Vol.2019-CE-150 No.4
2019/6/8

小学校教育に適した教育用プログラミング言語の提案

久保 文乃[1,a)]　久野 靖[2]

概要：近年の情報教育，中でもプログラミング教育の重要性の高まりから，2020 年に小学校でのプログラミングが必修となる．小学校におけるプログラミング教育の目的はコンピュータの役割を知りプログラミング的思考を養うことであり，この内容を学ぶに当たっては，初学者が取り組みやすい教育用の言語が用いられると考えられる．一方で，中学以降に汎用のテキスト言語へ移行する者も存在するため，テキスト型への移行に考慮する必要もある．そこで，初学者が取り組みやすく，かつテキスト型言語への移行を助けるような言語の開発を行なった．本稿では，まず既存の言語の特性について整理してから提案する言語が持つべき特性を検討した．その特性を元に言語を試作し，今後の評価計画について述べた．中学校以降ではテキスト型の言語を使用することも求められるため，ブロック型からテキスト型への移行を考慮した提案言語は，有用であり得ると考える．

A proposal for educational programming language suitable for elementary school education

1. はじめに

近年の情報技術の進展により，情報機器や情報システムが日常生活に大きく関わってくるようになり，情報教育の重要性が高まっている．中でもプログラミング教育は，情報機器の中核となるコンピュータの動作原理や特性を知るという点から，その拡充が急がれている．

わが国ではこのような流れを受け，2020 年から小学校でのプログラミング教育が必修となる [1]．文部科学省は，プログラミング教育の目的について，次のように定めている [2].

> プログラミング教育の目的は，コンピュータに命令を与え意図した処理を行わせることにより，コンピュータの役割を知ったり，論理的に考えて問題を解決するプログラミング的思考を養うことである

この内容を学ぶ上では，テキストでプログラムを記述する従来型のプログラミング言語 (以下「テキスト型言語」と記す) は必ずしも必要とされず，小学校段階に適するように工夫された教育用プログラミング言語の使用が想定される．実際，今日の多くのプログラミングスクールや小学校における試行では，Viscuit[3]，Scratch[4] など，図形的にプログラムを記述するビジュアル型の言語が多く使用されている．

これらの言語は，テキスト型で起こるような「構文エラー」が起きず，絵を簡単に動かすことができ，小学生に対する導入のハードルを低くし授業が進めやすいように工夫されている．

一方，中学校段階以降 [5] では，「ネットワークを利用した双方向性のあるコンテンツのプログラミング」を扱うこととしており，そのためにはテキスト型言語の使用が必要になるものと考えられる．

また，児童・生徒の中にはプログラミングに対してより深い興味・関心を持ち，教育用言語に提供されている範囲を超えた内容を扱いたいと考える者もいくらかは含まれる．そのような場合にも，汎用のテキスト型言語への移行

[1] 電気通信大学情報理工学研究科
Graduate School of Informatics and Engineering, The University of Electro-Communications, Chofu, Tokyo Japan
[2] 電気通信大学情報理工学研究科
Faculty of Informatics and Engineering, The University of Electro-Communications, Chofu, Tokyo Japan
[a)] k1831058@edu.cc.uec.ac.jp

が必要になる．

このため，小学校段階で使用する教育用プログラミング言語であっても，テキスト型への移行を考慮した設計のものが使われることが望ましい．筆者らは，このようなテキスト型言語への移行に配慮した，教育用プログラミング言語が持つべき機能や特性について検討し，実際にそのような言語を試作した．本発表では，その検討過程および試作した言語について報告している．

以下第 2 章では，既存の教育用プログラミング言語について，どのようなものがあり，それぞれどのような特性を持つかを中心に整理する．第 3 章では，第 2 章の結果に基づき，提案するプログラミング言語が持つべき特性や機能について検討する．第 4 章では，試作した言語の概要および特徴について説明する．最後に第 5 章では評価の計画およびまとめを述べる．

2. 先行研究

もともと，プログラミング言語はソフトウェア開発を職業とするプロフェッショナルが使用する道具として開発されてきた．これらはほぼ全てが，プログラムを文字の並び (テキスト) として記述するテキスト型である．以下ではこれらの言語を，(教育など) 特定の目的でなく，任意のソフトウェア開発を対象とするという意味で，汎用言語と記す．

