プログラミング教育への興味喚起を目的とした迷路型STEM教育ツールの提案と実装

坂元 律矛 * 大塚 孝信 †

概要. プログラミング人材育成のため, 世界的にSTEM教育の重要性が認識されている. それに伴い, 様々なSTEM教材が提案・販売されている. しかし, STEM教育そのものには決定的なメソッドや教材が未だ存在しない. そのため, 各国で研究者や企業をはじめとして試行錯誤が繰り返されている. 本研究では, プログラミング教育そのものを行うものではなく, プログラミングに慣れ親しみ, 興味を喚起することを目的とし, 国際的にも年代的にも親しみやすい迷路を題材とした. 迷路を遊び・作る過程で浮上する様々な問題の原因を突き止め, 部分ごとにデバッグを重ね, 修正し, 完成を目指すことで, 論理的思考を養う迷路型STEM教育ツールを提案し, 実装した. また, 計15回のユーザ向け体験会を開催することで, 実際にユーザへの興味喚起が可能であることを示した.

1 はじめに

近年, 急速に進む社会の情報化に伴い, 子供達に対するプログラミング教育制度の整備が進められている. 例えば英国では2014年ごろより”Computing”と呼ばれるプログラミング教育カリキュラムが初等教育から導入され, 他にもハンガリーやロシアでも初等教育段階からプログラミング教育を必修とする施策が実施されている. また, エストニアやアメリカ, 香港, 韓国, 及びシンガポールなど様々な国でプログラミング教育に関して先進的な試行錯誤が行われている [1]. 日本国においても2020年度より初等教育におけるプログラミング教育の導入が決定され, 小学校・中学校・高等学校を通じて, プログラミング教育の推進が予定されている [2].

プログラミング教育の中でも特に重要とされているのが, 科学技術の基礎となる理数系科目の教育や既存の情報技術への理解や知識であり, これらを効率的に習得させる方法の一つとしてSTEM（Science, Technology, Engineering, and Mathematics）教育が注目されている [3]. STEM教育とは科学, 技術, 工学, 数学の分野における教育や学習を指す. 一般的に, STEM教育は学校の授業から課外活動までの正規・非正規を問わず, 就学前から博士後期までのすべての学年の生徒を対象としている [4]. STEM教材には, レゴマインドストーム [5] やlittle Bits[6]を始めとした電子工作とプログラミングの双方を利用して車やロボットなどを作る教材や, Scratch[7] やViscuit[8] のようなブラウザ上で編集可能なビジュアルプログラミングツールなど様々なものがある.

また, 我が国においてもプログラミング教育の必修化が決まったが, 様々な課題が浮上している. 日本の小学校プログラミング教育の狙いは, ”プログラミング的思考を育むこと”, ”プログラムの働きやその良さ, 情報社会がコンピュータ等の情報技術によって支えられていることに気づくとともに, コンピュータ等を上手に活用して身近な問題を解決し, より良い社会を築こうとする態度を育むこと”, 及び”各教科等での学びをより確実なものとする”の３つである [2]. 前提として, 生徒がプログラミングに取り組み, コンピュータを活用することの楽しさや面白さ, 物事を成し遂げたという達成感を味わうことが重要とされている. よって, プログラミング教育はプログラミング言語の習得が目的なのではなく, あくまで楽しさや面白さや達成感を味わうことで, プログラミングやコンピュータを上手に活用したいといった意欲を喚起することが前提となっている点が重要である. また, 狙いの一つとなっている”プログラミング的思考”とは, 自分の意図する結果の実現のために, どのような動きの組み合わせが必要であり, どのように改善していけば意図した結果に近づくのかといったことを論理的に思考する力だとされている [2].

しかし, こうしたプログラミング教育の狙いや指導要領が策定されているにも関わらず, 現状では, 具体的なカリキュラムや教材は決定しておらず, 各学校に応じたカリキュラムの展開や授業形式の工夫を各々に委ねている状況である. 本研究では, 特に日本の初等教育に狙いを定め, 諸所の課題を解決し, 幅広いニーズに対応するSTEM教材の開発を行う.

