ブロックプログラミング環境を用いたプログラム入力補助機能の提案

山本 大地[1]　本多 佑希[1]　島袋 舞子[1]　兼宗 進[1]

概要：初学者のプログラミングの入力を補助することを目的とし、教育用プログラミング言語「ドリトル」のプログラムをブロックの命令を組み合わせることで作成できるプログラミング環境を提案する。提案する環境では、プログラムの作成をブロックで行うことで、キーボードによる入力を不要にした。また、ブロックで作成したプログラムをテキストへと変換し、編集できるようにすることで、同一言語で入力方式をブロックからテキストへとシームレスに移行できるようにした。本稿では、開発したプログラミング環境の概要を報告する。

キーワード：プログラミング教育、プログラミング環境、入力補助

1. はじめに

現在使われているプログラミング言語の多くは、プログラムをテキストで記述する。初学者がテキストエディタを用いたプログラミングを学ぶ際、キーボード入力によるタイプミスなどが原因で構文エラーが発生する場合がある。このような構文エラーや記述ミスが何度も発生すると、学習者が意欲や興味を失ってしまう事も考えられる。このようなつまづきを防ぐために、これまで多くの教育用言語で工夫がなされてきた [1]。その一つに、命令を表すアイコンやブロックを組み合わせることでプログラムを作成することが出来る、ビジュアル言語がある。

本研究では、教育用のプログラミング言語「ドリトル」[2] にブロック入力機能を実装した。本環境をテキスト言語学習の前段階としてのブロックプログラミング環境として提案する。

2. 先行研究

Scratch [3] や MakeCode [4] などのビジュアル言語では、画面上の命令を表すブロックを組み合わせてプログラムを作成する。ブロックを組み合わせてプログラムを作成するため、キーボード入力ミスによる構文エラーは発生しない。また、ブロックの色分けによって、繰り返しや条件分岐の構造がわかりやすいという利点もある。数値やパラメータの入力はドロップダウンリストやキーボード入力で行う。

ブロックで作成したプログラムをテキスト言語へと変換する環境も存在する。Alice2 [5] は、ブロックで作成したプログラムを Java や C++に変換することができる。また、ブロックとテキストを相互変換できる環境も存在する [4] [6]。ブロックとテキストの変換を可能にすることで、ブロックとテキスト言語との対応を確認することができ、テキスト言語への移行をスムーズに行う手助けとなる。また、テキスト言語で文法の確認を行う際などにも役立つ。

このように、ブロックを利用したプログラムの開発環境は多く存在する。一方で、ブロックでプログラムを作成する際にキーボード入力がなく、かつテキスト言語への変換として、教育用言語に変換できる環境は多くない。

3. ブロックプログラミング環境の開発

3.1 プログラミング環境の設計

前章の内容をふまえて、ドリトルのプログラムをブロックの命令を組み合わせることで作成するプログラミング環境を開発する。キーボードによる入力を不要とするため、ブロックのパラメータはドロップダウンリストから選択し、入力できるようにする。また数値に関しては、ドロップダウンリストでは大きな数字などは扱いづらいため、任意の値も入力できるように対応する必要がある。ブロックとテキスト言語との対応を確認できるように、ブロックで作成したプログラムをテキストに変換できるようにする。また、ユーザーの学習環境に合わせるため、コンピュータとタブレット端末の両方の環境で動作できるようにする。

[1] 大阪電気通信大学
Osaka Electoro-Communication University

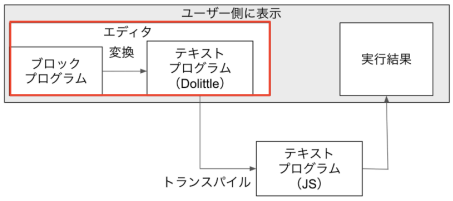

図 1　システム構成

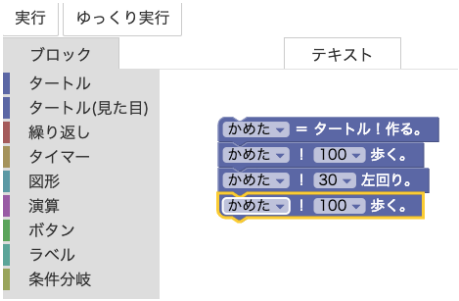

図 2　ブロックでのプログラム編集画面

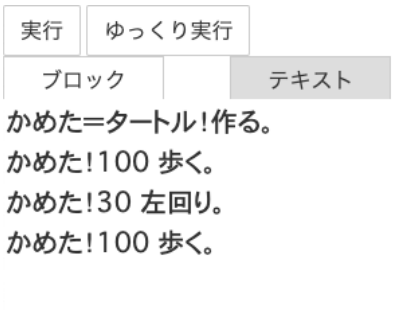

図 3　テキストでのプログラム編集画面

表 1　ブロック入力に対応した命令

オブジェクト	命令
タートル	作る/歩く/左回り/右回り/ 位置/変身する/移動する/衝突する/ ペンあり/ペンなし/線の色/現れる/消える
図形	図形を作る/歩く/左回り/右回り
ボタン	作る/動作/位置/次の行/塗る
タイマー	作る/実行/時間/回数/間隔
ラベル	作る
ブロック	繰り返す/なら/そうでなければ/実行

