各教科内で Computational Thinking を育成することを目的とした Block 型言語環境の開発

The development of environment using block type language that aims to foster Computational Thinking at each subject

福井　昌則[*1]，　萩倉　丈[*2]，番庄　智也[*3]，　森山　潤[*4]，　平嶋　宗[*5]
Masanori FUKUI [*1]，　Jo HAGIKURA[*2]　Tomoya BANSHO [*3]　Jun MORIYAMA [*4]　Tsukasa HIRASHIMA [*5]
*1, *4 兵庫教育大学 学校教育研究科
*1, *4 Graduate School of Education, Hyogo University of Teacher Education
*2, *3 関西学院大学 理工学部
*2, *3 School of Science Technology, Kwansei Gakuin University
*5 広島大学 工学研究科
*5 Graduate School of Engineering, Hiroshima University
Email: m16195c@hyogo-u.ac.jp

あらまし：2017 年 3 月に，小中学校次期学習指導要領が公示され，2020 年から義務教育段階でプログラミング教育が必修化となることが具体化された．その中でプログラミング的思考の涵養が掲げられ，各教科内でコーディングを必須としない形で身につけさせることが期待されている．プログラミング的思考は，より広い概念である Computational Thinking(以下 CT と略記)を下地としており，CT を身につけうる方略を検討することで，プログラミング的思考を身につけうる方略を示すことに接続することができる．また，CT は広い範囲に適用可能な汎用的なものであり，CT の涵養は，多様な分野で活用できる力を身につけることに接続するだけでなく，CT の具体的な要素である分解や抽象化などを明示的な形で，具体的に行うことで問題を効率的に解決するためのスキルや技術に接続することが期待できる．そこで本研究では，Scratch を用いたことがある生徒が，各教科内で Computational Thinking を，コーディングを活用した形で身につけることへ接続するシステムを開発したので報告する．

キーワード：Computational Thinking，Scratch，openBlocks Framework，プログラミング的思考

1. はじめに

本稿は，Scratch を用いたことがある生徒が，各教科内で Computational Thinking を，コーディングを活用した形で身につけることへ接続するシステムの開発について報告することを目的としている．

2017 年 3 月に小中学校次期学習指導要領の公示がなされ，その中では以前から必修化が予定されていたプログラミング教育についての内容が具体化された[(1)]．そして，義務教育段階におけるプログラミング教育では，「プログラミング的思考」の育成が掲げられている．プログラミング的思考とは，自分の意図した内容をどのように実現するかという考え方を各教科内で身につけること，およびコーディングを必須としない特徴を有しており，問題解決を行うために活用することが掲げられている[(2)]．プログラミング的思考は，Wing の掲げた「Computational Thinking(以下 CT と略記)」を下地として作成されている．CT は，様々な問題をコンピュータが解決できるような形で整理し表現するときに使われる思考プロセスのことであり，Wing は概念を提示したにとどまっているが，適用範囲について広く言及している[(3)]．プログラミング的思考は CT を踏襲しながらも限定的に捉えている側面があり，CT を身につけること，および CT 育成の方略を検討することは，より広く汎用的な力を身につけることにつながる．また，CT の要素として，分解，抽象化，パターン化，アルゴリズム化などがあるが(例えば Barefoot Computing の資料を参照[(4)])，それぞれを明示的な形で，つまり具体的な作業として行うことができるようにし，コンピュータやコーディングを活用してもしなくても，問題を効率的に解決するためのスキルや技術の取得を目指すことが重要である．

CT の育成に関して，イギリスの National Curriculum で用いる Tip として，Computing in National Curriculum[(5,6)]があり，その中で CT を身につけるための具体的な方法論について述べられている．ISTE and CSTA は，Varr and Stephenson[(7)]を参考として，独自により実践的な Operational Definition of Computational Thinking を定義[(8)]し，その育成に関して Computational Thinking Teacher Resources[(9)] や Computational Thinking Leader's Toolkit[(10)]といった Tips を Web などで公開している．それらを見ると，CT の中心的な要素である Decomposition などに対して評価基準を示しており，CT を身につけるための具体的な方略などについて述べている．イギリスの National Curriculum では，プログラミングは教科「コンピューティング」の中で行うこととなっており，Scratch などを用いたコーディングの活用が必須となっている．そして，具体的な学習内容や目的を Computing Progression Pathway[(11)]に示している．その

