MR Code Weaver: 投影型ミクストリアリティによるタンジブルなプログラミング学習ツール

坂本恋[†1]　大島登志一[†1]

概要：本研究では，小学生以下の子どもがプログラミングを始める際の最初の足がかりとなるプログラミング学習ツールを検討する．このツールは，物理的な形状を持ったパーツを直接手で組み合わせることでプログラミングを行うことができるタンジブルな（触れられる）ユーザインタフェースをもつ．これにより，一般的なプログラミング言語における学習の入門段階で障壁となる，キーボードを介した文字入力や，使用するプログラミング言語における厳密な構文規則の要求，複数人での共同的な学習の難しさなどの問題の解決を試みる．

MR Code Weaver: Programming Educational Tool with Physical Mixed Reality Interface

REN SAKAMOTO[†1]　TOSHIKAZU OHSHIMA[†1]

Abstract: We propose a new tool for programming education for children. We focus on teaching common feature of programming, which are sequence, selection and iteration. We adopt physical user interface by using mixed reality in order to avoid difficulty of coding with a keyboard and/or studying about syntax and notation of programming language, which are not essential for novice of programming. The system provide Papert's Logo-like graphical feedback as result of execution of program in order to show apparently if the program is correct or not.

1. はじめに

情報技術や輸送技術の発展にともなう社会の IT 化とグローバル化により，子どものころからのプログラミング学習が注目されている．平成 29（西暦 2017）年 3 月に公示された，平成 32（西暦 2020）年 4 月 1 日より施行される小学校学習指導要領[3]では，「児童がプログラミングを体験しながら，コンピュータに意図した処理を行わせるために必要な論理的思考力を身に付けるための学習活動」を計画的に実施するようにと明記され，プログラミング学習が必修となることが決まった．

しかし，プログラミングを小学校で学習するためには解決しなければならない問題点も多い．例えば，一般的なプログラミング言語におけるキーボード入力はそれ自体がスキルを要するため，入門段階では大きな負荷となる．その結果，子どもの学習意欲を減退させ，学習の継続から遠ざけてしまう．こうした問題点は，プログラミング学習ツールの改善によって克服されるべきである．

本研究では，プログラミング言語の構文規則ではなく，プログラミングに普遍的な概念を学ぶことができる学習ツールの開発を目的とする．それに加え，本研究を進めていくうえでは，先行研究が取り組んでいきたような，子どもの興味を持続させるプログラミングコンテンツを擁していることも必須の要件と捉えて開発を行う．

今回開発したプログラミング学習ツールでは，プログラミング操作をタンジブルユーザインタフェースを介して行い，プログラム実行結果の提示をプロジェクションによるミクストリアリティ表現によって行った．タンジブルユーザインタフェースを採用することによってキーボード入力を排し，さらに複数人での共同作業やディスカッションを行うことも可能にする．

2. 関連研究

本研究のプログラミング学習ツールが目指す方針を以下に三つ挙げる．

（1）実行結果がグラフィカルである
（2）プログラミング操作がグラフィカルである
（3）手で触れられる物体である

これらに関連する先行研究を以下に示す．

2.1 図形を描くプログラミング言語

子どもがプログラミングを学ぶためには，まずプログラミングにたいして興味を持ち，学習を継続する必要がある．そのため，プログラムの実行結果は子どもにとっても分かりやすいものでなければならない．MIT の Papert らは，子どもがプログラミングを学ぶことのできる言語として Logo[2][3]を開発した．これは，図 1 のような画面上に描か

†1 立命館大学映像学部
College of Image Arts and Sciences, Ritsumeikan University

れたタートル（亀）をプログラムによって動かし，その軌跡によって図形を描くことができるプログラミング言語である．プログラムは「前へ進め」「右へ回れ」といった命令から成り，そのプログラミングはキーボード入力によって行われる．

