情報処理学会論文誌　デジタルプラクティス　Vol.4　No.2　36–45 (Apr. 2023)

対戦型ゲームを利用したソフトウェア開発演習を通したプログラミングの動機づけの変化

國近 秀信[1,a)]　赤川 啓行[2]

受付日 2022年8月15日, 採録日 2023年1月19日

概要：一般的なプログラミング教育では，まず講義により知識を獲得し，その後演習を通して獲得した知識の定着を図るという方法がとられている．この方法は効率的である反面，ソフトウェア開発のサイクル全体を経験することは難しく，また，動機づけの点で問題がある．そこで本研究では，対戦型ゲームを利用し，実践的な経験を積みつつ，プログラミングに対する動機づけを高めることができるソフトウェア開発環境を実現した．評価として，プログラミングの講義を受講しているプログラミング初学者に本システムを使用させたところ，ソフトウェア開発のサイクルを繰り返すことにより，プログラミングについての動機づけとして，効力期待について有意な向上がみられた．

キーワード：ソフトウェア開発演習，プログラミング，動機づけ，タイルスクリプティング，対戦型ゲーム

Changes in Motivation of Programming through Software Development Exercises using a Shooting Game

Hidenobu Kunichika[1,a)]　Hiroyuki Akagawa[2]

Received: August 15, 2022, Accepted: January 19, 2023

Abstract: In general programming education, students are first taught knowledge through lectures and then acquire the knowledge through exercises. While this method is efficient, it is difficult for students to experience the entire software development cycle, and there are problems in terms of motivation. Therefore, we have implemented a softwtare development environment using a shooting game in which students can experience the entire software development cycle and increase their motivation for programming. For evaluating the system, we gave the system to novice learners in programming. As the result of the evaluation, we found that repeating the software development cycle by using our system significantly increased efficacy expectations as a motivation for programming.

Keywords: softwtare development exercises, programming, motivation, tile scripting, shooting game

1. はじめに

一般的なプログラミング教育では，まず講義により文法を中心とした知識を獲得し，その後演習を通して獲得した知識の定着を図るという方法がとられている [1]．ここで，講義内の演習で行う問題解決は，その主目的である知識の定着をより短期間で効率よく行うための簡略化が行われている．たとえば，解決の対象となる問題は，演習課題として教員から与えられる物であり，実社会での問題解決のように問題を自ら発見・選択する必要がない．また，検証作業についても同様に，実装したプログラムの動作結果を読み解くまでもなく，あらかじめ与えられた明確な目標状態と照らし合わすのみである．

この手法は，時間的，場所的に限られた条件下であっても，学習者に対し効率よく情報を伝達できるよう配慮されている一方で，2 つの問題点を持つ．1 つは，解決する問題を自ら発見し，解決策を導出し，コンピュータが実行可能な形式で表現し，問題が解決できるか否かを検証するソフトウェア開発に対応した実践的な問題解決のサイクルを

[1] 九州工業大学大学院情報工学研究院
Faculy of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka, Fukuoka, 820–8502, Japan
[2] 九州工業大学大学院情報工学府
Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka, Fukuoka, 820–8502, Japan
[a)] kunitika@ai.kyutech.ac.jp

経験することが難しいという点である．2 つ目は，課題の設定や解決法などについて選択の幅が少ない場合が多く，動機づけにおいて重要な内発的動機づけにつながる要素が限られている点である．

1 つ目の実践的なサイクルの経験不足に関しては，学習者に対しより実践的な問題解決の経験を提供する先行研究が複数存在し，その多くが自身の力である程度プログラムを書くことができる学習者を主な対象としている．たとえば，尾崎 [2] は，プログラミング教育の最終目標は手段としてプログラミングを活かす問題解決能力であるとし，問題解決能力を獲得させる手法として，ボードゲーム戦略を題材とした問題解決型のプログラミング演習を採用している．演習は各々が実装したボードゲームのルーチンを競わせる形で行われ，競争要素とゲームという題材そのものによるモチベーションの維持を図りつつ，学習者が自主的に問題解決に取り組むことができる．この研究では，学習者に対し，最低限必要な前提知識として，ボードゲームのルーチンを実装可能なレベルの Java 言語に関する理解および周辺知識を要求している．

また，プログラミングに不慣れな初学者を対象にした教育支援については，Scratch [3] など，プログラミングに必要とされる諸要素を視覚化，簡単化もしくは隠蔽することで，初学者に高度な専門知識を要求しない実装手段を提供する先行研究が多く存在する [4]-[6]．これらは多くの場合，初学者に対する配慮から問題設定や問題解決のプロセスに対する単純化が行われており，一般のプログラミング教育での演習と同様の理由から，実践経験を積む手段としては不足がある．

ソフトウェア開発に目を向けると，Project-Based Learning（PBL）として演習が実施されている [7]-[9]．PBL では，一通りプログラムを書くことができる参加者がチームを組み，作業を分担して協力しながら演習を進めることにより，具体的な問題を解決する．チームで実践的なプロセスを遂行するため，コミュニケーション能力や自分で課題を発見する力，学生間の能力差を解決するための力などが養われると考えられる [9]．一方で，分担しての作業となるため，作業量や負担にばらつきが生じる点や各自が一通りのプロセスを経験することは難しい点が問題点として考えられる．また，一通りプログラムが作成できることを前提としているため，プログラミングの初学者を対象とすることは難しい．