本論文の構成を示す. まず2章にて関連する既存研究について述べる, 次に3章にて提案するシステムの要件定義について述べ, 4章で実装手法について述べる. その後, 5章にてユーザ向けの体験会の実施結果について述べ, 6章においてまとめと今後

* 名古屋工業大学 情報工学科
† 名古屋工業大学大学院 情報工学専攻

の課題を示す.

2 関連研究

2.1 STEM 教材に関する事前調査

本章では, STEM 向けに製造されている製品および既存研究について網羅的に調査した結果について述べる. STEM 教材の代表的な例として, ビジュアルプログラミング言語, 電子工作ツール, 自動車等のロボット教材がある. 次節より, 各々の手法について述べる.

2.2 ビジュアルプログラミング言語

ビジュアルプログラミングの代表例として Scratch や Viscuit, プログラミン [9] がある. ビジュアルプログラミング言語とは, 従来の言語のようにテキストベースでプログラミングを行うものではなく, 視覚的な要素を用いることで, 積み木を組み上げるかのように直感的に理解し, プログラミング可能な言語である.

従来のプログラミング言語よりも覚えることが少なく, 可読性が高く, 工夫次第で算数や理科の学習にも用いることができる. また, 簡単にゲームを作成でき, 自分の作ったプログラムを手軽に友人に遊んでもらうことが可能であるため, 他者とのコミュニケーションのきっかけにもなり得る.

2.3 電子工作ツール

センサやスイッチ, モータや LED など様々な機能を持った電子モジュールを用いて工作する STEM 教材の代表例として, MESH[10] や little Bits が挙げられる. また, センサ類を 1 つのボードにまとめたマイコンボードの代表例として micro:bit[11] がある.

MESH は各ブロックとタブレット端末が無線により接続され, タブレット端末上でブロック型のビジュアルプログラミング言語を用いてプログラムすることで, 図画工作や理科の実験授業, 課題解決を取り扱う総合的な学習の時間で使用できる. 手軽に無線ネットワークを用いた電子工作を楽しむことができるのが特徴である.

さらに, little Bits ははんだ付けや配線作業, プログラミング知識を必要とせず, 電子回路が学べるマグネット式電子工作ブロックである. マグネット式のモジュールをつなぎ合わせるだけで電子回路を学ぶことができ, コンピュータを用いることなく STEM 教育を行うことができるのが特徴である. また, 用途に応じて, モジュールを付け替えることが可能であり, 簡単に試行錯誤が行える.

micro:bit とは英国の BBC[1]が主体となって作った教育向けマイコンボードであり, 英国では 11 歳から 12 歳の生徒全員に無償配布されている. LED やスイッチ, 加速度センサや磁力センサなど様々なセンサを搭載しており, ビジュアルプログラミング言語を用いて簡単にプログラムを書き込むことができる. 加速度や地磁気, 温度, 及び照度など, 普段は意識することのない身の回りの環境情報データを可視化して取り扱うことで, 自分の生活がより便利になるように身近な課題を解決する体験や, 既存の遊びに新たな要素を加え, 更に面白い遊びにする体験が行える教材である. micro:bit はコーディングに不慣れである先生や生徒であっても, ハードウェアとソフトウェアの違いを意識せずに利用ができる設計になっている [12].

[1] British Broadcasting Corporation の略. イギリスの公共放送局.

2.4 ロボット教材

ロボット・自動車教材の代表例として mBot[13] やレゴマインドストームがある. mBot は, 40 種類程度のパーツを組み立てて作った自動車型のロボットを, ビジュアルプログラミング言語や Arduino[14] プログラムを用いて操作することができる教材である. 超音波センサや赤外線センサを使い, センサ情報を用いた走行制御もプログラムできる. プログラムとテストを繰り返し, それぞれの意図した動きを実現するために試行錯誤することで, 論理的思考力を養うことが可能だとされている. 理科や数学と融合させたカリキュラムも存在している.

レゴマインドストームも mBot と同じく, ロボットを組み立て, プログラミングを施し, テストを繰り返すことで意図するロボットを作り上げる教材である. 組み立てモデルが自動車の他にもロボットアームやカラーソータなど, 幅広い種類が存在し, プログラミング言語もブロック型言語や C 言語, Java, Python など多くの言語に対応している点が特徴である. 上記の教材は丁寧な設計図が用意されているとはいえ, 組み立てが複雑で, 教材も高価である点がデメリットとして挙げられる.

