AR 技術を用いた物理/ビジュアルプログラミングのハイブリット型開発環境の提案

服部 圭介† 井上 馨太† 志賀野 桐生† 平井 辰典†

2020 年度から小学校でのプログラミング教育が必修化されることに伴い，誰もがプログラミングを学ぶことができる教材が求められている．そこで，実体のある物体を手に取って簡単にコーディングができる物理プログラミングと，比較的柔軟性が高い実行結果が得られるビジュアルプログラミングの二つを AR 技術により統合した新たな開発環境を提案する．これにより，現実世界の制限がある物理プログラミングと，PC 内でしか操作ができないビジュアルプログラミングの欠点を補う．評価実験により提案システムの有効性を評価する．

1. はじめに

2020 年度から小学校でのプログラミング教育が必修化されることに伴い[9]，効果的なプログラミング教育を実現させるための教材に対する注目が高まっている．初学者である児童がプログラミングを学習する上で一般的なテキストベースのプログラミングはハードルが高いため，視覚的にコーディングを行うことができるビジュアルプログラミングが注目されている．代表的なビジュアルプログラミング言語として MIT メディアラボが開発した Scratch [7] が挙げられる．Scratch は，命令が書かれたブロックを繋げていくことで手軽にプログラミングを行うことができるツールであり，初学者へのプログラミング入門教材として多くのスクールや教室で導入されている．一方で，Scratch を始めとした多くのプログラミング言語/開発環境ではコンピュータを使用しなければプログラミングができないということが課題であると我々は考えている．この場合，PC やタブレットの画面の中の世界（プログラムの実行画面）において命令を与えているという認識を持たなければプログラムと実行結果との間のギャップを埋めることが難しい．これは，児童がそれまでにどれだけ情報機器に触れてきたかによっても理解度に差が生じることが考えられる．画面の中のキャラクタをコントローラで動かすようなゲームで遊んできた児童もいれば，そのような機器に生まれて一度も触れてこなかった児童もいることを想定しなければならない．カメラ等のセンサを用いたプログラミングを行うことで，現実世界の映像を踏まえたプログラミングを行うことも可能ではあるが，ツールの前提として画面の中の実行結果を理解することは求められる．

このようなコンピュータを操作することでプログラミングを行うツールとは対照的に，実際に手に取ることができる物体を使ってプログラミングを行うプログラミングキットのことを物理プログラミングと呼ぶことにする．物理プログラミングでは，命令に対応した物体を現実世界で手に取って組み立てることでロボットを命令通りに動かすことができる．実世界での命令が実世界におけるロボットの動きに直接対応しているため，子供でも直感的にプログラミングできるようなツールとなっている．しかし，物理プログラミングは現実の物体を扱ってプログラミングを行うため，命令可能なことが物理的に実行可能なことに制限されてしまう．例えばキャラクタが空を飛ぶような動作は，コンピュータの実行画面の中ではプログラミング可能であるが，物理プロミングで実現することは困難である．

そこで，我々は現実世界の物体を利用することによる直感的なプログラミングと，プログラミングの醍醐味とも言える自由な実行結果との両方を合わせ持った物理/ビジュアルプログラミングのハイブリット型開発環境を提案する．具体的には，物理プログラミングのように現実世界の物体を使ってプログラミングを行い，その結果を AR 技術によって現実世界に重畳させる形で提示することによって現実世界におけるプログラミングをベースに，拡張現実空間における CG アニメーションという形で自由なプログラミングを実現する．これにより，コンピュータの画面の中に制限されていたビジュアルプログラミングと，物理法則やロボットの駆動域に制限されていた物理プログラミングの両者の制限を払拭したようなプログラミングツールの実現を目指す．

2. 関連研究

プログラミング言語の種類は年々増加しているが，プログラミングの学習に重きをおいた言語やツール，研究事例についてもこれまでに多数提案されてきた．特に，初学者に向けたプログラミング入門言語として，ビジュアルプログラミング言語に関する研究は長きに渡り行われてきた．ビジュアルプログラミング言語の歴史は古く，1960 年代に Ellis らによって”Grail” [10]という言語が発表されている．Grail は，画面上に図形を構成していくことでプログラミングができるシステムであったが，当時のコンピュータの性能ではグラフィック処理を行うことが難しく，教育用途のビジュアルプログラミング言語の普及にはまだ時間を要する状況であった．現在では PC が一般に普及し，その性能も向上してきたため，ビジュアルプログラミングによるプログラミングの学習は比較的容易に行うことができる環境になってきたといえる．

多くのビジュアルプログラミング言語は命令が記されたブロックを組み合わせることによってプログラミングを行うものであり，その代表的な言語として Scratch が挙げら