2 つ目の問題点である動機づけについては，外からの報酬や要請により行動が生じる状態である外発的動機づけよりも，完全に自律的で自分の興味から行動が生じる状態である内発的動機づけのほうが好ましく，深く，持続する学習を導くと考えられる．また，行動の自律性を高め，内発的動機づけをはぐくむのに有効な手段は，自己選択の機会を与えることだといわれている [10]．動機づけに配慮したプログラミング学習支援としては，松本らによるプログラミングゲームの活用がある [11]．当該研究では，プログラミング導入科目の前段階で，学習意欲を高めるとともにプログラミングに必要な概念を理解させることを目的としてプログラミングゲームが活用されており，プログラミングゲームの理解度とプログラミング学習後の達成度テストとを比較し，それらの得点の間に正の関係が示唆されたことが報告されている．このプログラミングゲームは，学習者の興味を引き，学習意欲を高めるために有効であったと判断できるが，課題は提供され，自己選択の機会は限られているため，内発的動機づけに対する効果は限定的であると考えられる．

PBL では，問題発見，つまり，解決すべき課題や作成するシステムの自由度が高い場合 [7], [8] も低い場合 [9] も存在する．特に自由度が高い場合は，課題の決定に関する自己選択の機会が与えられるため，内発的動機づけに対する効果が期待できると考えられる．しかしながら，チームとしての選択となるため，選択への各自の関与の度合いを考えると，その効果にはばらつきが生じる可能性がある．また，各自の能力差により，自分の知識やスキルに合った作業をすることができない可能性があり，そのような場合は動機づけの低下につながると考えられる．

以上を踏まえ，本研究では，初学者を対象とし，プログラミングの上流工程であるソフトウェア開発に対応した実践的な経験を積みつつ，プログラミングに対する動機づけを高めることができるソフトウェア開発環境の実現を目的とする．より具体的には，学習者は，システム内のゲームにおいて自ら問題を発見し，要件分析，設計，実装，テスト検証というサイクルを繰り返すことにより問題解決を目指す．本システムは，一人での作業を前提としており，チームによる作業は対象外となるため，その点に関連する経験はできないものの，プログラミングの初学者の段階で，一通りのソフトウェア開発プロセスを実践可能である．情報処理技術者になるための教育の初期段階において，将来のソフトウェア開発プロセスを経験させつつ，プログラミングの動機づけ向上を目指す．本システムは，問題発見の自由度が高く，自らの能力に合った物を自己選択する機会を与えるという点で，内発的動機づけの育成に寄与すると考える．ゲームという与えられた環境下での活動ではあるが，自律的に自ら選択しながら，従来の演習よりも実践的なプロセスを繰り返すため，成功体験による相乗効果もあり，動機づけが高まることが期待される．

2. 支援方法

本研究では，実社会における問題解決を模した一連のプロセスとして，一般のソフトウェア開発 [12] で行われる要件分析，設計，実装，テスト検証に，問題発見過程を加える．これは，問題発見と要件分析は密接に関連してお

り，PBL 形式のソフトウェア開発演習でも問題発見に相当する「課題の設定」が含まれることが多い [7], [8] ためである．また，動機づけを高める観点から，自己選択の機会を増やすためにも有用であると考える．まず問題発見では，あらかじめ用意された問題が与えられるのではなく，与えられた環境下において，自ら解決すべき問題を発見する．次に要件分析においては，一般の演習課題のように問題を取り巻く環境が明示され与えられているわけではないため，自ら問題設定や環境等を理解し，詳細な仕様を決める必要がある．続いて設計においては，発見した問題を解決するための具体的な方策を考案する．実装では，設計に沿ってコーディングを行う．最後にテスト検証では，実装したプログラムによって，問題が解決されることを確認する．このとき，一般的な演習のように明確なゴールは与えられないため，検証方法を自分自身で考える必要がある．

前述のとおり，本研究では，初学者を対象とし，より実践的な経験を積ませることにより，プログラミングに対する動機づけを高めることを図る．本研究では，プログラミングの初学者を対象にすることから，プログラミング言語に対する理解や周辺知識の不足が原因で，問題解決が困難になる恐れがある．よって，前提知識の不足による影響を緩和することが求められる．そのため，実装手段としてタイルスクリプティングを導入し，実装プロセスに最低限必要とされる前提知識を引き下げ，プログラミング言語に対する理解やその他周辺知識に乏しい初学者であっても実践に取り組むことができるようにする．また，問題解決の過程で初学者がモチベーションを減退させる要因の１つに，問題の難度や実装手段が要求する知識量が，自身の知識量と比較してあまりに多く，解決の糸口すらつかめない状況に陥ってしまうことが挙げられる．そこで本研究では，問題の難易度や自身の習熟度に応じた実装手段を選択できるような機構を用意し，知識不足を原因としたモチベーションの減退を防ぐ．

より実践的な経験という点では，従来の演習で不足している問題発見や検証のプロセスを経験させることが重要となる．一般的には，曖昧かつ不確定要素の多い環境下で問題発見が行われると考え，既存の手法では省略されていた曖昧な環境からの問題の切り出しや単純化，解決に必要な情報の取捨選択や，それらを行うための問題設定に対する理解が必要な環境が望ましいと考える．また，動機づけの観点からも望ましいと考え，本研究では，対戦型ゲーム環境でのプログラミングを採用する．ゲーム内の環境は，一般的な演習で与えられる問題設定と比較して，曖昧かつ不確定要素が多いと考えられる．そのような状況下での問題発見のプロセスでは，自身が解決したいと考える箇所を分析し，自身の能力で解決可能なレベルの問題に切り出すという従来にない工程が必要となる．同様に，検証のプロセスについても，あらかじめ決められた入出力が与えられるなどの簡単化が行われていないため，何を持って問題が解決されたとするのかを自ら考え，考案した条件に従って検証作業を行う必要が新たに発生する．加えて，知識定着を目的とした演習において，出題された問題を解決するための方策は，基本的に直前に学んだ内容となるよう単純化が行われているが，ゲーム内において自ら発見した問題にはそのような単純化が行われていないため，解決につながると思われる方策を列挙したうえで取捨選択および試行錯誤を行う必要がある．