また, プログラミングとテストの過程に重点を置き, 組み立てを必要としない教材として, 紙とペンを利用して論理的思考力を学ぶことのできる Ozobot[15] やホワイトボートの上を動き音楽再生や描画などができる Root[16] などが存在する.

3 迷路を題材とした STEM 教育ツールの提案と実装

3.1 システム設計

前章で述べた STEM 教育プロダクトには共通点が存在する. それは, どの教材にも **”意図した動作の実現のために, どんなパーツの組み合わせが必要なのか, またどのように組み合わせを改善していけば意図した動きに近づくのかを論理的に考えていく**

フェーズ”が含まれる点である．本研究では，この論理的思考力を養うフェーズを，迷路を題材にすることで，迷路で遊ぶ過程と，迷路を作る過程において論理的思考力を養うことが可能な STEM 教材を提案する．加えて，迷路型教材として実装することで，STEM に対する興味を喚起する教材を目的として開発した．

3.2 要件定義

日本の初等教育におけるプログラミング教育導入の狙いや問題点，ほかの STEM 教材調査の結果から，より幅広いニーズに対応する STEM 教材を開発するには，以下の要件定義を満たす必要がある．

1. 論理的思考力を養うことができる
2. 利用者の負担とならず，楽しみながら学ぶことができる
3. 事前説明を必要とせず，直感的に学ぶことができる
4. ペアやグループでの制作が可能
5. 幅広い年齢に対応が可能
6. 各環境に対応するために，総授業時間を任意で指定できる
7. 作品の共有や他人のアイデアの取り入れが容易
8. PC やタブレット端末が無くても動作する
9. 手軽にプロトタイピングを繰り返すことが可能

4 基本構成と実装

迷路とは，複雑に入り組んだ道を抜け，ゴールにたどり着くことを目指すパズルである．古くから知育としても利用され，ゴールにたどり着くまでの各分岐点において，どの道を選択するのか先を見越して考えることで論理的思考力を養うことができるとされる．しかし，迷路は機械的に探索を行うことで必ずゴールできてしまい，その方法は論理的思考力を養うとは言えない．そこで，スイッチを押すと開閉するドアや回転する床などの可動ギミックを追加し，論理的にギミックを動作させていかなければゴールにたどり着けないよう制限をかけることで，論理的思考力を養える STEM 教育ツールとして実装する．加えて，迷路を遊ぶ際だけでなく，作る際にも論理的思考力を養えるような仕組みを設計する．迷路とは，ビー玉迷路のような形式を指す．本研究では実際に木材やアクリルを利用し，実装している．また，センサの通過判定のため，鉄球を採用した．

システム構成は，大きく分けて基本ブロック，ギミックブロック，マイコンボード，ハーネス，及び筐体からなる．ブロックを筐体に設置することで簡単に迷路を作成可能なシステムを構築する．この際，何度も作り直すことができるよう，接着剤やはんだによる配線は行わず，容易に抜き差しが可能なハーネスとコネクタを用いる．

迷路をグリッド状に分割して考えた場合，要素として”直線”，”直角カーブ”，”十字路”，”T 字路”，及び”行き止まり”が必要となる．これら 5 種類の要素がブロックとして具現化することで任意の迷路が作成できる．これらの条件を満たすために，図 1 に示すように ”床パーツ”，”低い壁”，”高い壁”の 3 種類のパーツを用いて 5 種類の基本ブロックを実装可能なシステムを考案した．ブロックの種類を少なくすることで，小学生や未就学児でも理解可能なシンプルな構造とした．