中では，学習領域として① Algorithms，② Programming & Development，③ Data & Data Representation，④ Hardware & Processing，⑤ Communication & Networks，⑥Information Technology の 6 つの領域を掲げており，具体的な到達目標として，①Abstraction，②Decomposition，③Algorithmic Thinking，④Evaluation，⑤Generalization を設定し，発達段階に応じて内容を明確にしている．

またアメリカではイギリスのように National Curriculum はないものの，CSTA が教科「コンピュータ科学」(以下 CS と略記)を提案しており，そのカリキュラムとして CSTA Standards を公開している[12]．その中では，学習領域として① Computational Thinking，②Collaboration，③Computing Practice and Programming，④ Computers and Communications Devices，⑤Community，Global and Ethical Impacts Computational Thinking の 5 つの領域をあげており，問題解決のためにコーディングを含むプログラミングを活用することが掲げられている．アメリカにおいては，各州や各学校に教科の選択権があるものの，実際には CS が一つ重要な位置を占めており，方向としてはコーディングを活用する形が基本となっている．つまり，CS を基にした指導方法の中で検討されている内容は，CT 育成において重要である．

そして前述した通り，我が国ではコーディングを必須としない形でプログラミング的思考を育成することが期待されており，CT をコーディングなしで身につけるための方略の検討が必要となる．またその一方で，Computing Progression Pathway に示されているように，連続性を持たせたカリキュラムを実施するためには，Scratch を用いたことがある生徒が，次に扱うことができるシステムやツールを準備し，それを用いてコーディングを用いた授業へと繋ぐことが，我が国のプログラミング教育においても重要と言える．

ここで，CT の育成を行うための方略を検討する必要があるが，例えば，(1)コーディングなしで CT を育成する方略の検討，(2)コーディングを用いて CT を身につけさせる方略の検討，(3)コーディングを行なったことがない生徒が，コーディングを用いて CT を育成する方法へと接続する方略の検討，など様々なアプローチが考えられる．

総務省の資料によれば，プログラミング人材育成において身につけることが期待される力として，「創造力の向上」「課題解決力の向上」「表現力の向上」「合理性，論理的思考力の向上」「意欲の向上(内発的な動機づけ効果)」「コーディング・プログラミングスキルの向上」「コンピュータの原理に関する理解」の 7 つの力を掲げており，コーディングスキルの向上が重要な力の一つであるとしている[13]．また，それら 7 つの力は，21 世紀型スキルの大半をカバーできるとしている．ここで，プログラミング人材育成について，「総務省におけるプログラミング人材育成の取組とは，プログラミングに関する高度な技術者を直接的に育成しようとするものではなく，青少年(18 歳以下程度を指す．以下同じ．)の発達段階に応じたプログラミングに関する教育を通じて，将来の高度 ICT 人材としての素地の構築・資質の発掘を図ろうとするものである．」としており，より汎用的な力として，プログラミングを位置付けている．

21 世紀型スキルとは，白水らによれば，21 世紀の知識基盤社会で求められる能力であり，他者との対話の中で，テクノロジーも駆使して，問題に対する解や新しい物事のやり方，考え方，まとめ方，さらに深い問いなど，私たち人類にとっての「知識」を生み出すスキルのこと[14]であり，これらの力を育成することは，グローバルな視点で活躍する人材育成においても重要であり，コーディングを活用することは 21 世紀型スキルの涵養において重要なキーワードである．