† 駒澤大学

れる．Scratch は，子供でも簡単にプログラミングを学習することができるようにデザインされている．さらに低年齢の子供向けに開発された ScratchJr や，Scratch を参考に文部科学省が開発したプログラミンなど，プログラミング教育を支援するための様々なビジュアルプログラミング言語が存在している．ビジュアルプログラミング言語は，必ずしも教育用途だけで利用されるものではなく，UEI による MoonBlock [2]はプロが行うような実用的なプログラミングをより簡単にするという目的で開発されている．ビジュアルプログラミングを実用的なプログラミングへと繋げるために，Google は Blockly [5]という言語を開発している．これは，ビジュアルプログラミングによってプログラミングした結果を，JavaScript や Python などのテキストベースのプログラミング言語のソースコードに変換した結果を表示させるものである．これにより，ビジュアルプログラミング言語からテキストベースのプログラミング言語への移行を円滑に行うことを支援している．

ブロックを用いないビジュアルプログラミング言語として，Viscuit [6]が挙げられる．Viscuit は，メガネと呼ばれる書き換え用の機能にユーザが書き換え前後の絵を描いていくことによってプログラミングを行うものである．元の絵から後の絵へとどのように変更するかをプログラミングしていくため，単純でありながらも高い表現力を誇っている．一方で現在普及しているテキストベースのプログラミング言語との間の隔たりが大きいため，テキストベースのプログラミング言語を習得するためには再度学習をする必要が生じる．

初学者がプログラミングを学ぶ上で，その言語を用いたプログラミングが直感的であるかどうかは非常に重要な要素である．プログラムと実行結果との間の因果関係が直感的でなければ，お手本となるプログラムの理解や，既存のプログラムの編集も困難である．我々は，プログラミングにおいて直感性を低下させる要因の一つに，コンピュータの利用があるのではないかと考えている．これまでに紹介した多くのプログラミング言語は，画面の中で構築した命令が，画面の中の実行結果として得られるものである．たとえば，「前に進む」というシンプルな命令によってキャラクタを動かすことを考える．実行結果では，キャラクタが「画面内」で前に進むが，この画面の中の世界を理解できることが前提となっている．全国の小学生が全員プログラミングを学ぶということを考えたとき，コンピュータに触ったことがない児童は当然多くいることが予想できる．そのような場合，コンピュータの操作に加え，画面の中の世界を認識するということが最初の課題として挙げられるのではないかと考えている．

コンピュータそのものの操作や理解を必要とせずに，実際に手に取ることができる物体を使ってプログラミングを行うことができるツールを物理プログラミングと定義する．物理プログラミングの例として，Microsoft による Project Trino [1]では，目が見えない子供たちがプログラミングを学ぶためのツールを開発している．このツールは，音に関する命令を実行するパーツを組み合わせていくことで，音を出力していくような物理プログラミングツールである．

Fisher-Price による Code-A-Pillar [4]は，イモ虫型のロボットに命令を表すパーツを加えていくことでロボットの動作をプログラミングする玩具である．このツールを用いることで，現実世界で組み合わせた命令が現実世界のロボットに対して反映されるため，直感的にプログラミングが出来る．しかし，現実の物体に対するプログラミングであるため，物理的な制約を超えた自由なプログラミングは実現できない．

八城らが提案した物質プログラミング[11]は，Scratch のブロックを実体のあるブロックとして制作し，それらを組み合わせることで PC 画面内の車，もしくは車型ロボットの動きをプログラミングできる開発環境である．この開発環境についても，車型ロボットの動きは物理的な制限を超えてプログラミングすることができない．

これらの先行研究，既存ツールを踏まえ，我々は以下の二点を合わせ持ったプログラミング言語の実現を目指す．

・ビジュアルプログラミングを始めとしたコンピュータを用いる従来のプログラミング言語における実行結果の柔軟性
・物理プログラミングにおけるプログラムと実行結果との間の直感性

上記の二つの要素を同時に実現するために，現実世界に直結していながらも，物理的な制約を受けずに現実世界を拡張可能な AR 技術を用いることが有効であると考えられる．Fusté らは拡張現実空間内を歩くキャラクタの動きをマーカによって操作する Paper Cubes [3]を提案した．このツールでは，キャラクタは現実の床の上を歩くため，直感的に命令を与えることが可能である．しかし，このツールは AR アニメーションのコントロールという域に留まっているものであり，プログラミングを行うためのツールではない．そこで我々は，AR 技術を用いることによって，物理/ビジュアルプログラミングのハイブリット型開発環境を実現する．