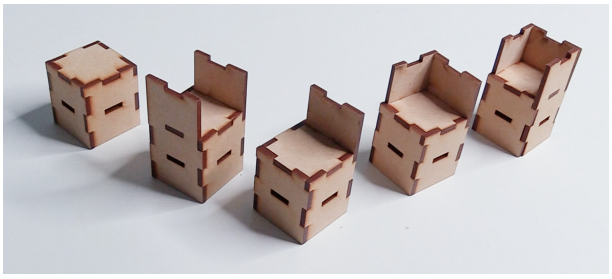

図 1. 基本ブロックの図. 左から十字路，直線，T 字路，直角カーブ，及び行き止まり

これに加えて，同じ規格で 1 グリッド四方のドアギミックブロックと 2 グリッド四方・3 グリッド四方の回転床ギミックを実装した．これらは，別に用意したスイッチブロックと連動する．ドアギミックと回転床ギミックは内部にサーボモータを利用しており，サーボモータのハーネスをマイコンボードに適切に配線することで通電し，動作可能状態になる．スイッチブロックにもハーネスが付属しており，これをマイコンボードに配線することで，スイッチとギミックの紐付けの前処理が完了する．この後．専用のエディタを介してマイコンボードにプログラムを書き込むことでスイッチとギミックを紐付けることができる．このハーネスシステムの採用により，任意の場所にギミックを配置することが可能となるほか，柔軟なギミック作成が可能となった．ギミックの作成例として，例えば，2 個の赤いスイッチと 1 個の赤いドアギミックを用意することで，開けるスイッチと閉めるスイッチを配置したドアを作成できる．スイッチとギミックは，色ごとにペアとして考える．他にも，プログラムを工夫することで，2 個の赤いスイッチを両方押さなければドアが開かないといったギミックも作成できる．

ドアギミックは 1 グリッド四方のブロック内にサーボモータを配置し，2 グリッドの長さのドアバーを左右に移動させることによりドアの開け閉めを表現している．開閉の状態を図 2 に示した．よって，ドアは一方の通路に対して開ける動作を実行すると，反

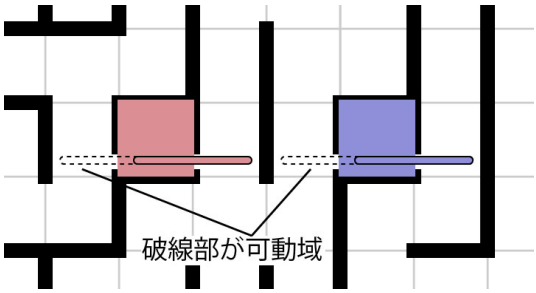

図 2. ドアギミックの状態図

対側の通路に対しては閉まる動作が実行され, 排他的なギミックとなる. これを用いることでより幅広いパターンの迷路を作成が可能となり, 逆に迷路作成時の制約にもなり得る.

回転床ギミックは, スイッチを押すことによって 0 度, 90 度, 180 度の状態に遷移するギミックである. 回転床上の壁の配置も組み替えることができる.

スイッチブロックは, 鉄球の通過を判定するために, 金属パターンの入った専用の基板を作成した. これにより, 鉄球が基板上を通過すると通電し, センシングできる. この基板は指などで触れても通電することはなく, 金属の物体が通過したときのみ反応する. また, スイッチが動作可能状態であるのか, またすでに押されていて動作しない状態なのかを判別するために LED によるデバッガ機能を搭載している.

本システムで実装した図 3 に示したマイコンボードは Arduino シールドとして実装しており, ドアギミックや回転床ギミック, スイッチブロックのハーネスを接続するコネクタを搭載している. これらはマイコンボード上にグルーピングされており, ギミックごとに整理してハーネスを接続することができる.

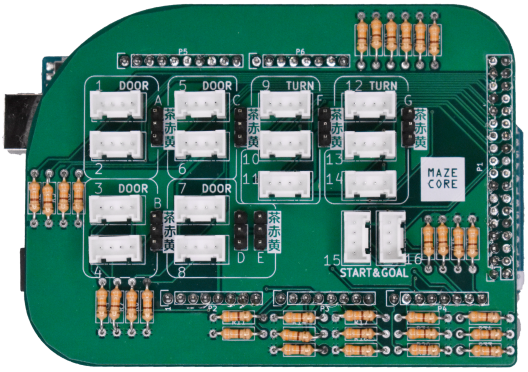

図 3. 作成したマイコンボード

プログラムエディタは図 4 に示したように, Google Blockly[17] を利用したビジュアルプログラミング言語を採用している. エディタ上でブロックをつなげて配置していくと同時に Arduino 用の言語に変換され, マイコンボードに書き込むことが可能となる. このエディタは, if 文や for 文を用いたセンシングや LED の操作, ギミックの詳細なハードウェア的操作は表面上露出しないようにユーザには見せない設計としている. 本エディタは Scratch などのように最小単位の全ての命令ブロックを実装しているわけではなく, 迷路に必要な処理をひとまとめにしたブロックとして実装している. これは, この迷路教材は既存の教材と比べ, 製作過程において, コーディング作業自体よりも迷路を作る作業を重要視しているためである. 用意したブロックは”1 番のスイッチが押されたときの処理ブロック’”という if 文とセンシングの結果を融合させたブロックや”A の回転床を右に回す”, ”B のドアを開ける”などのギミックを動作させるブロックなどがある. このブロック以上の処理をしたい場合には, Arduino の IDE を用いて直接マイコンボードに書き込むという選択肢があり, 初等教育以上のカリキュラムに利用可能である.