その後，教育においてプログラミングを扱いたい場合，汎用言語そのままでは，機能が高度だったり複雑な記述を前提とするなどの理由から難しいため，教育における使用を考慮した言語が作られるようになった．その初期のものはテキスト型であったが，GUI (Graphical User Interface) の普及とともに，GUI を活用した教育用言語が作られ，広まるようになった．

以下では教育用言語をテキスト型，フローチャート型，ルールベース型，ブロック型に大別し，それぞれ具体例およびその得失について述べる．なお，テキスト型以外の言語はすべて，GUI 上でプログラムを構築するため，適正でない要素の配置を許さないなどの方法で，構文エラーを回避できるという性質は共通している．

テキスト型言語

前述のように，汎用言語がテキスト型であったことから，それをもとに初期の教育用言語が作られたことは自然である．テキスト型の教育用言語としては，Basic[6]，LOGO[7] が古くから使われている．とくに LOGO はタートルグラフィクス (タートルと呼ばれる小さな目印を画面上で指示に基づいて動かし，その軌跡によって図形を描画する) を取り入れたことで知られている [7].

また，兼宗ら [8] は，オブジェクト指向の考え方を大幅に取り入れ，日本の児童・生徒にとって扱いやすいように日本語の記述を基本とした教育用言語ドリトルを開発している．

テキスト型言語の利点としては，次のものが挙げられる．

- 汎用言語に近く，汎用言語への移行が容易である．
- アルゴリズムや手続きの記述において，汎用言語で培われた教育の知見がそのまま適用でき，コンピュータサイエンスを学ぶのに適している．
- 行数の多い，複雑なプログラムでも記述できる．

一方，テキスト型言語の弱点としては，次のものが挙げられる．

- すべて文字で打ち込むため，タイピングの技能に大きく依存する．
- 変数，手続きなどの概念はすべて文字で表されているため，これらの概念を文字上の記述から読み取り理解することが求められる．
- 構文や使用する API(ライブラリ) などは覚えている必要がある．(これらをサポートしてくれる開発環境はあるが，小学校レベルでそのようなツールを使うことは難しい．)
- プログラムの構文が少しでも正しくないと「構文エラー」が生じ，その除去は児童・生徒にとって負担となることが多い．

フローチャート型言語

フローチャートは計算処理，条件判断などを表す図形を実行の流れに対応する矢線でつなぎ合わせてプログラムの構造を表現する記法であり，中学校 (技術・家庭) や高校 (情報科) の教科書にも多く登場する．このため，ロボチャート [9](図 1) など，GUI でフローチャートを作成することでプログラムを表現できるようなツールが複数存在し，使われている．これらを総称してフローチャート型言語と呼ぶことにする．

フローチャート型言語の利点としては，次のものが挙げられる．

- フローチャートは実行の流れという学習すべき概念を分かりやすく表すことができる．
- 教科書にフローチャートが掲載されている場合，それとの対応がある．

一方，フローチャート型言語の弱点としては，次のものが挙げられる．

- テキスト型言語で使用されている構造化された条件分岐やループ構文と対応しない．([10] など，これらと対応する構造化チャートを用いるものもあるが，その場合は実行の流れを表すという利点が失われる．)
- 記述に場所を取り，作成が繁雑であるとともに，複雑なプログラム (1 画面に収まらないプログラム) を作るのは難しい．

ルールベース型言語 / Viscuit

Viscuit[3] は，NTT 研究所の原田によって開発された，ビジュアル型プログラミング言語である．Viscuit では，「めがね」と呼ばれる部品の左右に絵を入れ，左に入れた