本研究ではこの Logo を参考に，キャラクタを命令によって動かし，その軌跡によって図形を描くプロトタイプを制作した．

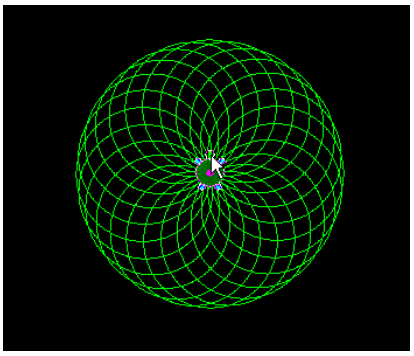

図 1　Logo の実行結果

2.2 キーボード入力が不要なプログラミング言語

2.1 で挙げた Logo はキーボードによる文字入力を必要とした．MIT Media Lab の Lifelong Kindergarten Group は，命令を表す画面上のブロックをマウスポインタによって組み合わせてプログラミングを行うことで，キーボードによる文字入力をほとんど不要にしたプログラミング言語 Scratch[4]を開発した．これによりユーザはキャラクタアニメーションを作ったり，インタラクティブなコンテンツを作ったりできる．Scratch のグラフィカルユーザインタフェースを図 2 に示す．このような，キーボードによる文字入力を必要としないプログラミング言語はビジュアルプログラミング言語と呼ばれ，Scratch のほかにも数多く存在する．プログラミングへの敷居を低くすることができるため，子どものプログラミング学習の現在の主流となっている．

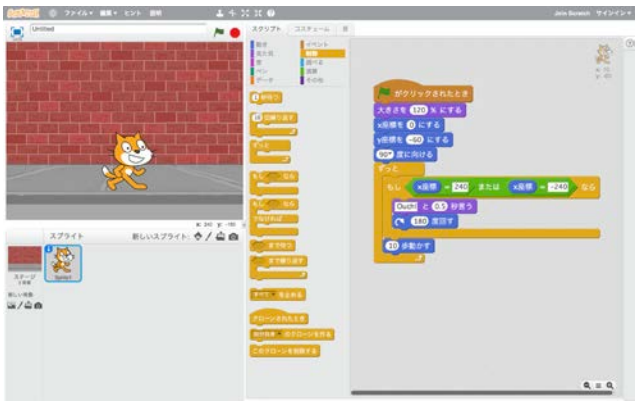

図 2　Scratch のグラフィカルユーザインタフェース

2.3 プログラム可能な玩具

子どもの創造性と思考力を育む手段としてプログラミングを用いたものは玩具にもある．LEGO MINDSTORMS[5]は，インテリジェントブロックと呼ばれる頭脳部品に，通常の LEGO ブロックやモータ，センサなどを組み合わせてロボットを組み立てることができる．組み立てたロボットは，2.2 で挙げた Scratch のような専用のビジュアルプログラミング言語や，他のさまざまな言語によってプログラムし，動かして遊ぶことができる．

3. システムデザイン

3.1 想定するプログラミング内容

今回は小学生程度の子どもを対象とし，プログラミングにおける最も基礎的な考え方を身につけることを目標とする．本稿ではその基礎的な考え方を，一つ一つ順番に命令を実行していく「逐次処理」，条件に応じて実行する処理を分岐させる「条件分岐」，条件のもとで同じ処理を繰り返す「反復処理」の三つと定義した．

それらを学習するためのコンテンツとして，2.1 で関連研究に挙げた Logo を参考に，キャラクタを動かしてその軌跡によって図形を描くコンテンツを制作した．プログラムで描く図形の例を図 3 に示す．Logo は，実行結果が期待通りなのか(そうでない場合にはどこが間違っていたのか)を分かりやすい形でフィードバックすることができるため，実行結果の確認が容易である．また，間違いを修正（デバッグ）する上でも，ユーザは自分の身体でタートルと同じ動きをすれば良いため，プログラミングを学び始めたばかりの子どもにも取り組みやすいコンテンツといえる．