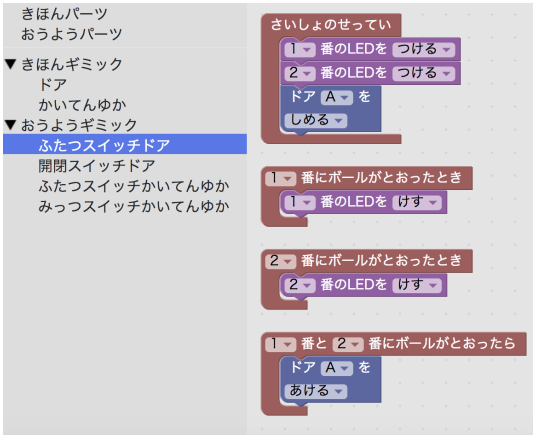

図 4. 作成したビジュアルプログラミングエディタ

この迷路教材はハードウェア教材であり, littleBits 等のように, ハードウェアによるアプローチのみで迷路を作成可能であり, PC やタブレット端末のない環境下においても教材を使用することを想定し, デフォルトコードを用意した. デフォルトコードにはいくつかのギミックのコーディングが施されており, このコードをあらかじめマイコンボードに書き込んでおくことで, PC やタブレット端末のない環境下においても迷路教材を使用できることとした. マイコンボードにはデフォルトコードに対応した印字とグルーピングが施されており, どの位置のコネクタにハーネスを接続するとどのギミックにひも付けされるかが容易に理解できるようになっている. デフォルトコードには, 開閉ドア（スイッチブロック 2 つ, ドアギミック 1 つ）が 4 セットと, 回転床（スイッチブロック 3 つ, 回転床ギミック 1 つ）が 2 セット分のコーディングが施されている. この方法を利用すれば, エディタを用いたプログラミングの過程を省略することができる.

筐体はグリッド四方で設計しており, 図 5 に示したような 6 グリッド四方や, 9 グリッド四方のものを基本として用意した. 筐体の床面にはハーネスを通す穴を多数設置し, ブロックを設置してハーネスを下に通し, マイコンボードに接続できる設計とした.

4.1 遊び方

スタートに鉄球を置くとゲームがスタートする. 筐体を傾けて鉄球を転がすことで迷路を解いていき, 回転床やドアギミックをうまく利用しながらゴールを目指す. ゴールに鉄球が到達するとゴールエフェクトとしてすべてのスイッチの LED が点滅し, ギミックが初期位置にリセットされる.

4.2 作り方

まず, 基本ブロックやギミックブロック, 筐体などを組み立てる. 続いて, ブロックを設置していく. スタートからゴールまで指でなぞり, ギミックを想像で作動させながら, きちんとゴールできるかデバッグをする. マイコンとブロックを接続する. デフォルトコードを使った際は, 適切な場所にハーネスを繋ぐことで, ハードウェア的アプローチのみで迷路を実装できる. デフォルトコードを用いず, 自らプログラミングを行う場合は, エディタを用いてソフトウェア的アプローチで迷路の実装が可能である. ゴールできるか実際に鉄球を転がしてテストを行い, 適宜ブロックを設置し直し, デバッグを繰り返すことで完成させる.

