吹き出し表示によるプログラム実行部分の可視化機能

小林 冠太[1] 伊藤 皓基[1] 寺田 和記[1] 高田 秀志[2]

概要：近年，情報化社会の発展に伴い，全国で多くの児童向けプログラミングワークショップが開催されており，プログラミング学習への世間の注目が集まっていると言える．しかし，現状のワークショップにおいて，児童にとって作りたいものを思い通りの形にするのが難しいという問題がある．この問題は，プログラムを動かした時に，分岐や反復をよく理解していないため，どの部分がいつ，どのように動いているか把握することが難しくなることが原因の一つであると考える．こうした問題点に基づき，Scratch において，作ったプログラムを組み込めるオブジェクトである「スプライト」に対して，吹き出しを使い，実行しているブロックをテキスト形式で出力することで，実行部分の可視化を行う．本機能により，児童は分岐や反復を理解し，プログラムの実行がどのように遷移しているか把握することが容易となることが期待される．実際に本機能を児童向けワークショップで適用し，有効性を検証した．その結果，本機能の使用により，分岐や反復の理解に一定程度繋がるとともに，間違いの修正を補助していることが確認できた．

Visualization Function Displaying Program Execution Part on Balloon

KANTA KOBAYASHI[1] KOKI ITO[1] KAZUKI TERADA[1] HIDEYUKI TAKADA[2]

1. はじめに

近年，情報化社会の発展に伴い，プログラミングの重要性が一般の人々にも広まりつつある．例えば，日本においては 2020 年度から，小中学校においてプログラミング学習が必修化される [1]．それに伴い，全国で児童向けプログラミングワークショップが多く開催されている．我々は 15 年以上に渡って，NPO 法人と協力し，児童向けプログラミングワークショップを開催している．これらのワークショップでは，児童がプログラミング能力を身につけるのが目的ではなく，あくまでプログラミングを通して，作品を創造的に作ることが目的である．

このようなワークショップにおいては，児童にとって自分の作りたいものを思い通りの形にするのが難しいという問題がある．作っているプログラムが複雑になるに連れて，分岐や反復などの要素が入り組んでしまう．そのため，分岐や反復をよく理解していなければ，作ったプログラムのどの部分がいつどのように動いているか把握できなくなる．このような場合に，児童は分からないことをサポーターに教えてもらうのではなく，自分で試行錯誤しながら正しく動くプログラムを作ることが重要であると考える．

そこで本研究では，上記で述べた問題点を解決すべく，プログラムの実行部分を可視化する機能を構築する．プログラムの実行部分を可視化することによって，児童が容易に目で見てプログラムの動きを確認できるようになり，分岐や反復の動きを理解して，プログラム全体の動きを把握することに繋がると考える．本研究ではワークショップで多く取り入れられている Scratch [2] というビジュアルプログラミング環境を使って，この機能を構築する．可視化方法は，スプライトに吹き出しを付けて，そこへ実行されているブロックをテキストとして出力する．例えば，「10 歩動かす」というブロックが実行されているとすると，それによって動いているオブジェクトの吹き出しにも「10 歩動かす」と出力される．

本機能を利用することによって，児童の創作活動にどのような影響を与えるかを検証するために，児童向けプログラミングワークショップで実験を行った．実験では，実験中の児童の PC 画面や，児童の様子の録画を行い，児童が分岐や反復の要素を理解し，創作活動へ繋げられたかを評価した．

1 立命館大学院 情報理工学研究科
2 立命館大学 情報理工学部

© 2020 Information Processing Society of Japan

2. プログラミングによる創作活動

本節では，児童向けプログラミングワークショップとはどのようなものか，そのようなワークショップで使われているビジュアルプログラミング言語である Scratch と Snap! について，また，既に実現されている可視化機能に対する本研究の特徴を述べる．

2.1 児童向けプログラミングワークショップ

現在，我々は NPO 法人スーパーサイエンスキッズ (以下 SSK) と協力し，児童向けプログラミングワークショップを行っている．このワークショップ は，講師 1 人とサポーター 3 人から 5 人，児童は多い時で 18 人で開催されている．行っているプログラミングワークショップの流れは以下の通りである．