上述してきた Scratch は，プログラミング導入教育としてよく知られており，実際に現場でもよく用いられている．Scratch は米国マサチューセッツ工科大学のメディアラボが開発した Squeak eToys をベースとした Block 型言語環境[15]であり，視覚的・感覚的に様々なプログラムを作成することができるシステムである．つまり，多くの生徒が用いることができる，かつ多くの教員が扱える Scratch をベースとした Scratch の次に使えるシステムを活用して，CT を育成する方略を検討することが一つ重要となる．CT を Scratch で育成することについて，例えば Brennan and Resnick は，Scratch を用いるために提案されたフレームワークを活用した K-12 の生徒向けのプログラミングを通して見た CT について述べた上で，より実践的な内容について述べている．そして，CT を Computational Concepts，Computational Practices，Computational Perspectives の 3 領域とすることを提案している[16]．太田らは，CT の解釈はまだ定まっていないところがあると報告[17]しているが，いずれにしても CT を広く身につけるためには，Scratch を活用し，可能ならばコーディングへと接続することが有用な手法の一つである．

ここで，Scratch をベースとしたシステム開発，つまり上述した(3)に関係する内容に関する研究として主原らは，Scratch などの Block 型言語で活用されている openBlocks Framework を用いたシステム oPEN を開発しており，学習者の負担を軽減するために，利用可能な部品の集合を段階的に増やせる仕組み(ステージ機能)を実装している[18]．また稲葉らは，主原らの研究を踏まえた上で，oPEN のステージ機能を活用できる機能を実装している[19]．しかし，oPEN は，プログラミング学習用といった位置付けで開発されていること，および Scratch の次に使うことを想定したシステムとして開発されているが，CT の育成にどう接続するかについての議論はなく，また Block を動かすことでコードが表示されること，

プログラミング学習用に Block を設定するといったことにとどまっており，コーディングを用いたカリキュラムの中でどう実践するかについての言及はなされていない．他に openBlocks Framework を活用した大畑らによる研究[20]においても，プログラミング学習用システムとして開発されており，Scratch を扱ったことのある生徒が Scratch の次に使うことが可能なシステムとして提案はなされているものの，それらを用いて CT をどのように身につけるかについての方法論や活用方法についての報告は国内において見られず，また Scratch をベースとしたシステムで CT 育成を行うためのシステム開発はなされていないのが現状である．そこで本研究では，特に(3)を達成することを目的とした上で，その第一段階として，Scratch を用いたことがある生徒が，各教科内でコーディングを用いる形で適切に CT を身につけることへ接続する Scratch ライクなシステムを提案し，その実装を行なったので報告する．

2. CT を育成する Block 型言語環境の開発

2.1 システムの概要

本システム「Programming Next Step」は，Eclipse を用いて開発を行い，言語は Java(バージョンは 1.8x)を用いた．ここでライブラリとして openBlocks Framework[21]を用いた．openBlocks Framework とは，Scratch の GUI 部分を抜き出してプログラミング環境を構築することができるライブラリであり，ある反復や分岐の Block の中へ他の Block を入れた場合，反復や分岐の Block が自動的に伸長する．また各 Block と一対一対応するコードを XML によって自身で定義することができるなどといった機能を提供している．また，Java で開発していることから，Windows，Mac，Linux など様々なプラットフォーム上で動かすことができる．

2.2 システムの操作方法

本システムは，Scratch を扱ったことのある生徒であれば操作でき，Block を組み上げることでソースコードを表示することができる．図 1,2,3 に本ソフトウェアの画面を示す．ソフトウェアは，大きく分けて「プログラム作成画面」「ソースコード表示画面」「結果表示画面」の 3 つからなる．

ソフトウェアを立ち上げ，作成画面で左側の Block を右側の領域へ動かして，プログラムを行う．この操作は，Scratch を扱ったことがある生徒であれば，特に問題なく行うことができる．例えば，作成画面で以下の図 1 のように Block を組み合わせてプログラムを行うことができる．そして，左上に全体の中で今プログラムを行なっている場所がどこであるかについて表示する画面を実装している．そして，Block をドラッグし，右下のゴミ箱でドロップすると Block を削除することができる．また，Block で組み上げたプログラムの保存を行うことができる．

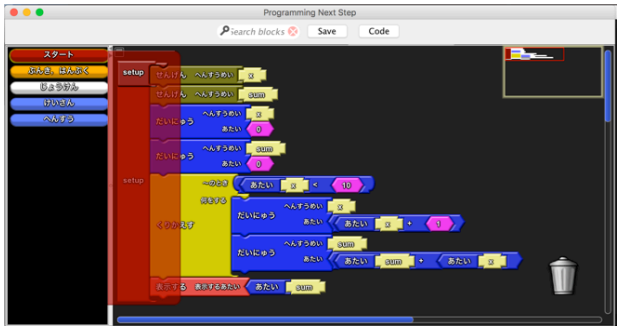


図 1 プログラムを行なっている様子
(「プログラム作成画面」)

