ビジュアル型プログラミング言語による プログラム構築過程把握の試み

An Attempt to Understand the Program Construction Process in Visual Programming Language

山　田　耕太郎
YAMADA Kotaro

We have explored to understand how learners construct program by using one of visual programming languages, Google Blockly. It is possible to understand the process of constructing the program by collecting events of blocks that make instruction and analyzing those events. This paper describes the result that we have obtained a significant difference indicating the quality of trial-and-error between learners by analyzing events when university students who have never experienced programming have programmed.

1. はじめに

令和２年度からプログラミング的思考 [1] [2] の育成を狙いとしたプログラミング教育が小学校で必修化され，中学校では令和３年度から，高等学校では令和４年度から新しい学習指導要領に基づくプログラミング教育が実施されるようになった。中学校および高等学校のプログラミング教育について，教育の情報化の手引き－追補版の第３章 [3] で「プログラミング的思考を含む情報活用能力を育成していくことができるよう，各教科等の特質を生かし，教科等横断的な視点から教育課程の編成を図るとともに（以下略）」と記述されているように，プログラミング的思考は小学校段階以降のプログラミング教育との接続に必要な要素のひとつとなっている。

プログラミング的思考とはどのようなものであるのかについては，文部科学省が「自分が意図する一連の活動を実現するために，どのような動きの組合わせが必要であり，一つ一つの動きに対応した記号を，どのように組合わせたらいいのか，記号の組合わせをどのように改善していけば，より意図した活動に近づくのか，といったことを論理的に考えていく力」と定義している。つまりプログラミング的思考とは「物事を論理的に考える力」ということになるが，論理的な思考を積み上げていくことは小学生には敷居が高いと思われる。そこで重要になるのが「試行錯誤」である。

プログラミング教育における試行錯誤については，文部科学省が「学習指導要領に示すとおり，児童がプログラミングを「体験」し，自らが意図する動きを実現するために試行錯誤することが極めて重要となります」や「児童は試行錯誤を繰り返しながら自分が考える動作の実現を目指しますが，思い付きや当てずっぽうで命令の組み合わせを変えるのではなく，うまくいかなかった場合には，どこが間違っていたのかを考え，修正や改善を行い，その結果を確かめなるなど，論理的に考えさせることが大切です」と述べていることからも，その重要性が分かる。しかし，児童の試行錯誤が「ある意図に基づくもの」であるのか「思い付きや当てずっぽう」であるのかは最終的な成果物を見ただけでは分からない。従って，学習者が行う試行錯誤の質や意図を把握し，それを教員が授業改善や学習支援に活かすためには成果物に至るまでの過程を詳細

に知る必要がある。

筆者はこれまで，大学生の論理的思考力育成を目的としてゲームやパズルを使い，その手順や必勝戦略を考える授業実践[4][5]を行ってきたが，学習者の試行錯誤を探るまでには至っていない。しかし近年，プログラミング教育において学習者のプログラミング過程を操作ログとして収取し，試行錯誤を含めた思考過程を把握しようとする研究が行われ始めている[6][7][8][9]。これらの研究は，分割されてランダムに並べられたプログラムコードや文章を学習者が並べ替える過程を操作ログとして収集・分析し，プログラミングの思考過程を探ろうとするものである。これらの研究では学習者の思考過程に踏み込む有用な結果が得られているが，テキスト型のプログラミング言語を使っているため初等教育への導入は難しいと思われる。そこで筆者は，ビジュアル型プログラミング言語であるGoogle Blockly[10]を利用して学習者のプログラム構築過程を把握する試みを行い，試行錯誤の質に有意差があることを示唆する結果を得たので報告する。

2. Google Blockly

Google BlocklyはGoogle社が開発したビジュアルプログラミングのためのライブラリであり，Apache License 2.0に基づくオープンソースとして公開されているため，独自のWebアプリケーションの開発や実装に使うことができる。操作性はScratchと同様であり，PC上でのマウス操作やスマホ・タブレット端末上でのタッチ操作で命令ブロックを組み上げてプログラミングすることができる。またGoogle Blocklyは，命令ブロックへの操作に応じてイベントメッセージを発する仕様となっている。このイベントメッセージには，命令ブロックの種類や行われた操作（ブロックの作成・削除・移動など），属性（座標や親ブロックのUUIDなど）が含まれているため，これらのイベントメッセージを時系列データとして収集することにより，学習者が行ったプログラム構築過程を把握・分析することができる。