講師は Scratch で，テーマに沿ったプログラムの作り方をプロジェクターを使ってスクリーンに映し出して説明をする．児童はこれを真似しながらプログラムを作っていく．プログラムの作り方や Scratch の操作方法が分からない場合は，近くにいるサポーターに質問することで問題を解決する．ある程度作品が完成すると，自由制作に入る．自由制作では，作ったプログラムを改変したりして，児童それぞれが好きなように創作活動を行う．ここでも，プログラムの作り方等が分からない場合は，サポーターに質問をする．最後には，スクリーンに児童一人一人が作ったプログラム映し出し，全員の前で発表する時間が設けられている．

このようなワークショップを実施してきた経験では，第 1 章でも述べたように，児童が思い通りのものを作り上げるのは困難であることが問題点として挙げられる．

2.2 創造的思考育成スパイラル

創造的思考を育成するプロセスモデルとして「創造的思考育成スパイラル」が M.Resnick によって提唱されている [3]．これは，「imagine(想像する)」「create(作る)」「play(遊ぶ)」「share(共有する)」「reflect(振り返る)」のプロセスを繰り返すことによって，創造的思考を育成していくというものである．「imagine」「create」「play」は作品の創作活動で行われるものである．また，「share」「reflect」は他者に自分で作った作品を発表するなどして行われる．

本研究では，実行部分を可視化することによって，児童が分岐や反復などを理解できるようになることを目指す．これにより，より自分が作りたいと思い描いているものを作品にできるようになり，「imagine」「create」をより促進することができる考える．

また，本研究では，創造的思考育成スパイラルや構築主義 [4] に基づいて開発されたプログラミング環境である Scratch を用いる．Scratch とは子供向けビジュアルプログラミング環境の一つであり，MIT メディアラボによって開発された

2.3 実行部分の可視化機能

Snap!とは Scratch と同じビジュアルプログラミング言語の一つである [5]．この言語では実行部分の可視化機能が実装されている．図 1 で赤丸で示すように，実行と同時に，例えば，「90 度回す」のブロックは，上にある「10 歩動かす」と同じ青色であるが，このブロックが今正に実行されていることが分かるように点灯する (色が薄くなる)．これにより，どの部分がいつ実行されているかを視覚的に理解することができる．

図 1 Snap!におけるブロックの点灯

2.4 本研究の特徴

前節で述べた Snap!の実行しているブロックを点灯させる機能を用いて，制御ブロックの動きを理解する際，児童は実行に合わせて点灯しているブロックと，スプライトの動作の 2 つの点に注目する必要がある．図 2 において赤丸で示したように，左側のスクリプトと，右側のスプライトの関係性を見なければ，制御ブロックの働きを理解することができない．しかし，これら両方に注目しながら，制御ブロックの働きを理解することは，児童にとっては困難と考える．

それに対して，本研究では実行しているブロックをテキスト形式でスプライトの吹き出しに出力する可視化方法をとる．これにより，児童は本機能を使用して，制御ブロックの働きを理解しようとする際に，注目しなければならない点はスプライト 1 点のみになる．このことは，Snap!のブロック点灯機能よりも児童にとって制御ブロックの働きをより理解しやすくなると考える．