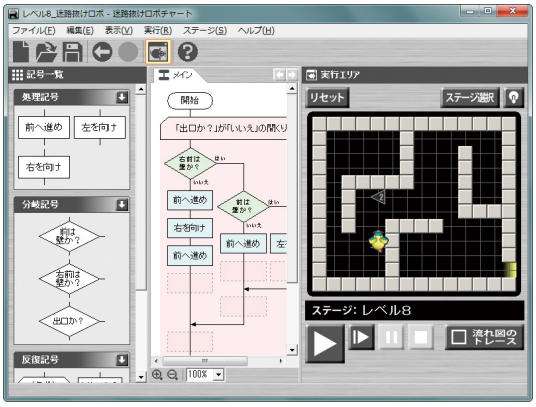


図 1 ロボチャートの画面

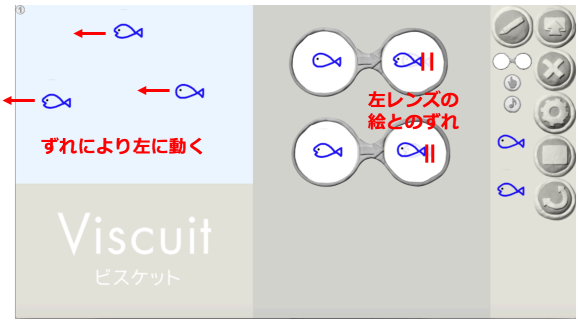


図 2 Viscuit の画面

絵の状態に当てはまる場面で，左右の状態の差異に相当する書き換えが行われることで，絵の移動や増減が記述できる (図 2)．すなわち，「めがね」が記述するルールの適用によってプログラムが実行されることから，本稿ではルールベース型言語と記す．

Viscuit の利点としては，次のものが挙げられる．

- 動作として絵を動かし，絵の配置によって状態が表されることは，直接的で分かりやすい．
- プログラムは文字を使用することも，また変数や繰り返しなど一般的なプログラミングで用いる要素も不要で，すべて「絵」と「めがね」によって構築される．
- 上記に関連して，文字の扱えない入学前の児童でも取り組むことができる．[11]

一方，Viscuit の弱点といては，次のものが挙げられる．

- 変数，制御構造などに相当するものが無く，テキスト型言語でこれらのものを使うことの準備とはなりにくい．
- 絵のパターンマッチに曖昧さを導入しているため[*1]，厳密な動作をおこなう他の言語と異なる特性を持つ．

ブロック型言語 / Scratch

ブロック型言語は，ループ，分岐，処理などに相当するブロックが部品として用意され，制御構造に相当するブロックの中に別のブロック (処理や他の制御構造) を「はめ込む」ことでプログラムを構成する方式である (図 3)．現在多くの国で子供にプログラミングを体験させる活動として行われている Hour of Code[12] の言語や，MIT で開発され児童・生徒むけのプログラミング言語として広く使われている Scratch[4] が代表的である．

[*1] 標準のモードでは曖昧さがあるが，曖昧さのないモードも存在する

図 3 Scratch の画面

とくに Scratch は，汎用言語での作品作りの代替となることをめざして開発されており，豊富な機能のブロック群が用意されている．また，Scratch は，今日の児童・生徒むけプログラミング活動で広く使われている．このため，本研究でも Scratch を対象として，その特徴を取り入れ，弱点を改める形で提案言語の設計をおこなって来た．このため以下では Scratch を前提に特徴を検討する．

Scratch の利点としては，次のものが挙げられる．

- 命令の順次実行，枝分かれや反復などの制御構造や，変数とその書き換え，計算式，条件式などの概念は，汎用言語と基本的に同じであり，汎用言語への以降にある程度適している．
- ブロックをはめ込んだ形は入れ子構造をよく表しており，入れ子になった制御構造の概念の理解に適している．
- スプライトと呼ばれる予め用意された絵を動かすプログラムが簡単に作れ，動くプログラムまでのハードルが低い．

一方，Scratch の弱点としては，次のものが挙げられる．

- はめ込まれたブロックの見た目はそれ自体では分かりやすいが，字下げで構造を表すテキスト型言語とは異なっており，その点ではテキスト型への移行にギャップがある．
- 機能が豊富であるぶん，多数の種類のブロックがあり，また各種の処理を行うブロックや条件を参照するブロックではその中に計算式，変数などをはめ込む必要があり，複雑である．
- 「○○と言う」というブロックと「○○と□秒言う」というブロックのように，指定の有無により使用するブロックが違う設計となっており，そのこともブロックの種類を多くしている．