動機づけの点では，より現実的な問題設定になることで，従来と比較して，問題解決がより困難かつ長期にわたる物になると考えられる．その結果，解決の過程で学習者のモチベーションが徐々に減退し，途中で問題解決を諦めてしまう恐れがある．本研究では，問題解決プロセスをゲーム環境で繰り返すという点，および，自分自身の習熟度に応じた実装手段を選択できるため，知識不足を原因とした動機づけの低下を抑制するという点で，動機づけの維持・向上に寄与すると考える．また，動機づけには自己選択の機会が重要となるが，一連の問題解決プロセスすべてにおいて自己選択の機会が生じることとなる．特に問題発見や戦略・戦術に関連する設計過程においては，一般に多くの可能性があり，その中から学習者自身の基準によって選択することとなる．学習者は，自分自身の習熟度に応じて設計・実装し，問題解決プロセスを繰り返すことで，動機づけが高まることが期待される．

3. ソフトウェア開発環境

3.1 必要な機能

前述のとおり，本研究では，対戦型ゲーム環境において，問題発見，要件分析，設計，実装，および，テスト検証を繰り返すことにより，問題解決の試行錯誤を行う．この環境を実現するためには，以下の機能が必要である．

- 対戦型ゲーム
  対戦型ゲームは，問題発見とテスト検証の場となるとともに，要件分析や設計にも関連が深い機能である．ここでは，学習者が主体的に問題解決プロセスを遂行できるようにする必要がある．また，自己選択の機会を提供するため，複数の解決すべき課題やその解決手段が存在する状況を提示する必要がある．
- スクリプトエディタ
  前述のとおり，本研究では，プログラミングの初学者を対象にすることから，プログラミング言語に対する理解や周辺知識の不足が原因で，問題解決が困難になる恐れがある．この問題の解決法の１つとして，タイルスクリプティングを導入する．そのプログラミング環境として，スクリプトエディタが必要となる．スクリプトエディタとしては，初学者が容易にプログラミング可能な操作性と，問題解決に必要なルーチンの実

装手段を提供する必要がある．さらに，一般的な計算機言語への橋渡しのため，一般的な計算機言語との連携について配慮することが望ましい．

3.2　概要

本システムの概要を**図 1** に示す．本システムは，対戦型ゲームおよびスクリプトエディタから構成される．また，開発用プログラミング言語として Java を利用した．まず学習者は，問題解決の起点としてゲームを行い，ゲーム内で自身が解決する問題を発見する．次に，要件分析および設計として，問題解決のための具体的な戦術と，それを実現するためのオブジェクトの動作を考案する．その後，スクリプトエディタを用い，自身の操作キャラクターに設定可能なルーチンとして，考案した戦術を実装する．最後に，テスト検証過程として，実装したルーチンの動作を実際にゲーム内で確認し，問題が解決できたか否かを判断する．もし，ルーチンが期待どおりの動作をしなければ，問題解決の各プロセスを繰り返し，試行錯誤を行う．なお，テスト検証過程において，新たな問題が発見される場合も考えられる．

3.3　対戦型ゲーム

本研究では，問題発見およびテスト検証の場として，対戦型ゲームを提供する．プレイヤーは最大 3 体の操作可能なキャラクター（以降，ユニットと呼ぶ）を持つことができ，ゲーム内においてプレイヤーは自身のユニットを駆使し，いかに自身のユニットを守りつつ，対戦相手のユニットを倒すことができるかを競い合う．本ゲームでは，マウスやキーボードを用いて手動で複数のユニットを操作するだけではなく，一部またはすべてのユニットの動作をプログラミング（作成したプログラムをルーチンと呼ぶ）できるようになっている．

対戦型ゲームの処理の概要は，次のとおりである．ルーチンランタイムは，ユニットのルーチンを実行する役割を持つ．ゲーム開始時に設定された各ユニットのルーチンは 1/60 秒（以降，1 フレームと呼ぶ）ごとに呼び出され，プレイヤーに代わり状況判断およびユニットに対する操作を行い，ゲームフィールド（以下，フィールド）上の物体の情報を更新する．最後に，各フレームの終了時に，フィールドレンダラがフィールド内の状況を視認可能な形で描画する．

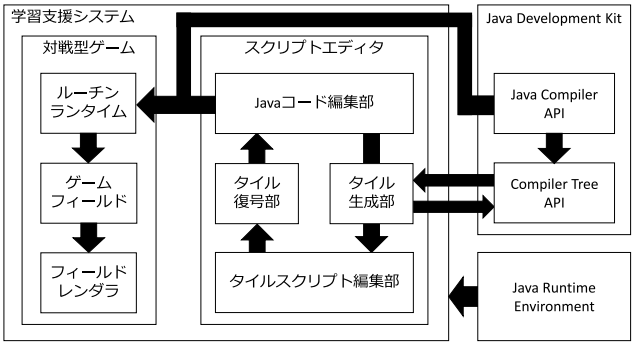

図 1　システム構成図
Fig. 1　The outline of the system.