そして，上部にある「Code」ボタンをクリックすると，「ソースコード表示画面」にコードが表示される．この図 2 では C 言語を表示しているが，様々な言語を表示することが可能である．

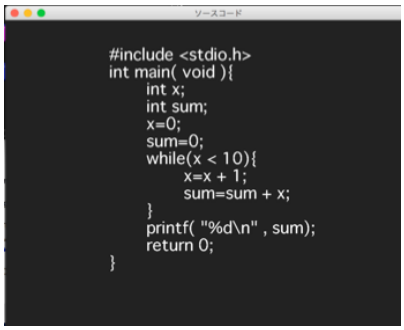


図 2 ソースコードが表示されている様子
(「ソースコード表示画面」)

このソースコードが表示されている状態で「結果表示画面」の「実行」ボタンをクリックすると，ソースコードに対応する結果が表示される．この様子を図 3 に示す．

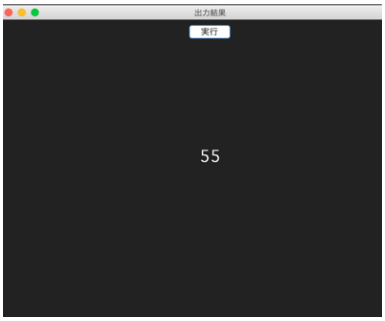
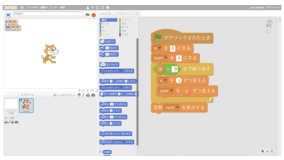


図 3 結果が表示されている様子(「結果表示画面」)

以上のシステムは，Scratch が扱える生徒であれば，問題なく操作を行うことができ，ソースコードを確認することができる．また，ソースコードの有用性を感じさせること，Block とソースコードが対応していることを理解させること，Scratch の学習後に使えるシステムの一つとしての役割を果たすことが期待できる．

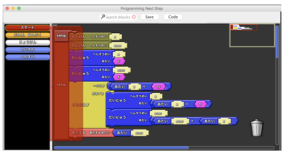
ここで，学びの連続性に着目し，Scratch，今回開発したシステム，実際のプログラミングにおける学びの形について，図 4 に示す．

①



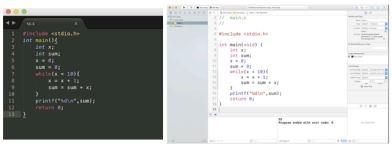
限定的で擬似的な世界の中で，遊びやゲームを通してSyntaxエラーを気にすることなしに，楽しく学ぶことができる．それによってCTを育成することができる．

②



Scratchを用いたことがあるユーザが，より広い世界の中で，様々な題材を通してSyntaxエラーをさほど気にすることなしに，より本格的なコーディングを行うことができ，アルゴリズムを学ぶことができる．それによってCTを育成することができる．

③



広い世界の中で，自由な題材を通して厳密なルールや文法に基づいてコードを書き，Syntaxエラーを回避しながら高いハードルを乗り越えることで，本格的で現実社会に用いることが可能な開発を行うことができる．それによってCTを育成することができる．

図 4 学びの連続性に着目した CT 育成の例

3. まとめと今後の展望

本稿では，Scratch を用いたことがある生徒が，各教科内で Computational Thinking を，コーディングを活用した形で身につけることへ接続する Scratch ライクなシステムの提案と開発について報告した．今後さらに CT 育成のために必要な機能の追加を行い，システムの有用性について検討を行う必要があろう．

参考文献

(1) 文部科学省："幼稚園教育要領，小・中学校学習指導要領等の改訂のポイント", (2017) http://www.mext.go.jp/a_menu/shotou/new-cs/__icsFiles/afieldfile/2017/06/16/1384662_2.pdf (2017.07.05 アクセス確認)