3. 迷路ゲームとイベントメッセージ

筆者はBlockly Games[11]のひとつとして公開されている迷路ゲーム（Maze）を研究用のWebサイトに移植し，学習者が迷路ゲームを解く過程のイベントメッセージを収集するシステムを構築した。

迷路ゲームではキャラクタを移動させるプログラムを作り，そのプログラムでキャラクタがゴールまで到達できれば次のレベルに進むことができる。ゲームはスタートからゴールまでが一直線の迷路を解くレベル１から始まり，迷路の複雑さに応じてレベル10（図１）まで用意されている。レベル１と２では「まっすぐ進む」「左を向く」「右を向く」の３種類の命令ブロック（図２）を使うことができ，プレイヤーは適切な命令ブロックを選んでプログラミングを行う。これらのレベルでは順次処理のみのプログラムを作成することになるが，レベル３からは「まで繰り返す」という反復処理の命令ブロック（図３），レベル６からは「もしまっすぐ進めるなら」というif文の分岐処理，レベル10では「もしまっすぐ進めるなら○○，それ以外△△」というif-else文の分岐処理に相当する命令ブロック（図４）が使えるようになる。また，レベル３以降は使える命令ブロックの数に上限が設けられるため，全てのレベルを順次処理のみでプログラムすることはできない。

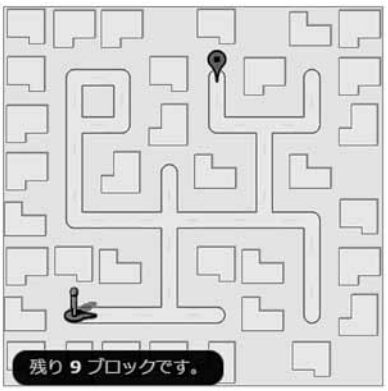

図1　レベル10の迷路

図2　順次処理の命令ブロック

図3　反復処理の命令ブロック

図4　分岐処理の命令ブロック

命令ブロックへの操作で発せられたイベントメッセージの例を図5に示す。これは迷路ゲームのブロックパレットから「まっすぐ進む」の命令ブロックをワークスペースにドラッグアンドドロップしたときのイベントメッセージである。ドラッグアンドドロップの一連の操作で4つのイベントが発生しており，それぞれのイベントメッセージをjson形式でまとめている。1つ目のイベントは命令ブロックの“作成”（create）に伴うものであり，そのブロックに付与されたUUID（blockId）やブロックを定義するXMLデータなどの属性が与えられている。2つ目のイベントは命令ブロックの“選択”（selected）に伴うものであり，ブロックの操作対象がこれまでのブロック（oldElementId）から新たなブロック（newElementId）に移ったことが分かる。3つ目と4つ目のイベントは“移動”（move）に伴うものであるが，最初の“移動”はユーザがブロックを置いた（ドロップした）ときのものであり，それに続く“移動”はブロックがワークスペース上のグリッドに吸着されたことによるものである。これら一連の動作により，位置座標（newCoordinate）の値が10の倍数に丸められていることが分かる。


```
{
  "type": "create",
  "group": "iEMeX=q2%eqF!fZqfj|v",
  "blockId": "X|he~m4k`$t$sYJ1jCJ1",
  "xml": "<block
xmlns=https://developers.google.com/blockly/xml
type=maze_moveForward id=X|he~m4k`$t$sYJ1jCJ1
x=-138 y=8></block>",
  "ids": [
    "X|he~m4k`$t$sYJ1jCJ1"
  ]
}
{
  "type": "selected",
  "group": "iEMeX=q2%eqF!fZqfj|v",
  "oldElementId": "D!5.Vvy4)_.n=UeWr2py",
  "newElementId": "X|he~m4k`$t$sYJ1jCJ1"
}
{
  "type": "move",
  "group": "iEMeX=q2%eqF!fZqfj|v",
  "blockId": "X|he~m4k`$t$sYJ1jCJ1",
  "newCoordinate": "196,226"
}
{
  "type": "move",
  "group": "iEMeX=q2%eqF!fZqfj|v",
  "blockId": "X|he~m4k`$t$sYJ1jCJ1",
  "newCoordinate": "190,230"
}
```