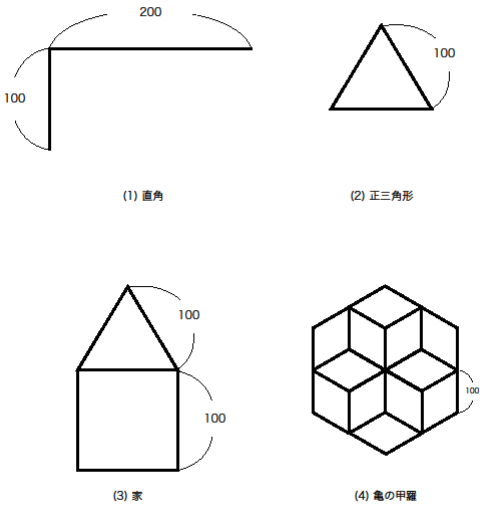

図 3　プログラムで描く図形の例

3.2 ユーザインタフェースのデザイン

本研究ではプログラムを構成する一つ一つの命令や，それに関わる数値を物理的なタイルに置き換え，手で直接触れながらプログラミングを学ぶことができるタンジブルユ

ーザインタフェースを採用した．ユーザはタイルをテーブル上に配置し，命令を実行したい順番に繋いでいく．また，すでに繋いであるタイル同士の間に，後からタイルを追加することも可能にした．これにより，プログラムのバグを修正したり，プログラムを発展させたりといった柔軟な学習が可能となる．

3.3 プログラムのモデル

本研究では，ノイマン型コンピュータによる逐次実行型のプログラムを想定している．すなわち，開始と終了が定義されていて，開始から終了にいたるまで順序立てて実行されるものである．本研究では，プログラムの開始と終了を，それぞれ開始命令と終了命令として割り当てる．開始命令に任意の命令を追加していくことでプログラミングを行い，最後に終了命令を追加することでプログラムが実行される．

3.4 プログラムの構成要素

プログラムの「命令」は，「オペコード」と「オペランド」から構成される．オペコードは，命令の種類を表すものであり，オペランドは，命令に関する数値を表すものである．

オペランドの内容はオペコードによって異なる．例えば，キャラクタを前進させるオペコードには，距離を表すオペランドを組み合わせることができる．

本システムでは，以下のような種類の命令を使用する．

（1）開始と終了
プログラムの開始と終了を，それぞれ「IN」「OUT」命令とする．オペランドは組み合わされない．

（2）条件分岐
条件分岐を「IF」命令とする．真か偽の値をとるオペランドが組み合わされる．

（3）反復処理
反復処理の開始と終了を，それぞれ「REPEAT BEGIN」「REPEAT END」命令とする．REPEAT BEGIN には反復する回数の値をとるオペランドが組み合わされ，REPEAT END にはオペランドは組み合わされない．

（4）前進
キャラクタの前進を「FORWARD」命令とする．前進する距離の値をとるオペランドが組み合わされる．

（5）右旋回と左旋回
キャラクタの右旋回と左旋回をそれぞれ「RIGHT」「LEFT」命令とする．マイナスとプラスがプログラムの本質ではないため，右旋回と左旋回とした．回転角の値をとるオペランドが組み合わされる．

3.5 プログラムの編集と実行

図 4 に示すように，本システムは開始時にプログラム編集モードとなる．ここでユーザはオペコードおよびオペランドを入力する．ユーザは，プログラムの開始点である IN 命令に続けて，オペコードを追加し，オペコードには，それがとりうるオペランドを組み合わせる．

プログラムに OUT 命令が追加されると，システムは実行モードへと切り替わる．実行前処理でエラーの有無をチェックし，エラーが見つかれば実行されずに編集モードへ切り替わる．エラーが見つからなければ開始命令から順に処理され，終了命令まで処理されると再び編集モードへ切り替わる．