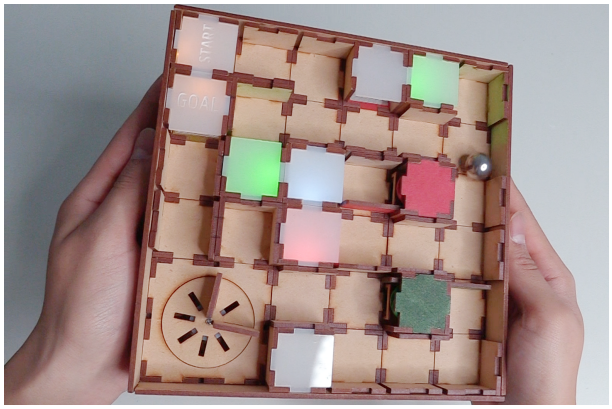

図 5. 完成した迷路教材 (6 グリッド四方)

5 ユーザ向け体験会の実施結果

CoderDojo[2]名古屋をはじめとして, 未就学児から小学生高学年までを対象に 2017 年 11 月より 2018 年 5 月までの間に計 15 回のユーザ向け体験会を実施した. CoderDojo 名古屋では, 主に Scratch や Viscuit などのビジュアルプログラミング言語を用いたプログラミングと, RaspberryPi[18] や micro:bit を用いた電子工作によるプログラミングを行っている. しかし, これらのコースを体験するには, PC やタブレット端末を持参する必要があり, また, まったくの初心者にとっては, Scratch や電子工作はハードルが高く, 親しみづらい題材であった. 最初は, 迷路教材の製作過程で必ずビジュアルプログラミング言語を用いたプログラミングの工程を含めるカリキュラムを組んでいたが, 途中からはデフォルトコードを用いて, 迷路により論理的思考を学ぶことで, PC やタブレット端末を利用しないカリキュラムを採用した. これにより, 誰でも気軽に迷路教材を利用してプログラミングに興味を持つことが可能となった. また, 迷路教材をきっかけに, Scratch や電子工作などの STEM 教材への興味喚起を果たすことも確認した. このように, 各環境のニーズに合わせてカリキュラムを柔軟に設定可能である.

グリッド状にブロックを配置していくシステムを採用しているため, 筐体の大きさを自由に変更できる. ワークショップを行って所要時間を計算した結果, 6 グリッド四方の迷路で, 前述した PC やタブレット端末を使わないカリキュラムであれば, 組み立てから完成までの全工程に平均 1 時間程度となり, 9 グリッド四方の迷路であれば, 2 時間から 3 時間程と, ワークショップや授業の総時間に合わせて変更できる.

ゴール不可を認めるかどうかというルール設定で難易度を変更できる. この迷路システムは, ギミックを作動させると迷路の形が逐次変更されるため, ギミックやブロックの配置によって不可逆性が発生することがある. ゴール可能ルートが 1 つあったとしても, その他のルートで完全に鉄球が身動きを取れず, ゴール不可になるパターンが存在してしまう可能性がある. このゴール不可をなくすためには, 迷路のギミック配置の際に綿密に全てのルートがゴール可能であるか考え, 調整を重ねてデバッグしていく必要がある. 迷路が大きくなればなるほど, 迷路自体の分岐数やギミックによる状態変化が増え, 難易度が増す.

この迷路教材は, 設計図やハウツーを見ながら迷路を作るフェーズを意図的に用意していない. 迷路という身近な題材を取り入れることで, 何を目指せばいいかわからないという状況を作らず, ユーザはひとまずスタートとゴールを配置し, その間の道を好きなように配置するという動作を気軽に実行できる. 加えて, この迷路教材は単純な機能のパーツやブロックしか用意していないため, それぞれに対して詳しい説明をせずとも, ひとまず触ってギミックの動作を各自で確認することができる. このようにユーザが仕様を容易に理解可能であるため, 完成し

[2] CoderDojo とは, 2011 年にアイルランドではじまった, ボランティアによる, 若い人のための, 地域に密着したプログラミング・クラブである.

た迷路が不調を起こしても, それがハードウェア的な不調であるのか, ソフトウェア的な不調であるのか, ハードウェアとソフトウェアの紐づけが間違っているのかなどを推測し, メンテナンスを行えることを体験会で確認した.

さらに, この迷路教材を何人でプレイできるか検証した. 6 グリッド四方の迷路であれば 2 または 3 人, 9 グリッド四方の迷路であれば最大 4 人までがグループになって教材を利用できることを確認した. これは, グループになった際に, なにもやることがないユーザが現れない人数である. ビジュアルプログラミング言語でコーディングするフェーズがある STEM 教材でペアでプログラミングをしようとすると, 端末は 1 人しか操作できないため, 端末の取り合いが起こることがしばしば見られた. 迷路教材は筐体にブロックを設置する際に, 担当領域を分担し, 自発的に協調していく様子が多数見られた. 迷路教材にペアで取り組む様子を図 6 に示した.