図5　イベントメッセージの例

上記の“移動”は，命令ブロックをワークスペース上に単独で置いたときのものであるが，順次処理のために他の命令ブロックに連結するように置いた場合は図6のようにnewCoordinateではなくnewParentId属性が付与されるため，命令ブロック相互の接続関係が分かる。また反復処理や分岐処理の命令ブロックに嵌めた場合は図7のようにnewInputName属性が付与される。図7でnewInputNameの属性値が“DO”となっているのは反復処理や分岐処理の「実行」部分に命令ブロックを嵌めたことを示しており，「それ以外」の部分に嵌めた場合は“ELSE”が属性値となる。


```
{
  "type": "move",
  "group": "k{@gbD^9FGkWoE{wud/E",
  "blockId": "%g`D{fF@{d6Gl_3.2X*n",
  "newParentId": "0MyBd6IgLt:1`E8F^RSo"
}
```


図6　イベントメッセージ（ブロック連結時）


```
{
  "type": "move",
  "group": ";-Xfz}`n3iF`vbSbYxrr",
  "blockId": "y:L|wbyF{59IV~RK2ps$",
  "newParentId": "wK$w(|}LrjsL2yynI*ev",
  "newInputName": "DO"
}
```


図7　イベントメッセージ（ブロック嵌込時）

以上のように，イベントメッセージとその属性を時系列でトレースすることによって学習者がどのような手順で命令ブロックを組み合わせていったのか，すなわち学習者がどのような順序でどのような操作を行っていったのかを詳細に把握することができる。

4. イベントメッセージの収集

イベントメッセージの収集は，本学で筆者の授業を履修している１年生21名を対象に実施した。学生が所属する学科は情報系や理系の学科ではないため，ほとんどの学生はプログラミングの経験がない。また，対象学生が履修している筆者の授業も，ゲームやパズルをアンプラグド的に解きながら論理的思考を身に付ける内容であり，プログラミングは行っていない。

イベントメッセージ収集の当日（６月21日）は，まず初めに紙面上に印刷された一般的な迷路（後に解く迷路ゲームとは無関係のもの）を学生に解かせた。このとき，どのように迷路を解いたのかを確認すると全員が「スタートからゴールまでを目で追いながら解いた」と回答したが，これは紙上で迷路を解く場合にはごく自然な方法であるため，予想通りの結果であった。しかしこの方法は迷路のコース全体が俯瞰できる紙面上では通用するが，自分が実際に迷路に入り込んだ場合はゴールの方向すら分からない状況となるため，ゴールに辿り着くためには別の方法を使わなければならない。そのことを学生に指摘した上で右手法と左手法を紹介し，必ずしも最短経路とはならないが必ずゴールまで到達できることを確認した（授業の本来の趣旨からは，右手法や左手法の発想が学生側から出るよう授業設計すべきであるが，当時学内で行われていた感染症拡大防止対策による授業上の制約を考慮した結果，天下り的に導入せざるを得なかった）。

迷路ゲームを解く前に右手法や左手法を学生に紹介した理由は，迷路ゲームのレベル10でそのようなアルゴリズムの発想が求められるためである。図８はBlockly Gamesの公式サイトで迷路ゲーム（Maze）のレベル10で表示されるメッセージであるが，これは明らかに左手法のアルゴリズムの適用を促している。ただ後に示すように，右手法や左手法のアルゴリズムを忠実にプログラム実装していない場合でもゴールまで到達できる場合があるため，本研究用のサイトでは図８のようなメッセージは表示しないようにしている。

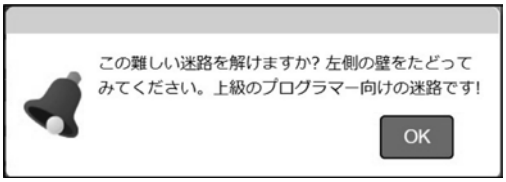

図８　左手法の適用を促すメッセージ

紙面上での迷路の解法を一通り説明した後，PC画面上で迷路ゲームを解いてもらった。先述したように学生はプログラミングの経験がなく，ビジュアル型プログラミング言語の操作も初めてである。そのため，プログラミングの初心者が初見でどのような操作を行うのか，ということを今回の分析の主目的としたため，学生の特定につながるデータの収集は意図的に行わなかった。

5. 結果と考察

本研究では，イベントメッセージの収集において学生の特定につながるデータは収集していないが，研究用サイトの迷路ゲームにアクセスしたクライアントのブラウザにランダムな文字列のIDを付与し，同一IDからのイベントメッセージを時系列データとしてまとめたものを同一学生の操作ログと見なしている。また学生には操作方法や命令ブロックの意味を実演を交えて説明し，レベル４までは正解のプログラムも示した上でレベル５以降を自力でプログラムするよう指示した。従って21名の各学生について，レベル１からレベル10までの迷路を解くためにどのような操作手順でどのようなプログラムを作ったのかを全て追跡することができる。