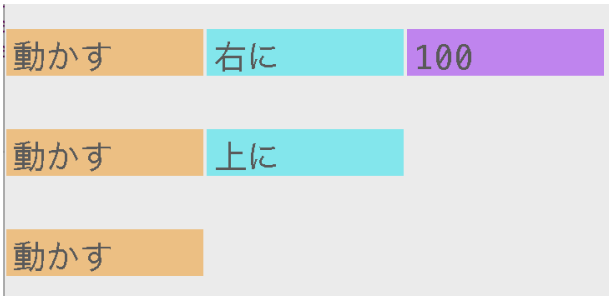


図 4 パラメタの有無

3. 提案言語の設計方針

本節では，従来の教育用言語の中でも，特に使用されることが多いブロック型言語について，問題と考えられる点を挙げ，それぞれについて提案する言語でどのように対応するかの方針を説明する．

ブロック型のテキスト型とのギャップ

ブロック型の言語には，プログラムを一から入力する手間が省けたり，文法の間違いによるエラーを防げるという利点がある．一方で，見た目や，プログラムの組み立て方においてテキスト型の言語とギャップが存在する．

例えば，ブロックにパラメータを与える際，そのブロックの中にブロックをはめられるという点である．ブロックの中にブロックをはめるという動作や，その見た目はテキスト型の言語と乖離がある．

開発する言語ではブロックの中にブロックをはめるということはできないようにし，ブロックを横に並べる，あるいは下につなげることのみによりプログラムを組み立てる．パラメータが必要なブロックの場合，パラメータはそのブロックの右につなげることにより表現する．また，パラメータを与えなかった場合のデフォルトの値を用意しておき，簡単に初学者がブロックを使用できるようにする (図 4)．

また，ブロック型言語では，繰り返しのような制御構造は，コの字型のブロックの中に処理するブロックをはめて表現する．これもテキスト型の言語との違いの一つだと考えられる．そこで，この言語では制御構造はインデントを用いて表す (図 5,6,7)．テキスト型の構造と親和性の高い表現を用いることにより，テキスト型への移行を助けると考える．

ブロックの配置できる位置の自由さ

ブロックを自由に配置し組み立てられるのには，プログラムを見たときに全体が整理されておらずわかりづらかったり，画面に表示しきれていないプログラムを見落とすかもしれないという可能性がある．そこで，テキストエディタのようにブロックを置ける位置を制限することにより，プログラムを整頓させて見やすくテキストに近い見た目のプログラムを実現する．このことにより，手続き型の言語