© 2020 Information Processing Society of Japan

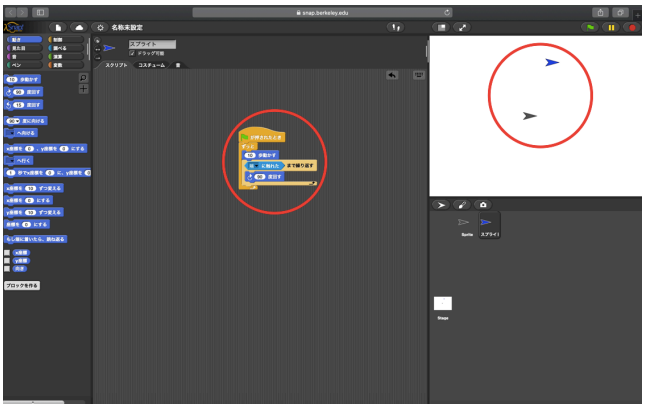

図 2　Snap!画面

3.　可視化機能

本節では，構築した可視化機能とその実装方法を述べる．

3.1　機能

3.1.1　プログラム実行部分の可視化

本研究におけるプログラム実行部分の可視化方法は，プログラムを組み込んだスプライトに対して，吹き出しを使って実行しているブロックをテキスト形式で出力する．実行ブロックが可視化されている例を図 3 に示す．このように，「90 度回す」というブロックが実行されているとすると，それによって動いているスプライトの吹き出しにも「90 度回す」と出力される．

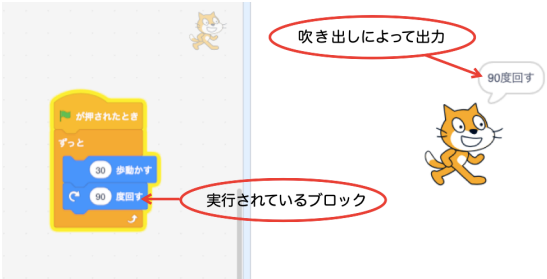

図 3　本研究における可視化方法

3.1.2　可視化するブロックの種類

2.2 節で述べたように，Scratch には様々なブロックの種類があるが，本機能はスプライトの動きに関わる「動きブロック」のみを可視化して，分岐や反復などの「制御ブロック」等は可視化しない．その理由は，動きブロックがいつどのように動いているかさえ分かれば，分岐や反復などの制御ブロックの働きを理解する上で児童にとっては，十分であると考えたからである．また，動きブロックのみを可視化することによって，制御ブロックを可視化するときに比べて，なぜ動きブロックが実行されるときとされないときがあるのかや，なぜ動きブロックが繰り返し実行されるのかなどをよく考えて制御ブロックの働きを理解しようとすると考える．

本研究では，さらに我々が実施しているプログラミングワークショップの内容を加味して，制御ブロックを理解できるようにする上で「〇〇を送る」「次のコスチュームにする」の 2 つのブロックの可視化が必要と考えたため，動きブロックではないが例外として可視化することとした．可視化するブロックの一覧を図 4 に，可視化しないブロックの一例を図 5 に示す．