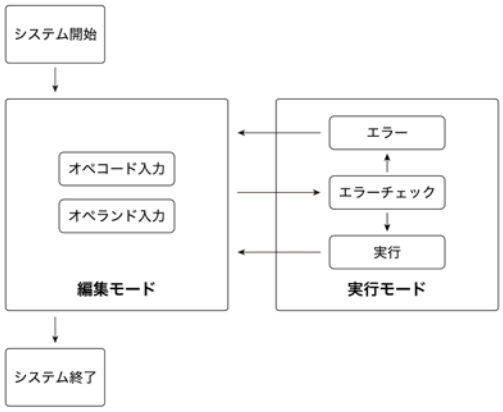

図 4　システムの状態遷移図

4. 実装

4.1 システム構成

図 5 に示すように，本システムは PC，カメラ，プロジェクタ，テーブル，マーカの描かれたタイル群といった要素によって構成される．カメラとプロジェクタは PC に接続され，それぞれ三脚によってテーブルの周囲に固定されている．カメラはテーブル上のマーカを撮影し，プロジェクタは図 6 のようにテーブル上に映像を投影する．カメラが取得したマーカの写った画像は PC で処理され，マーカ同士の距離を計測するのに用いられる．マーカ同士の距離に応じて PC はプログラムを編集し，プログラムが実行されると，実行結果がプロジェクタによってテーブル上に表示される．

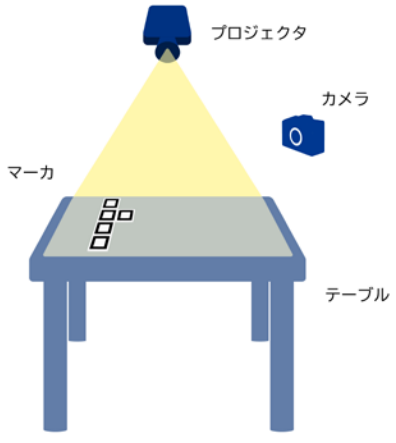

図 5　インスタレーション構成図

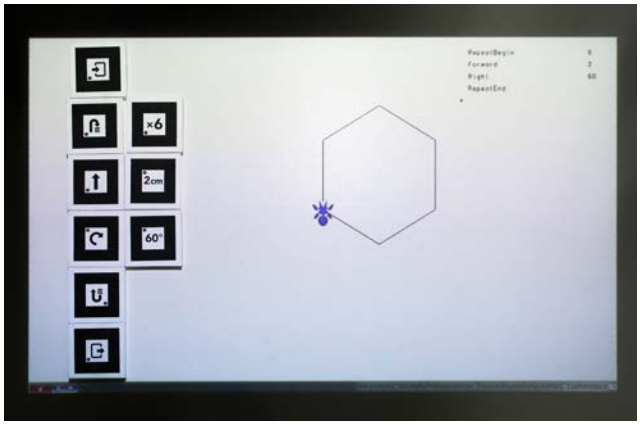

図 6　テーブルに投影した映像とタイル形インタフェース

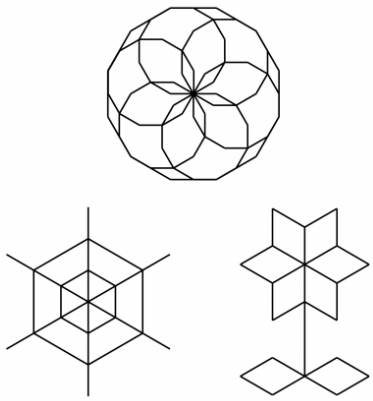

図 7　キャラクタによって描かれる図形の例

4.2 プロジェクションによる映像の提示

本システムでは，タイルを配置するテーブルの上部にプロジェクタを設置し，そこから映像の投影を行うことで，タイルによるプログラミング操作と映像による実行結果の確認を同時に可能にした．テーブルへのプロジェクションの様子を図 6 に示す．今回はキャラクタが図形を描く実行結果と，ユーザプログラムの状態を表す文字列を投影した．

マーカの位置情報取得にカメラによる画像認識を用いる関係上，マーカに人や物の影ができるとマーカが意図したとおりに認識されないことがある．テーブル上のマーカを置く領域に白色の映像を投影することによって，人や物の影をなくしマーカを鮮明に撮影することで，影による画像認識への影響を低減させている．