本ゲーム環境では，学習者による問題発見を促進する必要がある．また，その問題および解決方法は，学習者が自己選択可能なように，複数存在することが必要である．前者の問題発見の促進については，手動による操作のみでは勝利することが困難な敵を最初から用意することで実現する．そのため，学習者は自然と手動操作を困難にしている原因やユニットに行ってほしい理想的な動作について考えるようになり，問題発見およびその解決策となる戦術の考案を促すと考える．

後者の問題と解決方法の選択機会の提供については，ユニットとその特徴に多様性を持たせることで実現する．より具体的には，有利／不利の関係が三竦みを形成するように攻撃と防御のスキルが設計された 3 種類のユニットを用意している．学習者は，それらの中から 3 体のユニットを選択してチームを編成することが求められ，問題や解決方法の多様性が生じることとなる．

本ゲームで利用可能なユニットは，以下の 3 種類である．

- Knight：機動力と近距離戦に優れる一方で，敵の攻撃に対する耐久力は低い．
- Bishop：遠距離戦を得意としつつ味方ユニットの回復が可能である一方で，耐久力は最も低い．
- Rook：敵ユニットの行動阻害を得意とし，耐久力は高いものの，機動力は低く，射程距離は短いという欠点を持つ．

各ユニットはそれぞれを特徴付ける 4 つのスキル（攻撃や防御などの移動以外の行動）を持つ．各スキルは，その威力が異なるように設計されており，その威力に応じて，連続使用を制限するクールタイムが設定されている．このように各ユニットが異なる特徴をもつ複数のスキルを持ち，各クールタイムも異なるよう設定しているため，戦術のバリエーションの増加に寄与している．

図 2 に，対戦ゲームの画面例を示す．ユニット（同図 (1)）は，種類に応じたシンボルと，その HP[*1] 残量を示す円で表現され，画面内のフィールド（同図 (2)）に配置される．フィールド内の (3) は弾を表しており，ユニットと同様に，その色により敵か味方かを区別する．味方のチームは，ユニットセレクタ（同図 (4)）に表示されており，その中の 1 つを選択することで，操作可能なユニットとして (5) に表示される．(6) は，スキルインジケータであり，

[*1] ヒットポイントの略．HP が 0 となるとユニットは行動不能となる．

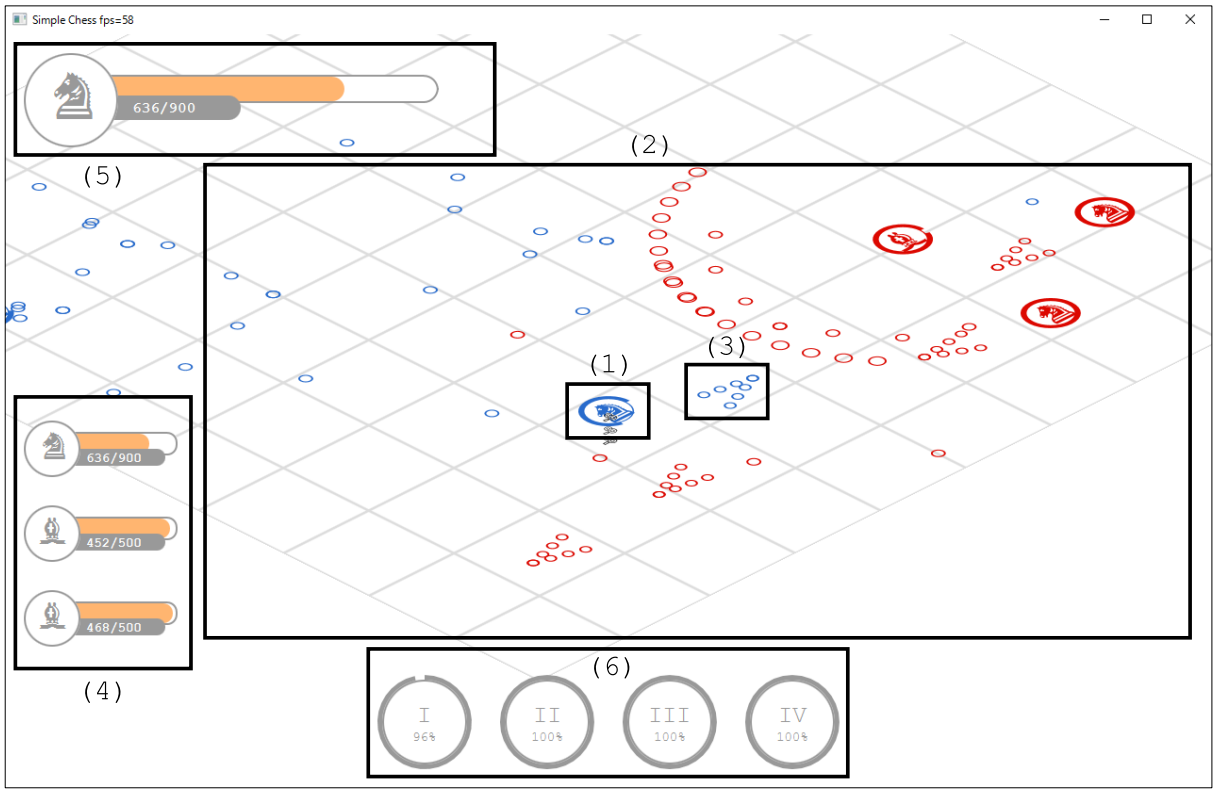

図 2　対戦型ゲームの画面例
Fig. 2　A snapshot of the shooting game.