3.2　プログラミング環境の実装

本環境で使用するブロックの実装には Google 社が公開しているビジュアルプログラミングエディターの Blockly [7] を利用した。コンピュータとタブレット端末の両方で利用することができるように、ブラウザで動作するオンライン版ドリトルで実行できるようにした。図 1 に本環境のシステム構成を示す。実装した環境のプログラム編集画面を図 2、テキスト編集画面を図 3 に示す。プログラムの実行画面は図 9 のように表示される。また、ブロック入力に対応したドリトルの命令を表 1 に示す。

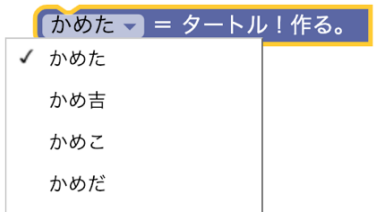

図 4　変数名選択リスト

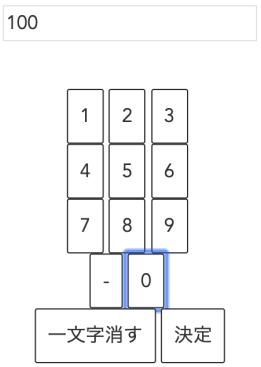

図 5　数値入力パネル

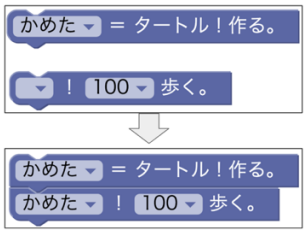

図 6　変数名の自動入力の例

3.2.1　変数名やパラメータの入力

変数名やパラメータはドロップダウンリストに予め格納されている候補から選択することで入力を行う（図 4）。

数値の入力にはドロップダウンリストの他に、画面上での入力できるパネルを実装した（図 5）。これにより、キーボードを使わず任意の数値を入力可能にした。

3.2.2　変数名の自動入力

ドリトルは、オブジェクト指向言語であるため、基本的にオブジェクトに対して命令する形でプログラムを記述する。今回、提案するプログラミング環境では、オブジェクトに対して命令する度にドロップダウンリストで対象の変数名を選ぶ必要がある。そこで、オブジェクト（タートル、図形など）に対して命令を行うブロック（作る、歩くなど）を下に繋げたときに自動で上のブロックと同じ変数名を設定するようにし、毎回の変数名の選択を不要とした（図 6）。

3.2.3　テキストへの変換

本環境では、ブロックで作成したプログラムをテキストに変換することができる。ブロックとテキストは、編集画面（図 2）の「ブロックタブ」「テキストタブ」で切り替える。ブロックとテキストを相互に確認し、照らし合わせることで、テキスト言語への移行をスムーズに行うことがで

きると考えられる。また、テキストで記述する際の文法の確認などにも役立つ。

3.2.4 プログラムの実行

作成したプログラムは、編集画面（図 2）左上の「実行」または「ゆっくり実行」のボタンをクリックすることで実行できる。「ゆっくり実行」ボタンは、タートルや図形の動きをアニメーションのように実行する。プログラムの流れが確認できるため、ユーザーの理解、プログラムのデバッグ等に役立つと考えられる。

4. 小学生向けプログラミング教室での利用

4.1 イベント概要

本環境を 2019 年 2 月に行われた小学生向けプログラミング教室で利用した（図 7）。イベントには、小学生が 20 名程とその保護者数名が参加した。プログラミング教室では講師役の学生が 1 人とサポートの学生 4 人が参加した。その場にいた主催者側の方がサポートに入る場合もあった。時間は 40 分で、本環境のブロック入力環境を利用し、簡単なゲームを作成した。作成したプログラムのサンプルを図 8 に実行画面を図 9 に示す。このプログラムは、まっすぐ歩く主人公のカメを画面上のボタンで左右に操作し、画面上にある花を集めるゲームである。プログラムの作成は、操作画面の見方や基本的な操作方法を伝えた後、プログラムを上から順に作って実行するということを繰り返す、という流れで行った。プログラムに関して、行き詰まったり不具合が出た場合は、その場で手を上げてもらい、サポートの学生が補助に入った。