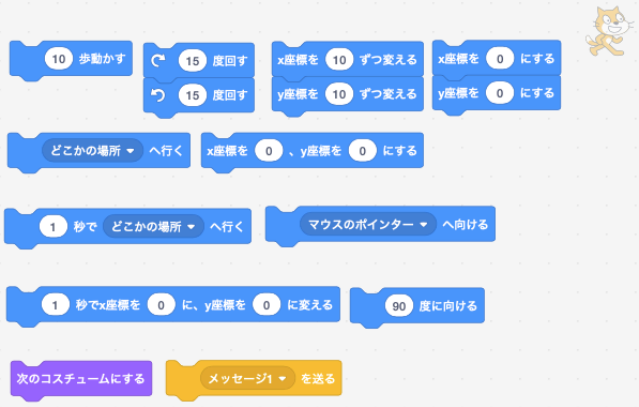

図 4　可視化するブロックの一覧

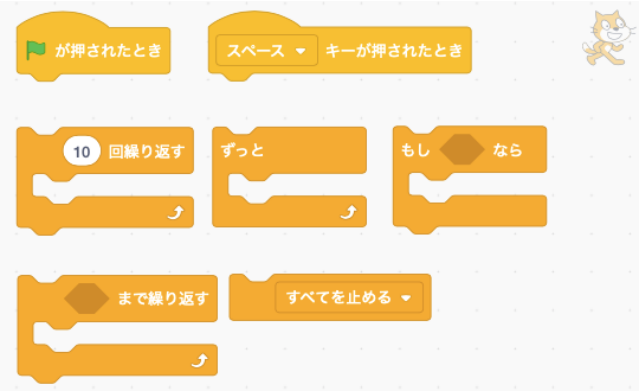

図 5　可視化しないブロックの一例

3.1.3　実行速度の調整

本機能ではプログラムの実行部分を可視化すると同時に，実行速度を遅くする．実行部分の可視化を行う時に，そのままでは実行速度が速すぎて，複数の動きブロックが連続して実行された場合に，吹き出しへの出力がすぐに切り替わってしまう．そのため，どのようにプログラムが遷移しているかを確認することができない．このことから，児童が制御ブロックがどのような働きをしているか把握するためには，実行速度を遅くし，実行されているブロックとスプライトの動きの関係を注意深く観察できるようにする機能が必要不可欠である．

© 2020 Information Processing Society of Japan

3.1.4　可視化機能の使用方法

プログラム実行部分の可視化と実行速度を遅くする 2 つの機能をブロックによって同時にオンとオフが切り替えられるようにする．Scratch の画面の左下をクリックし，追加するブロックの一覧から「NewBlocks」を選択する．すると，使用できるブロック一覧に「可視化機能」が追加される．それをクリックすると，図 6 のように，「オン」ブロックと「オフ」ブロックが出てくる．この 2 つの機能をそれぞれ押すことによって，本機能をオンにしたり，オフにすることができる．オンの時は，実行ブロックがスプライトの吹き出しに出力される．また，同時に実行速度が遅くなる．オフの時は，本機能の使用が解除され，実行ブロックがスプライトの吹き出しに出力されることはなく，実行速度も通常通りの速さである．

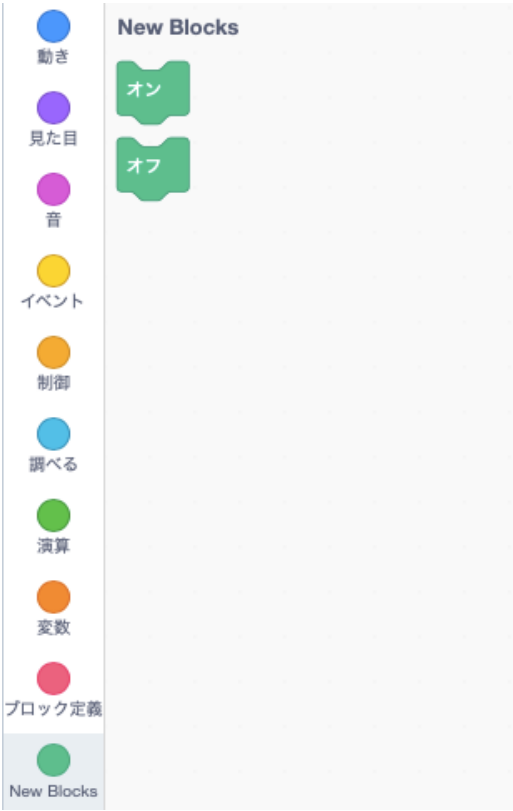

図 6　本機能のオンオフ画面

3.2　想定されるシーン

本機能を実際にプログラミングワークショップで使用する際，制御ブロックの動きを理解した時に想定されるシーンを以下に述べる．

図 7 では赤マルで示している「ストライクを送る」ブロックを，正しくは，ストライク判定をする制御ブロックの中に配置しなければならないが，制御ブロックの外側に配置してしまっている．これでは，ピッチャーがボールを投げた直後からストライクのメッセージを送るため，ストライク判定がキャッチャーへ到達する前に行われてしまう．児童はこのことに気付き，本機能を使用する．可視化されたプログラムを確認してみると，ボールがピッチャーに到達する前からボールの吹き出しによって「ストライクを送る」と「〇〇歩動かす」が交互に繰り返し実行されていること，及び，「ストラクを送る」は，その上にある分岐ブロックの「〇〇色 (キャッチャーミットの色) に触れた」ときにだけ実行されなければならないことに気づく．したがって，児童は，図 7 の矢印が示すように，ストライクの判定をしている「もし〇〇なら」ブロック内に移動させる．これにより，ボールがキャッチャーに到達した時に，1 度だけ「ストライクを送る」ブロックが実行されるようになる．