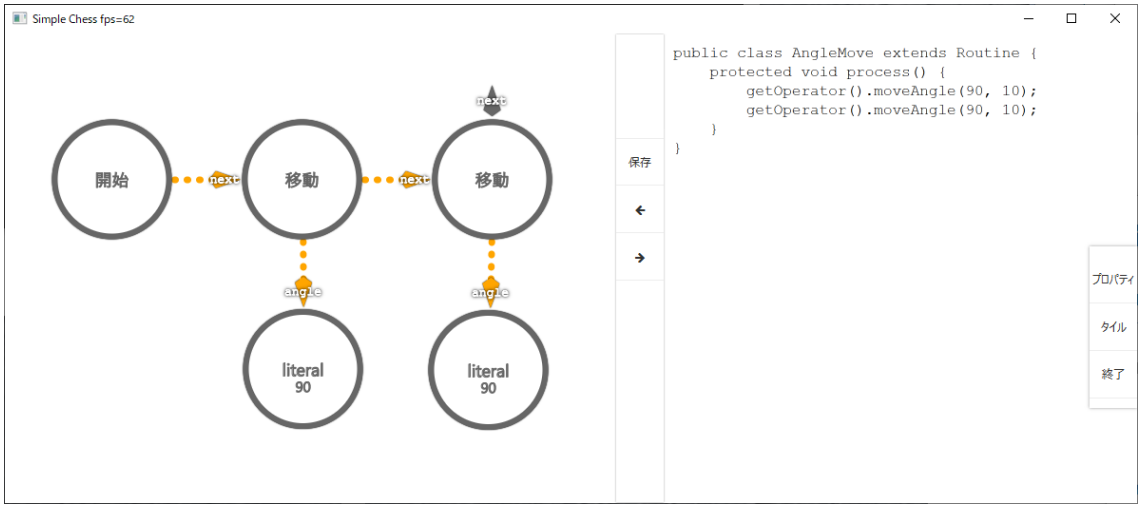

図 3　スクリプトエディタの画面例
Fig. 3　A snapshot of the script editor.

スキルを再使用するためのクールタイムを表す．

3.4　スクリプトエディタ

プログラミング言語の扱いに不慣れな初学者であっても，容易にルーチンを実装することができるようにするため，タイル形式でのルーチン作成が可能なエディタを具備する．学習者は，タイルスクリプト編集部にタイルを配置し，コネクタを用いてタイル同士を接続することにより，ルーチンを作成する．スクリプトエディタの編集画面例を**図 3** に示す．本スクリプトエディタは，左側のタイルスクリプト編集部および右側の Java コード編集部からなり，タイルスクリプト言語と Java 言語から学習者自身に適した実装手段を選択することが可能である．

タイルには，Java 言語の構文に対応するものと，定数回の繰り返しや自身に最も近い敵ユニットの取得などよく使われる処理に対応するものがある．また，前提知識が不足した初学者であっても専門的な知識を要求されることなくルーチンを実装できるようにしつつ，学習進度や問題の

難度に応じて徐々に複雑なことが実現できるようにするため，本研究では，タイルを階層的に用意している．基本的には，上層のタイルほど抽象的で学習者が考案する戦術に近く，高度な知識を要求されることなくルーチンを実装することができる．このようなタイルの階層構造を活用することで，発見した問題の難度や学習者の習熟度が向上するのに伴い，徐々により下層のタイルを用いた実装へと移行することが可能になり，自分のペースでより発展的な手段へと取り組んでいくことができる．

例として，ユニットを移動させる場合に用いるタイルのうち，上層の「移動（angle）タイル」と，下層の「移動（angle，distance）タイル」を用いて説明する．移動（angle，distance）タイルは，angle および distance の 2 つの引数を持つのに対し，移動（angle）タイルは distance が 10 に固定されており angle のみが指定可能である．移動（angle）タイルは引数が少ないためより簡単に利用することができ，さらに細かい制御が必要となった場合に，より下層の移動（angle，distance）タイルを利用する．

前述のとおり，本システムでは，直接 Java 言語によりルーチンを記述することも可能である．初期段階ではタイルスクリプティングによるルーチン作成を想定しているが，学習の進度に応じて実装手段を使い分けることができる．また，2 つの手段による記述は相互に変換可能であり，対応関係を理解しやすいよう配慮されている．本機能により，本システムのタイルスクリプティングの習得に限定されることなく，一般的なプログラミング言語である Java 言語の学習への橋渡しとなることが期待できる．

図 4 に，タイルスクリプトの例を示す．このルーチンは，選択されていないユニットに自動的に敵を攻撃させるためのものであり，「当該ユニットに最も近い敵に対して，スキル 1（威力は低いものの短時間で連続使用が可能）を使用する」という行動を表現している．実行時にはこのルーチンが 1 フレームごとに実行されるため，結果として，スキル 1 のクールタイムが経過するたびに攻撃し続けることとなる．

4. 評価

4.1 目的と方法

本システムの有用性の確認のため，プログラミングに対する動機づけの変化を確認した．被調査者は，筆者らの所属する大学の情報工学部 1 年生 23 名であり，その内訳は，実験群 12 名および統制群 11 名である．実験群および統制群は，本調査と並行して，1 年前期の必修科目である C 言語のプログラミングの講義を受講した[*2] 本学部のシラバス [13] より抜粋した当該講義「プログラミング」の達成目標および授業項目を以下に示す．