4.2 利用した結果

本プログラミング教室ではほとんどの児童らが上手くゲームを作ることができた。筆者のこれまでの経験上、従来のテキスト環境では、テキストの入力に戸惑う様子などが見受けられる場合も多いが、本環境では分量の多いプログラムをスムーズに作ることができたと感じた。また、カメの歩く速度を変更したり、画面上の花を別の画像に差し替えたりと、自分なりにアレンジする様子なども多く見られた。配布した資料を見ながらどんどん先にプログラムを書き進めていく児童もいた。これは、ブロック入力による入力速度の向上と構文エラーの回避により、試行錯誤が行いやすくなったからであると考えられる。本プログラミング教室で本環境を利用した結果、本環境が小学生を対象として利用可能であることが確認できた。また、プログラムコードの長さや積極的な試行錯誤など、本環境における優位な点も確認できた。

4.3 改善点

児童が本環境を利用する際に、プログラムの編集画面をピンチインで拡大すると画面がフリーズする事があった。

図 7 プログラミング教室の様子

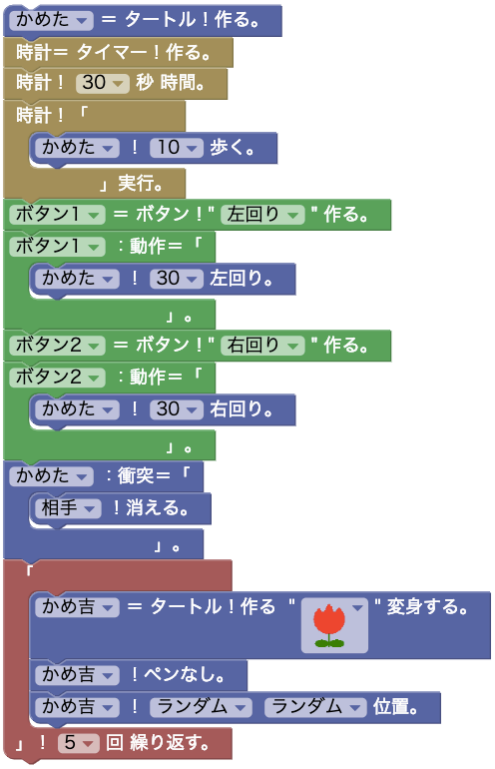

図 8 作成したプログラムのサンプル

フリーズした場合、ブラウザを再読込する必要があり、プログラムが消えてしまった。画面の拡大を禁止することで対応を行ったが、文字を大きくしたいケースも考えられる。そのため、この点に関しては、今後も検討を続けていきたい。また、実行ボタンが小さく押しにくいなどといった改善すべき点も見られた。

5. まとめ

キーボードによる入力を不要にし、構文エラーが発生しない、テキスト言語に変換できるブロックプログラミング環境を開発した。また、本環境を小学生向けプログラミング教室で使用することが出来た。試行錯誤やプログラムの分量などで、本環境における利点が示唆された。本環境でのタブレット端末では一部操作がしづらい部分があることがわかった。今後は改善点の改良と、ブロック入力環境か

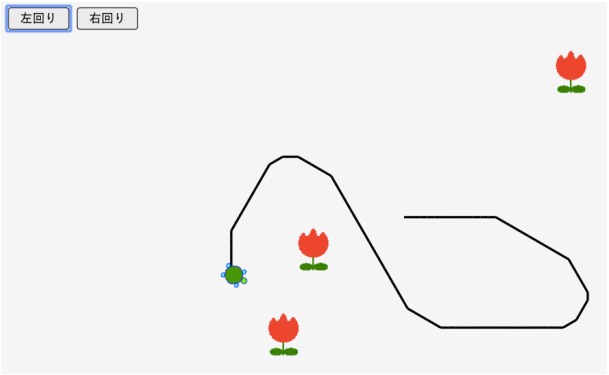

図 9 サンプルプログラム実行画面

らテキスト入力環境への移行についても検証を行いたいと考える。また、本環境と従来の環境でのプログラミングに関しての学習効果についても比較実験を行う予定である。

参考文献

[1] 兼宗進，阿部和弘，原田康徳：プログラミングが好きになる言語環境，情報処理, Vol.50, No.10, pp.986-995 (2009).
[2] 大阪電気通信大学 兼宗研究室：プログラミング言語「ドリトル」(オンライン), 入手先〈https://dolittle.eplang.jp/〉(参照 2019-5-30).
[3] Scratch Team Lifelong Kindergarten Group MIT Media Lab：Scratch - imagine, program, share (オンライン), 〈https://scratch.mit.edu/〉 (参照 2019-05-30).
[4] Microsoft：Microsoft MakeCode (オンライン), 入手先〈https://www.microsoft.com/ja-jp/makecode〉(参照 2019-05-30).
[5] Caitlin Kelleher, et al.: Alice2: Programming without Syntax Errors, In User Interface Software and Technology (2002).
[6] 松澤芳昭，保井元，杉浦学，酒井三四郎：ビジュアル-Java 相互変換によるシームレスな言語移行を指向したプログラミング学習環境の提案と評価, 情報処理学会論文誌, Vol.55, No.1, pp.57-71 (2014).
[7] Google：Blockly (オンライン), 入手先〈https://developers.google.com/blockly/〉(参照 2019-05-30).