また本稿の段階では未実装であるが，プロジェクションを用いることにより，タイルの周囲に情報を提示することができるという利点もある．ユーザがプログラミングを行なっている最中に，操作上のヒントや，エラーの原因となっている箇所を示すことなどが可能となる．

4.3 画像認識によるユーザプログラムの入力

本システムは，タイルに描かれたマーカをカメラで撮影・画像処理し，マーカ間の距離を計測することでタイル同士がつながっているかどうかを判断している．その計測には ARToolKit を利用した．

タイルの上面にはマーカが描かれており，カメラでリアルタイムに撮影したマーカ画像をコンピュータで処理することで，それぞれのタイルの位置を認識することができる．マーカ同士の距離が閾値より短ければ，タイル同士が繋がったとみなし，コンピュータ側のユーザプログラムで命令の追加を行う．また長ければ，タイル同士が離れたとみなし，命令の削除を行う．

4.4 本システムを利用したプログラムの実行例

本システムを利用したプログラムの実行例を図 7 に示す．ユーザのプログラムが実行されると，テーブル上にプロジェクションされたキャラクタが命令にしたがって動き，こうした図形が描かれる．

5. 評価実験と考察

試作しているシステムの現状までの機能をテストし，研究的な方向性を評価するための予備実験を行った．マーカによるユーザインタフェースの操作性と，視覚的なフィードバックによるプログラミング学習効果の検証を着眼点とした．今回の実験では，学習者に提示するプログラミングタスクの形式が本システムと同様である Logo（ACSLogo For Mac OS X[6]）を比較対象とした．

実験は 4 名の学生に協力してもらい，2 人ずつのグループに分けて行った．一方のグループには Logo を，もう一方のグループには本システムを体験してもらい，共通のプログラミング課題に取り組んでもらった．

実験では両グループに同じ内容のレジュメが配布され，被験者はその内容に従って課題の達成を目指す．レジュメには，それぞれのグループの体験するシステムの基本的な操作方法が同じ書式で説明してある．こちらからは実験の大枠だけを説明し，操作方法の説明についてはレジュメによってできるだけ両グループとも同じ説明となるようにした．加えて，プログラムによって描く図柄を両グループ共通の課題として 4 問出題し，50 分間でプログラムを成功させた時間と数を測った．また，実験後にはアンケートに回答してもらい，課題とシステムについてのフィードバックを得た．

はじめに，アンケートによって被験者のプログラミング経験について把握したところ，4 名全員が大学の授業等でプログラミングを学んだことがあると回答した．しかし，プログラミングが好きと回答したのは 2 名で，得意と回答した者はいなかった．つまり，逐次処理や繰り返しなどの，プログラミングに関する基礎知識は持っているが，プログラミングに熟達したものはいないとして理解できる．これを踏まえて考察を行う．

【ユーザインタフェースについて】

被験者が課題の各問に要した時間の測定結果を表 1 に示す．結果として，4 問目まで時間内に達成できたのは 1 名

だけであった．全体としては Logo より本システムのほうがやや時間がかかっている．本システムでの操作手順の習得に時間がかかった分, Logo よりも時間が必要であったことが，実験の観察からも見てとられた．また，タイルを探すことや並べることにも時間がかかっていると考えられる．すなわち，被験者の属性として，キーボードによる文字入力に慣れていたので Logo のほうが，早くプログラミングできたのではないかと推測される．

実験後に行なったアンケート結果を表 2 に示す．アンケートでは, Logo グループの方がプログラミング操作をより簡単に理解できたと回答した．また，プログラミング操作自体に関しては，Spider グループの一名があまり簡単に行えなかったと回答している．

今回のレジュメのように，文章による操作手順の説明と，物理的形状をもつタンジブルインタフェースの相性が良くないとも考えることができる．タンジブルインタフェースの手順を説明する際には，実際に目の前で実演するか，動画によるチュートリアルを見せることでより簡単な理解につながると考えられる．