5 歳のユーザに対して迷路教材による体験会を行ったところ, 保護者や大人とともにペアで行うのであれば, 十分に迷路を完成させることができることを確認した. 親子で迷路を作ることでコミュニケーションのきっかけにもなる. 小学生低学年以上であれば, 同世代同士で協力しながら迷路を作ることができる. すでに迷路を作ったことのあるユーザが, 初めて迷路を作るユーザに対してルールや作り方を教え, 不調になった際にはメンテナンスの仕方も教え合っている様子が見られた.

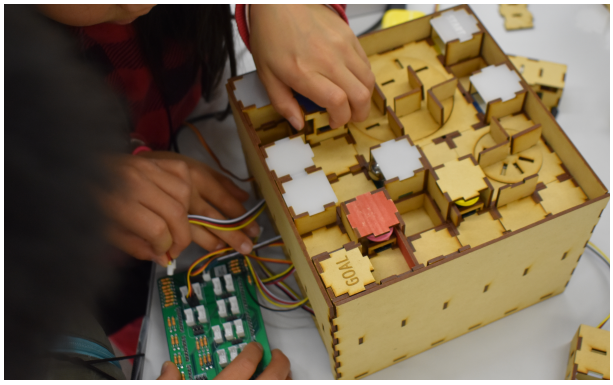

図 6. ペアで迷路教材に取り組むユーザ

6 まとめ

本研究では, プログラミング教育そのものを行うものではなく, プログラミングに慣れ親しみ, 興味を喚起することを目的とし, 国際的にも年代的にも親しみやすい迷路型 STEM 教育システムを提案し, 実装した. プログラミング教育に必要な要素をシステムに組み込み, 加えて論理的思考力を養うフェーズを迷路で遊ぶフェーズ・迷路を作るフェーズに適用することで, 年齢性別にかかわらず誰でも楽しく STEM 教育を行える教材になった. 体験会を行う中で, 年齢に応じて迷路の作成手順が違い, 完成した迷路にも差異がみられることに気づいた. 今後は, 年齢に応じた論理的思考力についての差異や, 論理的思考力そのものについての研究を行うことで, ユーザに対する興味を喚起しつつ, 論理的思考力を養えるシステムを発展させていきたい.

謝辞

本研究は独立行政法人情報処理推進機構による 2017 年度未踏 IT 人材発掘・育成事業の一部である.

参考文献

[1] 諸外国におけるプログラミング教育に関する調査研究（文部科学省平成 26 年度・情報教育指導力向上支援事業）報告書. 文科省. 2015.

[2] 小学校プログラミング教育の手引（第一版）. 文科省. 2018.

[3] 内海志典. イギリスにおける STEM 教育に関する研究―成立とその目的―. 科学教育研究 vol.41, No1, pp.13-22, 2017.

[4] H.B.Gonzalez, J.J.Kuenzi. Science, Technology, Engineering, and Mathematics (STEM) Education: A Primer. *CRS Report for Congress Prepared for Members and Committees of Congress.* 2012.

[5] レゴマインドストーム. https://www.lego.com/ja-jp/products/themes/mindstorms/mindstorms-ev3-31313.

[6] little Bits. https://www.littlebits-jp.com.

[7] Scratch. https://scratch.mit.edu.

[8] Viscuit. https://www.viscuit.com.

[9] プログラミン. http://www.mext.go.jp/programin.

[10] MESH. http://meshprj.com/jp.

[11] micro:bit. http://microbit.org.

[12] Thomas Ball, Jonathan Protzenko, Judith Bishop, Michal Moskal, Jonathan de Halleux, Michael Braun, Steve Hodges, Clare Riley. Microsoft Touch Develop and the BBC micro:bit. *IEEE/ACM 38th IEEE International Conference on Software Engineering Companion, pp637-640.* 2016.

[13] mbot. https://store.makeblock.com.

[14] Arduino. https://www.arduino.cc.

[15] Ozobot. https://www.ozobot.jp.

[16] root. http://www.codewithroot.com.

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

[18] Raspberry Pi. https://www.raspberrypi.org.