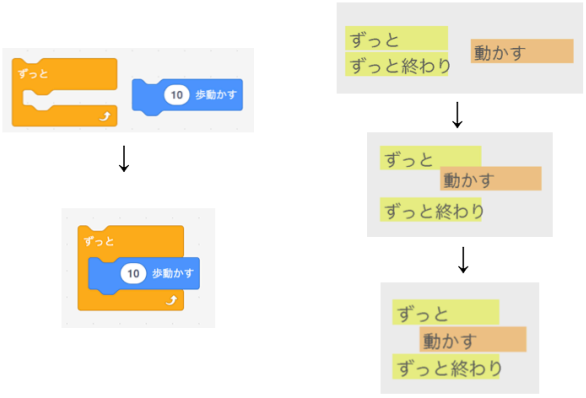


図 5 Scratch と開発した言語におけるブロック挿入の比較

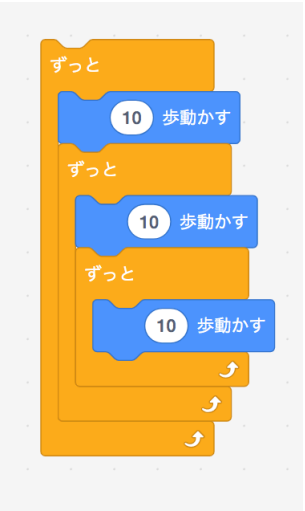


図 6 Scratch の制御構造

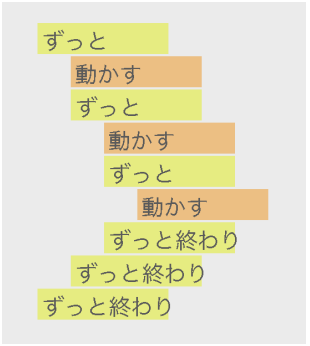


図 7 開発する言語の制御構造

における命令の順次実行の流れもわかりやすくなるのではないかと考える．

ブロックの多さ

Scratch の特徴としてブロックの多さが挙げられる．たくさんのブロックから必要なブロックを選ぶのは，初心者にとって困難である．ブロック数が多い理由の 1 つとして，様々な動きに対応するブロックが用意されているという点がある．しかし一方で，同じ命令のブロックでもパラメータ数によって異なるブロックとして扱っていたり，他のブロックの組み合わせで代用できるブロックが存在するという理由もある (図 8)．そこで，そのようなブロックをなくしたり，命令ブロックに対して任意の個数のパラメータを与えられるようにすることで，ブロック数を少なくする．

座標の難しさ

これまでの教育用言語では，タートルグラフィクスと直交座標が多く用いられてきた．タートルグラフィクスは相対的な座標でオブジェクトを動かすことができる．つまり，現在の座標がわかっていなくても簡単にオブジェクトを移動させることができる．しかし，相対的な角度や距離

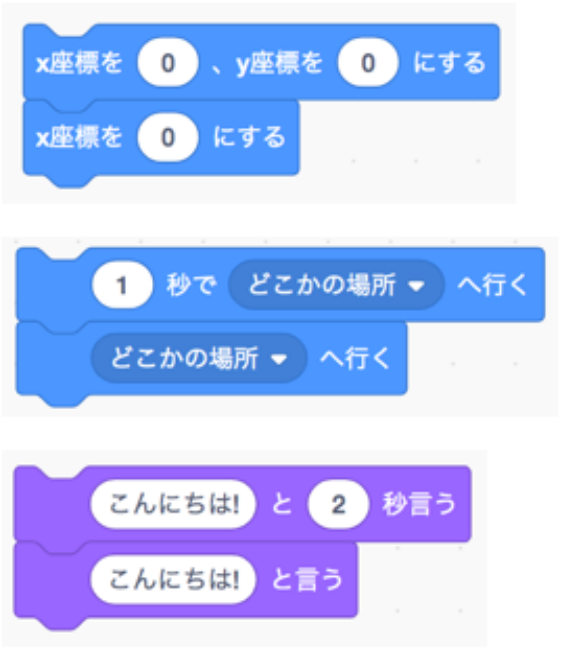


図 8 パラメータ数により異なるブロックとして扱われる例

を考えて任意の場所へ移動するというのが難しいという欠点もある．一方，直交座標では任意の場所への移動は座標を与えることにより簡単に行える．しかし，小学校までのカリキュラムでは座標は扱わず，かつ扱うグラフの範囲も正の値までである．そのため，オブジェクトの移動にマイナスを用いたり，x 座標・y 座標という言葉を用いるべきではない．

そこでこの言語では，相対角度ではなく，時計の 12 時の方向を 0 度とする絶対角度を用いることにし，オブジェクト移動の際の向きは，上下左右か絶対角度を指定するようにする．また，x と y を”たて”と”よこ”で表し，扱う範囲は正のみとする．さらに，画面外の座標を扱わないで済むよう，オブジェクトの座標が画面の範囲を越えた場合，その反対側につながるようにする．

4. 提案言語の設計と特徴

前節の方針を元に，言語を開発した．開発した言語の外観を図 9 に示す．

行単位のプログラム

従来のブロック型言語はブロックごとに形が異なっており，その凹凸が合うようにプログラムを組み立てる．これは文法の間違いによるエラーを防ぐことができる一方で，テキスト型の言語でプログラムを書く動作との乖離がある．そこでこの言語では，ブロックの形を全て凹凸のない長方形で統一し，コードの行を並べ替えるようにしてプログラムを組み立てる．

ブロックの出し入れは行の削除と挿入に対応する．また従来のブロック型言語の場合，繰り返しなどの制御構造はコの字型のブロックのくぼみに処理するブロックをはめて表すことができる．一方で開発した言語は行単位のプログラムのため，このような制御構造はインデントを用いることで表す．インデントを入れることは初学者にとって困難であることが考えられるため，インデントは必要な部分に自動で挿入されるようにし，入れ子構造も表現できるようにした．インデントを用いることにより，よりプログラムの構造を把握しやすくなると考える．