自力でプログラムするよう指示した後も操作方法に悩む様子や質問が出ることもなく，イベントメッセージの分析からレベル９までは21名全員がクリアできていたことが分かった。しかしレベル10をクリアした学生は21名中12名であった。ここで，レベル10の「クリア」とは，Blockly Gamesの公式サイトと同じ判定基準で判断しており，必ずしも左手法のアルゴリズムを実装したプログラムでなくても，ゴールまで到達できればクリアとなる。従って，レベル10をクリアできるプログラムはいくつもある。

レベル10をクリアした学生とクリアできなかった学生について，レベル10のみのイベントメッセージ数を比較すると，クリアした学生は平均485個だったのに対しクリアできなかった学生は平均1,248個であり，t検定で有意な差が見られた（$p<0.05$）。この結果は，クリアできた学生は何らかの意図や方針に基づいてプログラムしているが，クリアできなかった学生は思い付きや当てずっぽうによる無駄な操作が多く，両者の間で試行錯誤の質に差が生じていることを示唆していると考えられる。クリアできなかった学生がどのような躓き方をしているのかについては，鋭意分析を行っているところである。

次に，レベル10をクリアしたある学生のイベントメッセージを時系列で読み取り，プログラム構築過程を詳細に再現できることを示す。

まず最初に行われたのは，ゲーム開始時にデフォルトでワークスペース上に置かれている「まっすぐ進む」の命令ブロックを“移動”させる操作であった。このときのイベントメッセージと命令ブロックの状態を図９に示す。

次に行われたのは，分岐処理の命令ブロックを“作成”して，図９の「まっすぐ進む」の命令ブロックを嵌める操作であった（図10）。

続いて分岐処理の条件を「もしまっすぐ進めるなら」から「もし左に進めるなら」に“変更”した（図11）。


```
{
  "type": "move",
  "group": "bx$/UBjbgHDZiuY!_sH_",
  "blockId": "})#91is,:Sgg/fYJByoD",
  "newCoordinate": "127,105"
}
{
  "type": "move",
  "group": "bx$/UBjbgHDZiuY!_sH_",
  "blockId": "})#91is,:Sgg/fYJByoD",
  "newCoordinate": "130,110"
}
```


図９ “移動”時のメッセージ（上）とプログラム（下）


```
{
  "type": "create",
  "group": ".(_(y!T9@{M^-UraIiB%",
  "blockId": ".)Gc;{Sa{qsU*RF%Y:Ph",
  "xml": "<block
xmlns=https://developers.google.com/blockly/xml
type=maze_if id=.)Gc;{Sa{qsU*RF%Y:Ph x=-218
y=278><field
name=DIR>isPathForward</field></block>",
  "ids": [
    ".)Gc;{Sa{qsU*RF%Y:Ph"
  ]
}
{
  "type": "selected",
  "group": ".(_(y!T9@{M^-UraIiB%",
  "oldElementId": "TLylEubACz$_.NV(iC^Y",
  "newElementId": ".)Gc;{Sa{qsU*RF%Y:Ph"
}
{
  "type": "move",
  "group": ".(_(y!T9@{M^-UraIiB%",
  "blockId": ".)Gc;{Sa{qsU*RF%Y:Ph",
  "newCoordinate": "90,83"
}
{
  "type": "move",
  "group": ".(_(y!T9@{M^-UraIiB%",
  "blockId": "})#91is,:Sgg/fYJByoD",
  "newParentId": ".)Gc;{Sa{qsU*RF%Y:Ph",
  "newInputName": "DO"
}
{
  "type": "move",
  "group": ".(_(y!T9@{M^-UraIiB%",
  "blockId": ".)Gc;{Sa{qsU*RF%Y:Ph",
  "newCoordinate": "90,90"
}
```


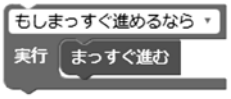

図10 “作成”時のメッセージ（上）とプログラム（下）


```
{
  "type": "change",
  "blockId": ".)Gc;{Sa{qsU*RF%Y:Ph",
  "element": "field",
  "name": "DIR",
  "oldValue": "isPathForward",
  "newValue": "isPathLeft"
}
```


