. 279–290.

[27] 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).

[28] J.Moreno: Digital competition game to improve programming skills, Journal of Educational Technology & Society, 15(3), pp. 288–297.

[29] 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).

[30] 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).

[31] Esprima, `https://esprima.org/`.

[32] escomplex, `https://github.com/escomplex/escomplex`.