パラメタ

ブロック型の言語では，パラメタはブロックの中に入力したり，パラメタとしてさらにブロックをはめることによって表す．この言語は，命令やオブジェクトにパラメタを与えることによってプログラムを構成するが，パラメタはブロックの後に続けてつなげる．

また，全ての命令においてどういったパラメタが必要なのかを考えることは初学者にとって困難な可能性がある．そこで，パラメタを与えなくても済むようにし，必要な場合のみ与えるようにした．パラメタを与えなかった場合は，言語側で用意したデフォルトの値をパラメタとし，動作する．そうすることにより，まずは簡単な命令で動かすことができ，より高度な動きをつけたい場合や期待する動きをしなかった場合に，パラメタを考えて与えられるようになると考える．

例えば，図 4 は「動かす」に対して方向と距離のパラメータを与えているもの，方向のみ，何も与えないものの 3 つの場合を示しており，どの場合でも動作する．

座標系

座標は原点 (0,0) を左下に置くことにより正の範囲のみを扱うようにし，位置を表す際にはたて：0 よこ：0 というように x,y という言葉を用いないようにした．また，オブジェクトの移動は，「上に１００」のように，向きと距離を指定することにより行う．

タートルグラフィックや Scratch に存在する相対回転はわかりにくいため，回転には時計の針の 12 時を 0 度として角度を指定し回転できるようにした．

5. 評価の計画とまとめ

提案言語は現在，設計を検討しながら実装をすすめている状態である．今後，実際に児童・生徒に使ってもらう形で評価をおこなうことを計画している．その際，Scratch の経験を持つ児童・生徒が多くいることを考慮して，Scratch と提案言語で類似した課題をやってもらって比較する方法が考えられる．そのとき，本言語の特徴となっている個々の設計について，その部分が重要となる課題を選ぶことで，本言語の想定している利点が実際に長所となっているかを検討することとしたい．

はじめにで述べたように，小学校でのプログラミング必修化はまもなく開始されるが，そこで使われる言語として

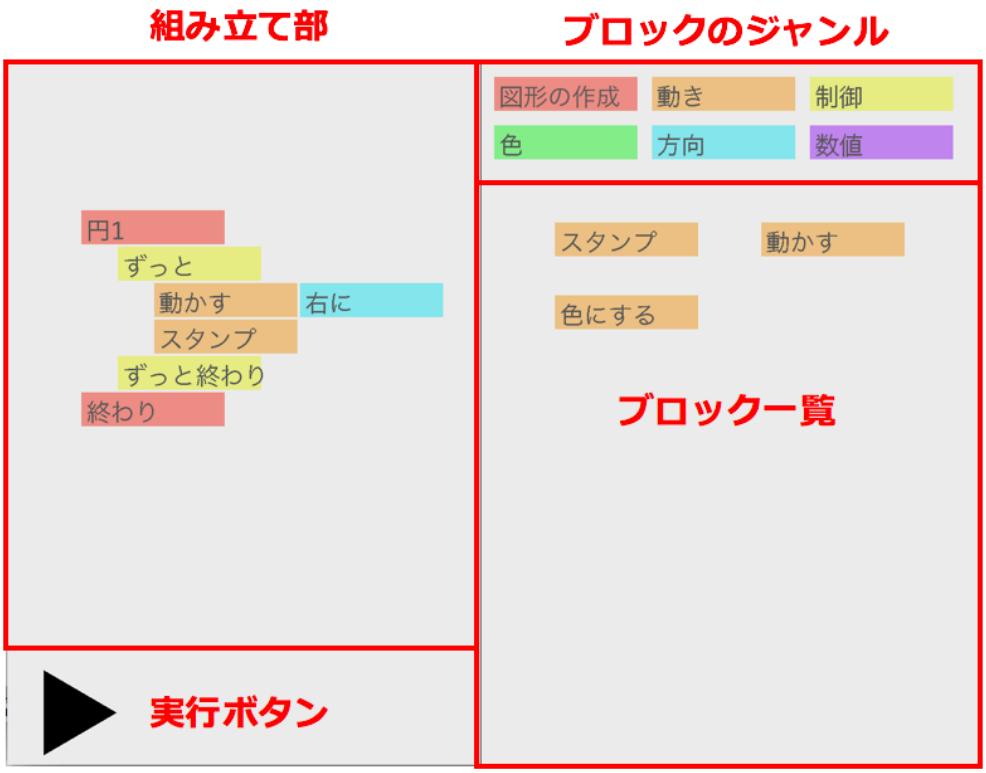
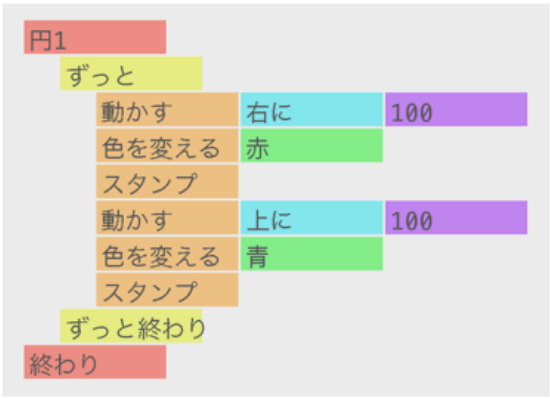