マーカに描かれているピクトグラムの意味はよく理解できたと回答されているが，間違えて配置する様子も観察された．ピクトグラムをおぼえることが新たなプログラミング上の負担とならないよう，文字も載せるなどの工夫を施す必要がある．

【実行結果のグラフィカルな表現について】

実験後のアンケートの結果，思い通りにプログラミングができたか，思い通りにいかなかった場合にうまく修正できたかについては，個人差のある結果となった．しかし，キャラクタを動かして図形を描くというタスクは，正しいかどうかがわかりやすく興味が持てたと全員が回答している．キャラクタを前進や回転などの命令によって動かすという実行結果の具体性が，インタフェースであるマーカのピクトグラムとの相性が良いとも考えられる．

6. おわりに

本研究の目的は，小学生程度の児童がプログラミングの基礎概念を学ぶにあたって、敷居を下げるという試みであった．キーボード入力や厳格な命令の記述を排除し，プログラムに手で直接触れることのできるタンジブルユーザインタフェースと，目で見てわかりやすく興味を持続させるグラフィカルな結果のフィードバックを採用した試作システムの実験により，これらの負荷を取り除くことができる可能性が認められた．一方で，プログラミングになれた人には，ピクトグラムよりも文字による命令の記載、マーカよりもコードの打ち込みのほうが効率がよい．すなわち，習熟レベルによる使いやすさの違いについては別途検討を行う必要がある。

プロジェクションによる操作ガイドや学習のヒントを表示するなど，システムにはそれらの課題を解決することができそうな機能上の余白がまだまだ残っている．今回の予備的な実験を踏まえて改善を進め，プログラミング経験のない小学生以下の子どもたちにも実際に体験してもらいたい．マーカの種類であったり，表現しうる情報量であったり，物理的に多くのマーカを整理してアクセスしやすくする方法も必要となる．このような，さまざまな観点からシステムの改善と実験とを繰り返しながら，総合的によりよいプログラミング学習ツールを研究していく．

表 1　被験者が各タスクに要した時間

グループ	被験者	問1	問2	問3	問4
Logo	A	3分	4分	9分	time up
	B	3分	3分	3分	time up
Spider	C	6分	11分	12分	time up
	D	6分	4分	17分	8分

表 2 アンケートの回答結果

		Logoグループ		Spiderグループ	
質問番号	質問内容	A	B	C	D
1	プログラミングの操作方法は簡単に理解できましたか	5	5	4	4
2	プログラミング操作は簡単でしたか	3	5	2	4
3	一つ一つの命令の意味はわかりやすかったですか	5	5	5	4
4	思い通りにプログラミングできましたか	3	2	2	4
5	プログラムが思い通りにいかなかった場合にうまく修正できましたか	4	2	2	4
6	キャラクタを動かして図形を描くというコンテンツに興味をもてましたか	5	4	5	5

謝辞　本プロジェクトに協力されている立命館大学映像学部の大島研究室各位に感謝します．本研究は，JSPS 科研費 16K00288 の助成を受けたものです．

参考文献

[1] 文部科学省．小学校学習指導要領．文部科学省告示第六十三号．平成 29 年 3 月 31 日．
[2] Papert, S.. On making a theorem for a child. *ACM '72 Proceedings of the ACM annual conference*, 1972, vol. 1, pp. 345 – 349.
[3] シーモア・パパート『マインドストーム―子供、コンピューター、そして強力なアイデア』未来社, 1995
[4] Maloney, J., Resnick, M., Rusk, N. and Eastmound, E.. The scratch programming language and environment. *ACM Transactions on Computing Education*, 2010, vol. 10, no. 4, Article No. 16. doi>10.1145/1868358.1868363
[5] Patterson-McNeill, H. and Binkerd, C. L.. Resources for using lego Mindstorms. *Journal of Computing Sciences in Colleges*, vol. 16, no. 3, March 2001, pp. 48 – 55.
[6] “ACSLogo For Mac OS X”. http://www.alancsmith.co.uk/logo/, (参照 2017-12-20).