図11 “変更”時のメッセージ（上）とプログラム（下）

以降はイベントメッセージの図示を省略し，ゲームクリアに至るまでのプログラム構築過程を順に示す。まず「左を向く」ブロックを挿入し（図12），次にif-else文の分岐処理ブロックを作成して図12に示すプログラムの下部に連結した（図13）。その分岐処理ブロックの「実行」部分に「まっすぐ進む」ブロックを，

「それ以外」部分に「左を向く」ブロックをそれぞれ新たに追加して嵌め込んだ（図14・図15）。そして，ここまでできあがったプログラム全体を反復処理ブロックに嵌め込み（図16），最初のプログラム実行を行った。

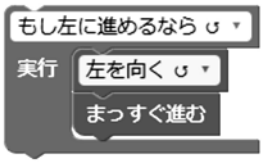

図12 「左を向く」ブロックの挿入

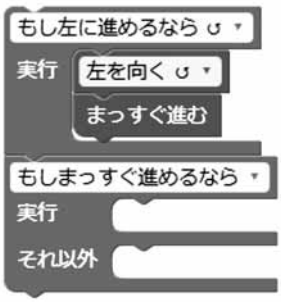

図13 分岐処理（if-else）ブロックの追加

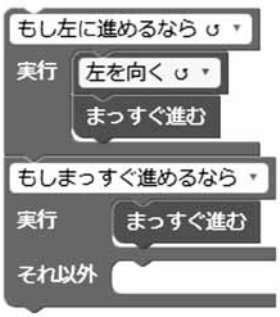

図14 「まっすぐ進む」ブロックの追加

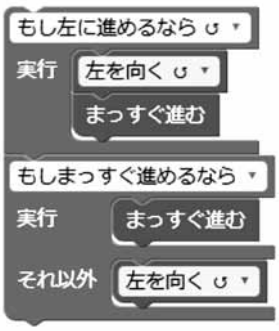

図15 「左を向く」ブロックの追加

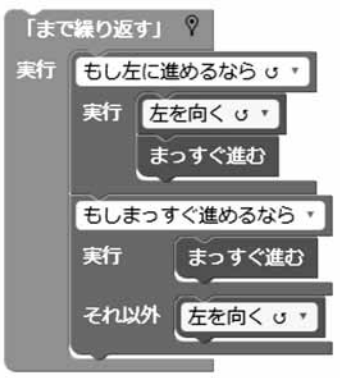

図16 反復処理ブロックの追加

図16のプログラムからは，学習者が左手法を意識していることがわかるが，左手法であっても右を向かなければならない場合がある。しかしここまでの構築過程を見ると，“左”に意識が集中してしまっているためか，右を向く場面の処理に思い至っていないように見える。この学習者はプログラムを実行してみてうまくいかないことに気付き，この後プログラムを修正している。プログラムの修正過程は次の通りである。

修正作業では，まず最初に図16の「もし左に進めるなら」を「もしまっすぐ進めるなら」という条件への変更を行い（図17），その分岐処理ブロックの「実行」部分に嵌っていた「左を向く」と「まっすぐ進む」の2つのブロックをペアのままとめて削除（図18）して「まっすぐ進む」ブロックへの差し替えを行った（図19）。

図17 条件の変更

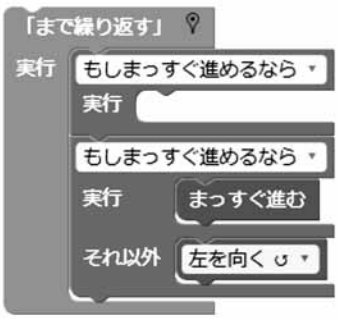

図18　ブロックの削除

図19 「まっすぐ進む」ブロックへの差し替え

次にif-else文の分岐処理ブロックの条件を「もしまっすぐ進めるなら」から「もし左に進めるなら」へ変更し(図20)，そのブロックの「それ以外」部分に嵌っていた「左を向く」ブロックを「実行」部分に嵌っている「まっすぐ進む」ブロックの上部に移動して挿入した（図21)。この移動と挿入によって空いた「それ以外」部分に「右を向く」ブロックを新たに追加して嵌め込み(図22)，再びプログラムの実行を行ってゲームクリアとなった。

図20　分岐処理（if-else）ブロックの条件変更

図21 「左を向く」ブロックの移動と挿入

図22 「右を向く」ブロックの追加

図22のプログラムは左手法を忠実に実装したものにはなっておらず，途中でキャラクタが無駄な動きをして左手が壁から離れる状況が生じたりするが，最終的にゴールまで到達できるためゲームクリアとなる。