- 授業の達成目標
  情報工学技術者が備えておくべき情報処理の基礎を育成することを目標とする．本科目は，全学科において，情報処理の基礎に関する学習・教育到達目標に位置付けられる．情報工学の最初の科目として，今後の学修に必要となる計算機の使用法や C 言語による手続き型プログラミングの基礎の修得をテーマとする．具体的には，次の事項を到達目標とする．
  —Linux が動作する計算機上で，ファイル操作などの基本的な計算機操作ができる．
  —C 言語の基本的なデータ型，演算，制御構造，入出力を理解し，簡単なプログラムを作成できる．
  —与えられた問題に対して，プログラムを作成して簡単な問題解決ができる．
- 授業項目
  (1) 計算機の利用法と情報倫理（情報モラル，著作権，不正アクセスなど）
  (2) エディタ/ C 言語によるプログラミング入門
  (3) 日本語入力/入出力関数，データ型
  (4) 電子メール/選択（if 文，switch 文），選択の入れ子
  (5) UNIX コマンド（ファイル，ディレクトリの操作）/反復（while 文，for 文，do-while 文）
  (6) UNIX コマンド（パイプとリダイレクション）/反復の入れ子
  (7) UNIX コマンド（プロセスとシェルスクリプト）/最大最小，不定個数のデータ入力
  (8) UNIX のユーザ管理とファイル保護/配列とソート
  (9) 多次元配列
  (10) 関数の定義と利用
  (11) これまでの復習と理解度の確認，および解説
  (12) ポインタ，再帰呼出し
  (13) 文字と文字列
  (14) 応用プログラム演習
  (15) 期末試験
  (16) 期末試験問題と解答の解説

実験群については，プログラミングの講義に加え，前期

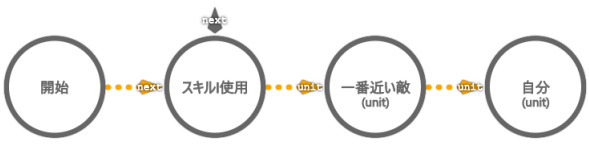

図 4　タイルスクリプトの例
Fig. 4　An example of tile script.

[*2] 実験群は対面により実施され，統制群はコロナ禍のため遠隔にて実施されたが，内容は同じであり可能な範囲で実施形式を揃えている．システム利用前の動機づけについて，実験群と統制群には統計的な有意差はみられなかった．詳しくは，4.2 節で述べる．

表 1　各尺度の質問項目
Table 1　Questionnaires for each category.

効力期待	
(1)	自分の意思で計画通りにプログラミングすることができる.
(7)	プログラムを完成させることができると思う.
(12)	プログラミングはやろうと思っても自力ではできそうにない. (*)
(16)	こういう風にプログラミングしようと思ったら，その通りにできると思う.
興味価値	
(2)	プログラミングのことを考えると楽しい気分になる.
(4)	プログラミングには興味がわく.
(8)	プログラミングのことを考えると，わくわくすることがある.
(13)	プログラミングは面白い.
私的獲得価値	
(3)	プログラミングすると，なりたい自分に近づける.
(5)	プログラミングすることは，自分にとって重要だ.
(11)	プログラミングは，自分で進んで行う価値がある.
(15)	プログラミングすると，自分がもっと成長できると思う.
身体的要因	
(6)	プログラミングを始めても，いすに座っていられない. (*)
(9)	プログラミングするときには，体が疲れている. (*)
(10)	プログラミングするときには，気分が常にだらけている. (*)
(14)	プログラミングしようと思っても気合が入らない. (*)

(*) 反転項目

の中旬である 6 月から週 1 回，90 分程度本システムを利用し，それを 5 週にわたり繰り返した．初回については，本システムの使用方法を説明した後で，キーボード操作のみでゲームを体験させ，問題発見，要件分析，設計，実装およびテスト検証を自ら行うよう指示した．また，本システムを使用する前と 5 週目の使用後に，プログラミングに関する動機づけについて，アンケート調査を行った．統制群については，本システムを利用せずプログラミングの講義の受講のみを行い，実験群の事前および事後と同様の時期にアンケート調査を実施した．なお，すべてのアンケートは，同一の物を使用した．

本調査では，総合的動機づけ診断 [14] より，主観的やる気との関連が示された以下の尺度を利用することとした．

- ある行動を適切に行うことができるという効力期待
- 楽しさや面白さを扱った興味価値
- 自らにとって望ましい自己概念を獲得できると価値づける私的獲得価値
- 疲労など身体的な影響を扱う身体的要因

アンケートでは，これら 4 種類の尺度に対応する各 4 項目，計 16 項目 [14] を用い，対象をプログラミングまたはプログラムとし，質問項目の意図が変化しないよう注意しつつ，できるだけ自然な文となるよう最小限の変更を加え，利用した．また，被調査者には，1（そう思わない）から 5（そう思う）の 5 件法で回答を求めた．**表 1** に，各尺度の質問項目を示す．同表において，括弧内の数字は項目番号を示す．集計時には，反転項目の値を反転処理し，各尺度および全尺度の合計を求め，それぞれについて群（統制群，実験群）×調査（事前，事後）の 2 要因混合計画の分散分析（非加重平均法）を行った[*3]．

4.2　結果と考察

実験群の被調査者は，本システム内の対戦型ゲームにおいて自ら問題を発見し，要件分析，設計，実装，テスト検証というプロセスを繰り返し遂行した．その結果，平均で 4.2 個（標準偏差 1.4）のルーチンを完成させた．なお，完成の基準は意図どおりに動作したか否かであり，各被調査者の判断による．1 つのルーチンが完成したということは，本システムの環境において被調査者が問題を発見し，要件を分析し，設計・実装し，テスト検証を行うことができたことを意味する．この結果より，プログラミングの初学者であっても，本システムの環境において，一連のサイクルを繰り返して問題解決を行うことが可能であったといえる．

続いて，動機づけに関するアンケート調査について述べる．統制群および実験群の基本統計量を**表 2** および**表 3** に示す．まず，質問項目全体の合計値について述べる．**表 4** に示すように，各主効果および交互作用については，統計的な有意差はみられなかった．統制群および実験群については，群ごとに希望者を募り被調査者を集めたため，事前調査結果をもとにした編成を行うことはできなかった

[*3] 統計処理には，js-STAR XR+ release 1.3.0 j（https://www.kisnet.or.jp/nappa/software/star/）を用いた．

表 2　統制群の平均と標準偏差
Table 2　Mean and standard deviation of the control group.