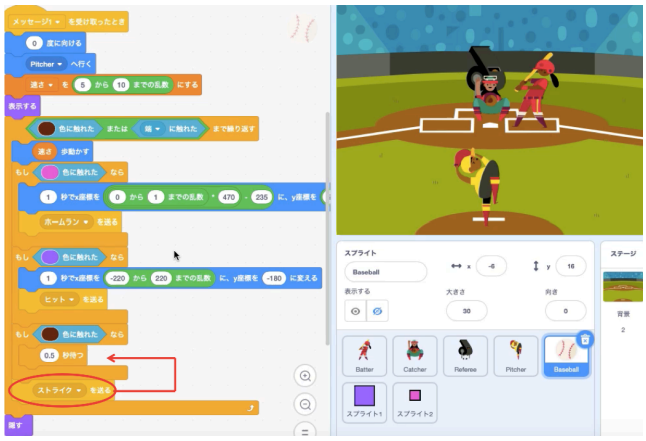

図 7　想定されるシーンの例

3.3　実装

本研究では Scratch のオープンソースを改変して可視化機能の実装を行った．本研究でコードを改変した部分は大きく分けて 2 点ある．1 つ目は，可視化するブロックそれぞれに，実行された時に，スプライトの吹き出しにブロックをテキストとして出力するコードを追加したことである．2 つ目は，実行速度を遅くするコードを追加したことである．実行速度を変化させるために，オンオフ切り替え時に Scratch 内で定義されているタイマーの時間を変更した．

4.　実験と結果

本章では，本機能を用いて実施した実験の内容と結果について述べる．

4.1　実験方法

本機能を利用することによって，児童の創作活動にどのような影響を与えるかを検証するために，NPO 法人スーパーサイエンスキッズが主催する児童向けプログラミングワークショップで実験を行った．本機能を利用することによって，児童が制御ブロックを理解し，創作活動へ繋げられたかを評価するため，実験中の児童の PC 画面と児童の

© 2020 Information Processing Society of Japan

様子の録画を行った．児童がプログラミングを行う端末には通常のワークショップで用いられているノートパソコンを使用した．それぞれのパソコンから，サーバにアクセスすることによって，本機能が追加されている Scratch を使用できる．また，パソコンそれぞれにあらかじめインストールしておいた画面録画ツールを起動し，PC 画面録画を行った．図 8 に実際に行ったワークショップの様子を示す．

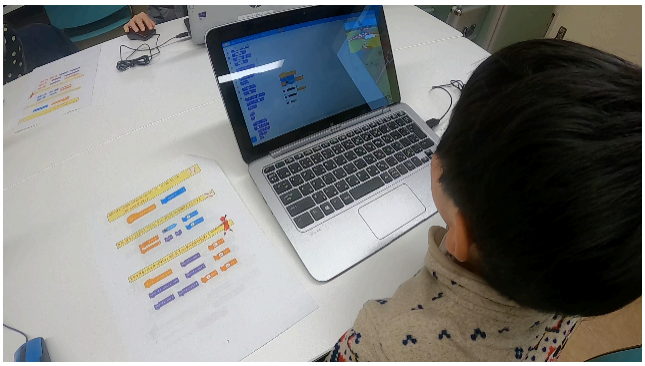

図 8　ワークショップの様子

4.2　ワークショップの設計

今回の実験では，野球ゲームを作るプログラミングワークショップを対象とした．今回作った野球ゲームを図 9 に示す．プレイヤーはバッターを操作し，ピッチャーが投げるボールのタイミングを見計らって，スペースキーを押してボールを打ち返す．ボールを打ち返す判定方法は，バッターがバットを振るタイミングで，判定をするための的が一瞬だけ表示される．的が一瞬表示された時に，ボールが的のどの部分に当たっているかを判定し，ヒットかホームランを決める．ボールが的に当たることなく，キャッチャーに到達すると，ストライクの判定となる．