図 9 開発した言語



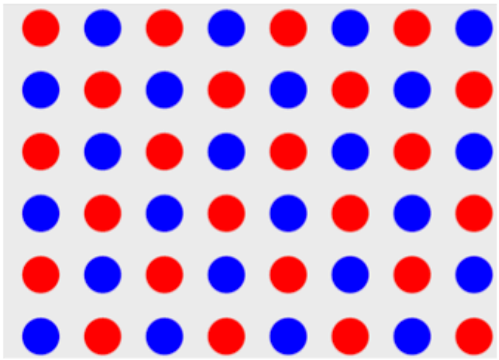


図 10 円の色を変えながら右と上にジグザグに動かしその軌跡を表示するプログラム

はビジュアル型の言語，とりわけ Scratch が多くなることが予想される．しかし，中学校段階以降ではテキスト型の言語を使うことも求められると考えられ，そのときブロック型からの移行のしやすさが問題となる．この点で提案言語は，ブロック型からテキスト型への移行を考慮した設計となっており，有用であり得るものと考えている．

参考文献

[1] 文部科学省，小学校学習指導要領 (平成 29 年 3 月告示)，2017 年
[2] 文部科学省，小学校プログラミング教育の手引 (第二版)，http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/11/06/1403162_02_1.pdf，2018 年
[3] Viscuit，http://www.viscuit.com/
[4] Scratch，https://scratch.mit.edu/
[5] 文部科学省，中学校学習指導要領 (平成 29 年告示 解説 技術・家庭編，2017 年
[6] JG Kemeny, TE Kurtz, The Dartmouth Time-Sharing Computing System Final Report, ED-024-602, NSF, 1967.
[7] S. Papert, Mindstorms: Children, Computers, And Powerful Ideas, Basic Books, 1980.
[8] 兼宗 進, 御手洗 理英, 中谷 多哉子, 福井眞吾, 久野 靖, 学校教育用オブジェクト指向言語「ドリトル」の設計と実装, 情報処理学会論文誌プログラミング (PRO)42(SIG11(PRO12)), pp.78-90, 2001 年
[9] ロボチャート，https://suzukisoft.co.jp/products/robochart/
[10] 斐品 正照, 松瀬 賢太, 河村 一樹, アルゴリズム学習を支援する JPADet の評価と教材開発，情報処理学会研究報告コンピュータと教育 (CE)2003(13(2002-CE-068)), pp.31-38,

図 11 2 つの円をランダムに動かすプログラム

2003 年
[11] 渡辺勇士, 中山佑梨子, 原田康徳, 久野 靖, ビスケットを使った未就 学児童に対するプログラミングレッスンの実践と考察, 情報処理学会 コンピュータと教育研究会, 2017-CE-142(13), 1-7 (2017-12-01), 2188-8930, 2017 年
[12] Hour of code, `https://code.org/learn`