	事前		事後	
	平均	標準偏差	平均	標準偏差
効力期待	13.00	2.63	12.91	3.29
興味価値	15.18	3.74	14.45	4.36
私的獲得価値	17.09	2.27	17.36	2.74
身体的要因	15.45	2.54	14.55	3.96
全体	60.73	9.51	59.27	13.18

表 3　実験群の平均と標準偏差
Table 3　Mean and standard deviation of the experimental group.

	事前		事後	
	平均	標準偏差	平均	標準偏差
効力期待	11.17	3.72	13.33	2.90
興味価値	14.50	3.01	14.33	3.20
私的獲得価値	14.58	2.43	15.42	1.85
身体的要因	15.08	2.90	14.83	3.39
全体	55.33	10.70	58.08	10.14

表 4　全体の分散分析結果
Table 4　ANOVA table on the total.

変動因	平方和	自由度	平均平方	F
群	124.37	1	124.37	0.51
調査	4.82	1	4.82	0.28
交互作用	50.73	1	50.73	2.94
誤差	362.49	21	17.26	
全体	5695.86	45		

表 5　効力期待の分散分析結果
Table 5　ANOVA table on efficacy expectations.

変動因	平方和	自由度	平均平方	F
群	5.70	1	5.70	0.29
調査	12.36	1	12.36	5.73*
交互作用	14.63	1	14.63	6.78*
誤差	45.29	21	2.16	
全体	493.93	45		

*($p < .05$)

が，事前調査における両群の統計的な有意差は認められなかった．また，交互作用が有意ではなかった，つまり，実験群においても事前と事後の調査結果に有意差はみられなかったため，質問項目全体での本システムの有用性は確認できなかった．

次に，各尺度の分散分析結果について述べる．**表 5**〜**表 8** に示すように，効力期待における調査の主効果と交互作用，および，私的獲得価値における群の主効果が有意であった．**表 7** のとおり，私的獲得価値については，群の

表 6　興味価値の分散分析結果
Table 6　ANOVA table on interest value.

変動因	平方和	自由度	平均平方	F
群	1.85	1	1.85	0.07
調査	2.29	1	2.29	1.86
交互作用	0.90	1	0.90	0.73
誤差	25.92	21	1.23	
全体	599.08	45		

表 7　私的獲得価値の分散分析結果
Table 7　ANOVA table on personal attainment value.

変動因	平方和	自由度	平均平方	F
群	56.94	1	56.94	5.88*
調査	3.51	1	3.51	1.54
交互作用	0.90	1	0.90	0.40
誤差	47.92	21	2.28	
全体	312.64	45		

*($p < .05$)

表 8　身体的要因の分散分析結果
Table 8　ANOVA table on physical factors.

変動因	平方和	自由度	平均平方	F
群	0.02	1	0.02	0.00
調査	3.86	1	3.86	1.70
交互作用	1.25	1	1.25	0.55
誤差	47.58	21	2.27	
全体	487.16	45		

表 9　効力期待に関する単純主効果検定の結果
Table 9　Simple main effects on efficacy expectations.

変動因	平方和	自由度	平均平方	F
事前における群の効果	19.29	1	19.29	1.68
事後における群の効果	1.03	1	1.03	0.10
統制群における調査の効果	0.05	1	0.05	0.02
実験群における調査の効果	26.94	1	26.94	12.49**
誤差	45.29	21	2.16	

**($p < .01$)

主効果が有意であり，実験群と統制群で等質に分けられていなかった（$F(1, 21)=5.88$，$p<.05$）．調査の主効果および交互作用は有意ではなく，両群ともに事前と事後の間での変化は確認されなかった．効力期待については，表 5 に示すように，交互作用が有意であったため，下位検定として，単純主効果検定を行った．その結果，**表 9** のとおり，実験群における調査の効果が有意であった（$F(1, 21)=12.49$，$p<.01$）．

効力期待は，ある行動を自ら成功裏に実行できるという確信であり [15], [16]，本調査においては「意図したプログラムを作成すること」がその行動にあたる．その成功体験

を繰り返すことで，「意図したプログラムを作成できるという確信」につながると考える．その支援のため，本システムでは，ゲーム環境下において学習者自身が解決すべき課題を発見し，学習者の習熟度に合わせて簡略化されたタイルスクリプティングにより発見した課題を解決できるプログラミング環境を提供した．これにより，プログラミングに不慣れな学習者であっても，課題を自律的に選択しながら繰り返しソフトウェア開発を行い，自分の意図どおりにプログラムが動作するという成功体験を得ることが可能になり，プログラミングについての動機づけとして，効力期待が高まったと考える．

5. おわりに

本稿では，プログラミングの初学者を対象とし，プログラミングに対する動機づけを高めることを目的としたソフトウェア開発環境について述べた．学習者は，対戦型ゲームにおいてプログラミングにより解決可能な問題を自ら発見し，戦術およびそれに伴うユニットの動作として解決策を考案し，スクリプトエディタを用いてルーチンを実装し，再びゲーム内においてルーチンの検証を行うという流れで，プログラミングに不慣れな初学者であっても，高度な専門知識を要求されることなく実践経験を積むことができる．問題発見の場である対戦型ゲームには，十分な問題発見の余地が確保されており，加えて学習者を実践へと向かわせる方向付けを行うことで，学習者はモチベーションを維持しつつ繰り返し実践に取り組むことが期待される．本研究では，情報工学の最初の科目としてC言語を学習している学生に対して調査を実施し，上流工程のソフトウェア開発を繰り返すことにより，効力期待の向上を確認することができた．本システムの利用により，プログラムを作ること自体の動機づけ向上の一助となり，C言語のような一般のプログラミング言語の学習へ良い効果が得られる可能性がある．