以上見てきたように，イベントメッセージを読み取ることでプログラム構築過程の全ての挙動を再現できることが分かった。ここで再現した学習者の例からは，最初のプログラム実行でうまくゴールに到達できなかったことで条件が不足していることに気付き，修正を加えてゲームがクリアできたことがよく分かる。また全ての過程で，命令ブロックの作成や移動，削除，修正等が詳細に把握できるため，例えば途中までできたプログラムを全て削除してやり直したとしても，削除したプログラムの構築過程と，新たなプログラムの構築過程とを比較したり，学習者がどのタイミングで間違いや修正の必要性に気付いたのかを分析することができる。このような分析は最終的な成果物からは行えないため，学習者の試行錯誤の質や意図を明らかにするために非常に重要である。

6. まとめと今後の課題

本稿では，ビジュアル型プログラミング言語であるGoogle Blocklyのイベントメッセージを利用してプログラムの構築過程を把握できることを報告し，プログラミング学習者がプログラム構築の際に行っている試行錯誤や思考過程の分析が行えることを示した。

イベントメッセージの収集は，本学で筆者の授業を履修している21名の学生を対象に行った。学生はプログラミングの経験がなく，ビジュアル型プログラミング言語の操作も初めてであったが，若干の操作説明を行った後は自力でプログラミングを進めることができた。小学校でのプログラミング教育はScratchに代表されるビジュアル型プログラミング言語の利用が一般的であるため，本研究で行ったイベントメッセージの収集と分析は，初等教育から高等教育まで幅広く適用可能である。

プログラミングの際の試行錯誤について，Blockly Gamesのひとつである迷路ゲームのレベル10のプログラミングで発生したイベントメッセージ数を調べたところ，レベル10をクリアした学生の平均イベントメッセージ数は，クリアできなかった学生のそれと比較して有意に少ないことが示された。この結果は，ある意図に基づいて作業を進めている学習者と，思い付きや当てずっぽうで闇雲に作業している学習者とを試行錯誤の質で分類できることを示していると考えられる。

本稿ではプログラミングの初心者である大学生のデータのみで考察を行ったが，今後はプログラミング経験者や小学生の操作ログなども収集して分析を行い，プログラミングを行う際の思考過程や試行錯誤の詳細を明らかにすることが必要であると考えている。

謝辞

本研究は科研費（基盤研究(C)課題番号18K02921）の助成を受けている。

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）」文部科学省 https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm

[2]「小学校プログラミング教育の手引（第三版）」文部科学省 https://www.mext.go.jp/content/20200218-mxt_jogai02-100003171_002.pdf

[3]「教育の情報化の手引-追補版 第3章」文部科学省 https://www.mext.go.jp/content/20200608-mxt_jogai01-000003284_004.pdf

[4] 山田耕太郎，有吉優菜「数理パズルを使ったアルゴリズム教育の実践と評価」比治山大学紀要第24号（2017）pp.67-73.

[5] 山田耕太郎「ボードゲームを使ったアルゴリズム教育の実践と評価」比治山大学現代文化学部紀要第25号（2018）pp.75-80.

[6] 矢野里奈，谷岡広樹，松浦健二，佐野雅彦，上田哲史「小学生のプログラミング的思考力の定量的測定とその分析」情報処理学会研究報告 教育学習支援情報システム（CLE）Vol.2020-CLE-30 No.22（2020）pp.1-4.

[7] 山口琢，大場みち子「編集操作の時間的共起分析の提案」情報処理学会研究報告 コンピュータと教育（CE）Vol.2019-CE-151 No.9（2019）pp.1-7.

[8] 中村陽太，大場みち子，山口琢，伊藤恵「学習進度に対応するパズルを利用したプログラミング思考過程の分析」情報処理学会研究報告 研究報告コンピュータと教育（CE）Vol.2019-CE-151 No.1（2019）pp.1-9.

[9] 藤原亮，山口琢，大場みち子「パズルを利用したプログラミング学習教材における操作パターンの特徴抽出」情報処理学会研究報告 コンピュータと教育（CE）Vol.2020-CE-154 No.10（2020）pp.1-6.

[10] Google Blockly https://developers.google.com/blockly

[11] Blockly Games https://blockly.games/

〈キーワード〉
ビジュアル型プログラミング言語，Google Blockly，プログラミング的思考，試行錯誤，プログラム構築過程

山田耕太郎（現代文化学部マスコミュニケーション学科）

（2022. 10. 31　受理）