(2) 文部科学省教育課程部会小学校部会："小学校段階におけるプログラミング教育の在り方について(議論の取りまとめ)", (2016). http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo3/074/siryo/__icsFiles/afieldfile/2016/07/07/1373891_5_1_1.pdf (2017.07.05 アクセス確認)

(3) J M.Wing: "Computational Thinking", *Communications of the ACM*, **49**(3), pp.33-35 (2006)

(4) Barefoot Computing: "Coumputational Thinking" http://barefootcas.org.uk/wp-content/uploads/2014/10/Computational-thinking-Barefoot-Computing.pdf (2017.07.05 アクセス確認)

(5) Computing at School: "Computing in the national curriculum-A guide for primary teachers-" http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf (2017.07.05 アクセス確認)

(6) Computing at school: "Computing in the national curriculum-A guide for secondary teachers-" http://www.computingatschool.org.uk/data/uploads/cas_secondary.pdf (2017.07.05 アクセス確認)

(7) V.Barr, C.Stephenson: "Bringing computational thinking to K-12". (2011) http://www.amanyadav.org/CEP991A/wp-content/uploads/2014/08/Barr_Stephenson_2011.pdf (2017.07.05 アクセス確認)

(8) International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA): "Operational Definition of Computational Thinking for K–12 Education". (2011) http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf (2017.07.05 アクセス確認)

(9) ISTE, CSTA, NSF: "Computational Thinking Teacher Resources" http://www.iste.org/docs/ct-documents/ct-teacher-resources_2ed-pdf.pdf?sfvrsn=2 (2017.07.05 アクセス確認)

(10) ISTE, CSTA, NSF: "Computational Thinking Leader's Toolkit" http://www.iste.org/docs/ct-documents/ct-leadershipt-toolkit.pdf?sfvrsn=4 (2017.07.05 アクセス確認)

(11) Computing At School:"CAS Computing Progression Pathways KS1 (Y1) to KS3 (Y9) by topic" (2015). http://community.computingatschool.org.uk/resources/1692 (2017.07.05 アクセス確認)

(12) The CSTA Standards Task Force: "CSTA Computer Science Standards", (2012). http://www.education2020.ca/Content/K-12ModelCurrRevEd.pdf (2017.07.05 アクセス確認)

(13) 総務省："プログラミング人材育成に関する調査報告書", p.40 (2016). http://www.soumu.go.jp/main_content/000361430.pdf (2017.07.05 アクセス確認)

(14) 白水始，三宅なほみ，益川弘如，望月俊男(監訳・著)："21 世紀型スキル:新たな学びと評価の新たなかたち", 北大路書房，pp.207-223 (2014)

(15) Scratch Imagine, Program, Share MIT, http:// www.scratch.mit.edu/ (2017.07.05 アクセス確認)

(16) K.Brennan, M.Resnick: "New frameworks for studying and assessing the development of computational thinking", *AERA 2012*, pp.1-25 (2012)

(17) 太田剛，森本容介，加藤浩："諸外国のプログラミング教育を含む情報教育カリキュラムに関する調査-英国，オーストラリア，米国を中心として-"，日本教育工学会論文誌，**40**(3)，pp.197-208 (2016)

(18) 主原佑記, 赤井昭二, 中村亮太, 松浦敏雄："OpenBlocks を用いたプログラミング学習用ソフトウェアの開発", IPSJ 2014，Vol.2014-CE-124 No.**9**，pp.1-7 (2014).

(19) 稲葉夏希，中村亮太，松浦敏雄："段階的学習機能を備えた初学者向けプログラミング学習環境"，情報処理学会第 77 回全国大会, 3ZF-02, pp. 4-941-942. (2015)

(20) 大畑貴史，酒井三四郎，松澤芳昭：BlockEditor Hinoki: オブジェクト指向に対応したビジュアル-Java 相互変換技術の開発，教育システム情報学会 2013 年度学生研究発表会(東海地区)，pp.21-22 (2013).

(21) Ricarose Vallarta Roque: "OpenBlocks An Extendable Framework for Graphical Block Programming Systems", Electrical Engineering and Computer Sciences, Master's degree (2007).