今後は，一般のプログラミング言語の学習への波及効果について確認する予定である．本研究の調査は，C言語の学習の中期に実施したため，プログラミングが苦手な学習者にとっては，プログラミングの学習に困難を感じはじめネガティブな印象が徐々に高まっていく時期であった可能性がある．そのような学習者を含め，本システムの利用により，プログラミングへの動機づけを高め，プログラミングの講義での成績向上へつながるか否かを確認する．なお，その際には，信頼性をより高めるため，他大学での実践や他のプログラミング言語を最初に学ぶケースでの実践も含めて検討したいと考えている．また，プログラミングの学習につまずく前，つまり，講義を受講する前段階での利用や，いくつかのプログラミングの講義を受講した後のプログラミングの動機づけが非常に低い学習者への効果も確認する予定である．さらに，動機づけの調査方法についても検討する予定である．本調査では，1つの尺度を複数の質問項目で構成することにより，回答者の意識的操作が現れるという質問紙法の短所を軽減しつつ，外的に観察不可能な意識的側面である動機づけに関する情報の収集を試みた．動機づけに関する情報については，被調査者の行動観察や提出物の分析等でも入手できると考えられるため，多面的な調査・分析を検討したいと考えている．また本調査では，統制群は通常の講義を受講し，実験群は通常の講義に加え本システムを用いた演習を行った．このような総合的な学習時間が異なることによる影響も考えられるため，時間を揃えた上での調査も行う予定である．

謝辞　システム評価のための調査に協力していただいた坂本洵一郎氏，池村政俊氏，山﨑祥平氏，山下聖嗣氏，および，被調査者各位に謝意を表する．

参考文献

[1] 竹田尚彦，大岩　元：プログラム開発体験に基づくソフトウェア技術者育成カリキュラム，情報処理学会論文誌，Vol.33, No.7, pp.944–954 (1992).
[2] 尾崎浩和，富永浩之，林　敏浩：ボードゲーム戦略を題材とする問題解決型プログラミング演習支援–持続的な取組みを促進する大会方式と運営サーバ，情報処理学会研究報告，Vol.2008, No.26, pp.1–8 (2008).
[3] Resnick, M., Maloney, J., Monroy-Hernáandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y.: Scratch: Programming for All, *Commun. ACM*, Vol.52, pp.60–67 (2009).
[4] 深谷和義，宮地晶子：小学生向けプログラミング授業のための「プログラミン」利用の検討，日本教育工学会論文誌，Vol.36, No.Suppl., pp.9–12 (2012).
[5] 古池謙人，東本崇仁，堀口知也，平嶋　宗：プログラミングの構造的理解を指向した部品の段階的拡張手法の提案と支援システムの開発・評価，教育システム情報学会誌，Vol.36, No.3, pp.190–202 (2019).
[6] 岩崎サオ里，阪東哲也，長野仁志，藤原伸彦，曽根直人，山田哲也，伊藤陽介：小学校家庭科「消費者教育」における能動的な学習を促進するプログラミング教育実践の提案，日本産業技術教育学会誌，Vol.63, No.1, pp.75–82 (2021).
[7] 下田　篤，矢吹太朗，田隈広紀，竹本篤郎，堀内俊幸：大学におけるソフトウェア開発PBL，プロジェクトマネジメント学会誌，Vol.16, No.2, pp.15–20 (2014).
[8] 佐藤和彦，服部　峻，佐賀聡人：主体的学習を促す実践的ソフトウェア開発演習の改善とその効果の検証，コンピュータ & エデュケーション，Vol.46, pp.64–69 (2019).
[9] 槇原絵里奈，井垣　宏，吉田則裕，藤原賢二，川島尚己，飯田　元：ソフトウェア開発PBLにおけるビルドエラーの調査，情報処理学会論文誌，Vol.58, No.4, pp.871–884 (2017).
[10] 藤田哲也：絶対役立つ教育心理学，ミネルヴァ書房 (2021).
[11] 松本慎平，加島智子，山岸秀一：プログラミング学習前に行われたプログラミングゲームの理解度とその学習後の到達度との関係分析，情報教育，Vol.2, pp.31–40 (2020).
[12] 平山雅之，鵜林尚靖：ソフトウェア工学，オーム社 (2017).
[13] 九州工業大学：九州工業大学シラバス，九州工業大学

（オンライン），入手先〈https://edragonsyllabus.jimu.kyutech.ac.jp/guest/syllabuses〉（参照 2022-11-01）．
[14] 中西良文，伊田勝憲：総合的動機づけ診断に関する探索的研究，三重大学教育学部研究紀要，Vol.57, pp.93–100 (2006).
[15] Bandura, A.: Self-efficacy: Toward a unifying theory of behavioral change, *Psychological Review*, Vol.84, No.2, p.191–215 (1977).
[16] 竹綱誠一郎，鎌原雅彦，沢崎俊之：自己効力に関する研究の動向と問題，教育心理学研究，Vol.36, No.2, pp. p172–184 (1988).

國近 秀信（正会員）

1992 年九州工業大学情報工学部知能情報工学科卒業．1994 年同大学大学院修士課程修了．1996 年同大学大学院博士後期課程修了．博士（情報工学）．1996 年同大学助手．現在，同大学准教授．知的学習支援システム，特に言語学習支援に関する研究に従事．情報処理学会，電子情報通信学会，人工知能学会，教育システム情報学会各会員．

赤川 啓行

2014 年九州工業大学情報工学部知能情報工学科卒業．2016 年同大学大学院博士前期課程修了．