図 9　野球ゲーム

今回実施したプログラミングワークショップの流れを以下に述べる．はじめに，児童に対してどのボタンを押せば可視化機能が使えるかなどの使い方を説明する．次に，図 10 に示したような野球ゲームの制作過程を複数のステップに分割した資料を一人一人に配布し，その資料についての説明をする．その後，配布した資料に沿って，児童それぞれが 1 時間 30 分程度，野球ゲームを作成する．最後に 1 時間 30 分程度，自由にこのゲームを拡張する．

今回の実験に関する情報を以下に示す．

実施日　12 月 8 日
場所　こども未来館 (京都府 京都市)
被験者　13 名 (小学 2 年〜6 年)

図 10　配布資料

4.3　実験結果と考察

本節では，評価実験時の児童の PC 画面録画や，児童の様子から得られた結果とそれから分かる本機能の評価と考察を述べる．

4.3.1　PC 画面録画による実験結果

実験後，児童全員の PC 画面録画から，本機能が児童にどのように使われて，どのような効果があったかを評価した．

被験者の児童全 13 人中 9 人が，個人によって使用回数にばらつきがあるものの，本機能を使用していたことが確認できた．また，実験中に合計 30 回本機能が使われていることが分かった．本機能を使ったことによる効果を以下に述べる．

4.3.2　制御ブロックの理解

本機能によって，児童の制御ブロックの理解につながっている場面が 3 回確認できた．その流れの一例を図 11，図 12 に示す．図 11 では赤丸で示している「ヒットを送る」ブロックが制御ブロックの外側に配置されている．これではボールの的への当たり判定に関係なく，ボールがキャッ

© 2020 Information Processing Society of Japan

チャーへ到達すると，ヒットのメッセージを送る．したがって，どんなボールでもヒットになってしまう．ある児童は，ヒットの判定をしているのが図 12 の赤丸で示している部分であるということを理解し，「ヒットを送る」ブロックを正しい位置に配置している様子が確認された．

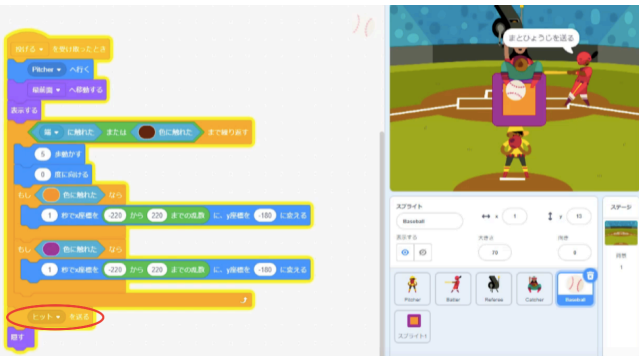

図 11　本機能使用前

図 12　本機能使用後

4.3.3　間違いの修正

本機能によって，児童が間違いの修正を行っている場面が 10 回確認できた．その流れの一例を図 13, 図 14 に示す．図 13 では赤丸で示したように「投げるを送る」ブロックの位置が制御ブロックの最後に配置してある．しかし，これではピッチャーの動作とボールが投げられるタイミングが不自然になる．児童はこのことを解決するために，本機能を使って「投げるを送る」ブロックの実行タイミングを調整し，図 14 の赤丸で示したようにピッチャーの動作とボールが投げられるタイミングが自然となるような配置に修正できていた．

全 30 回本機能が使われたうち，残りの 17 回については本機能によって，特に効果があったとは言えなかった．しかし，完成した動作を確認していると見られる場面も見受けられ，創作活動の補助となっていたことがうかがえる．

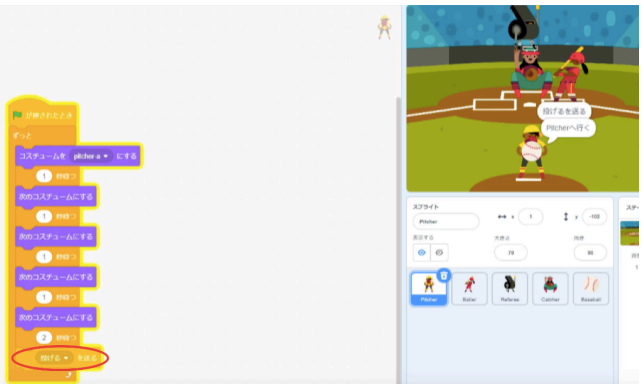

図 13　本機能使用前

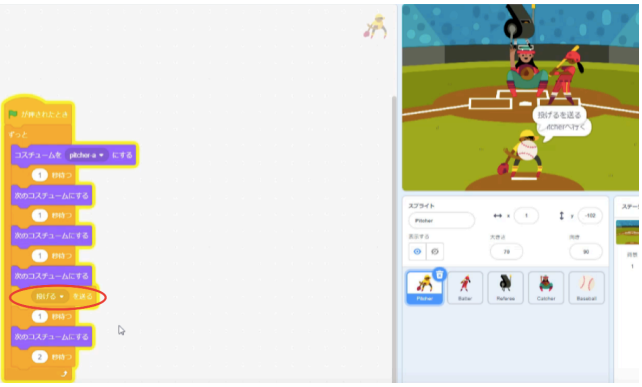

図 14　本機能使用後

4.3.4　ワークショップの様子

ブロックの動きが分からなくなったり，スプライトが自分の思い通りに動かなくなるなどの問題が発生した場合に，本来は児童が本機能を使って問題を解決するのを期待していたが，児童はまず先にサポーターに助けを求める場面が多く見受けられた．また，児童がサポーターによって，本機能を使い問題を解決するように勧められている場面も見受けられた．

4.3.5　考察

本機能が児童の創作活動の中で，制御ブロックの動きの理解に一定程度繋がったと考える．また本機能は，プログラムの簡単な間違いを修正するための補助もできていた．

一方で，本機能を使ってはいるものの，意味を成していないことが半数以上を占めていた．理由として考えられるのは，本機能を使うと，存在している全てのスプライトがそれぞれ実行しているブロックを吹き出しに出力するので，どのようにブロックが実行されているのか分かりにくくなってしまったと考える．これを解決するには，可視化させるスプライトを選択できるようにする必要がある．

また，ユーザインタフェースの面では，本機能を起動させるブロックが表示されるように，「可視化機能」を毎回選択しないといけないようになっていた．これにより児童が本機能を起動する際に手間がかかってしまう様子が見受け

© 2020 Information Processing Society of Japan

られた．そのため，普段プログラムを実行するために押す旗ボタンの近くに，本機能を有効にした上でプログラムを実行するボタンを配置することで，ユーザビリティの向上や使用回数の増加を期待することができる．

5. 終わりに

本研究では，プログラム実行部分の可視化機能を構築し，児童の制御ブロックの動きの理解に繋がるかを検証した．実際のプログラミングワークショップで実験を行った結果，児童の多くが本機能を使っていた．また，実際に児童の制御ブロックの理解に繋がっていたことが確認できた．他にも，本機能がプログラムの間違いを直すためにも機能していることが確認できた．それに加えて，本機能を使って，完成したプログラムの動作確認を行っていると考えられる場面もあった．

今後，本機能がより幅広い児童に対して有効に使われるようにするためには，本機能の使用方法や可視化するブロックの種類の見直し，可視化させる対象のスプライトを選択できる機能の追加などが必要であると考える．

参考文献

[1] 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），https://www.mext.go.jp/b_menu/shingi/chousa\\/shotou/122/attach/1372525.htm（参照 2020-01）.

[2] Resnick, Mitchel, et al:Scratch: programming for all,Communications of the ACM 52.11,pp.60-67,2009.

[3] Resnick, Mitchel:All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten,In Proceedings of the 6th ACM SIGCHI conference on Creativity cognition,pp.1-6,2007.

[4] Papert, Seymour:Constructionism: A New Opportunity for Elementary Science Education,MIT,Media Laboratory, Epistemology and Learning Group: National Science Foundation,1986.

[5] Price, Thomas W and Dong, Yihuan and Lipovac, Dragan:iSnap: towards intelligent tutoring in novice programming environments,Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education,pp.483-488,2017.

© 2020 Information Processing Society of Japan