3. システムの全体像

本稿で提案する物理/ビジュアルプログラミングのハイブリット型開発環境は，机の上で物理的にプログラミングした結果が，スマートフォンの画面を通じて現実空間に重畳される形で実行される．本システムにおけるプログラミングから実行までの流れを図 1 に示す．

プログラミングには「マーカブロック」という手に取って組み合わせることができる木製のブロックを利用する．マーカブロックは，ビジュアルプログラミング言語 Scratch におけるブロックの概念に AR マーカの機能を付加したものであり，一つひとつに命令が記述されている．ユーザがマーカブロックを取捨選択して並び替える作業が本システ

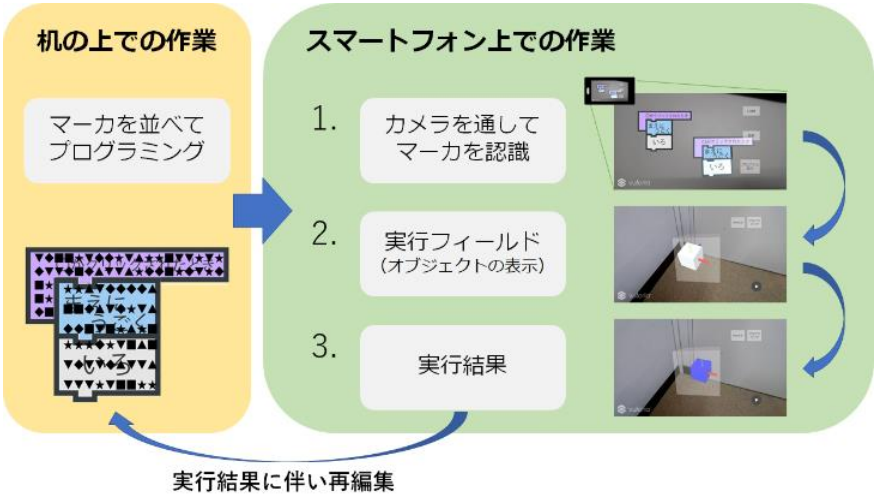

図 1　システムの全体像

ムにおけるプログラミングの作業にあたる．Scratch ではブロック同士の組み合わせの可否が決まっているため，実行不可能な組み合わせの命令を並べることができないようになっている．そこで，Scratch に倣って本システムのマーカブロックにも凹凸をつけることで不適切な順番で命令を並べることができないようなデザインにした．マーカブロックは，スマートフォンのカメラを使って認識する AR マーカの役割も担っているため，表面には特徴点を有した模様が描かれている．この模様のパターンによってマーカの種類を判別する．マーカには「まえにうごく」などの動作に関する命令や，「もし〜〜にふれたら」，「〇回くりかえす」などの if 文や for 文に相当する制御に関する命令が記載されている．

マーカブロックを組み合わせることによって作成したプログラムは，現実空間に重畳する CG オブジェクトに対してプログラミングされる．この CG オブジェクトは，Scratch におけるスプライト（キャラクタ）にあたるオブジェクトであり，初期状態では 3D の白い立方体オブジェクトである．ユーザが机の上でマーカブロックを並べ終えてプログラムを完成させたら，スマートフォンのカメラを通してプログラムを認識する．正常に認識されたマーカブロックについては，実物のマーカブロック表面に描かれていた特徴点の模様が消えた状態で画面上に重畳される．すべてのマーカブロックが認識されている状態で「ロード」ボタンを押すことでプログラムが読み込まれる．この作業が一般的なプログラミング言語におけるコンパイルの作業にあたる．

一般的なプログラミング言語では実行画面やコンソール等を通じてプログラムの実行結果を確認するが，本システムでは現実世界すべてが実行画面に対応する．そこで，本システムにおいて，実行結果を重畳するための現実世界のことを「実行フィールド」と呼ぶことにする．実行結果を表示するにあたり，まず，実行フィールドにおける座標の中心となる基準点を設定する．この作業は，現実世界における床をスマートフォンでタップすることによって行う．次に，実行フィールド内のオブジェクト出現させたい場所にスマートフォンを移動させて，オブジェクト出現ボタンをタップすることでフィールド内の任意の位置にオブジェクトが出現する．このオブジェクトは拡張現実空間における CG オブジェクトであるため，ユーザが移動して別の角度から見ても，初めに出現させた位置に留まり続ける．

オブジェクトにはプログラミングのプロセスで作成したプログラムが割り当てられている．たとえばユーザが，「もしオブジェクトにふれたら，まえにうごく」というプログラムを作成してオブジェクトを出現させた場合，オブジェクトをタップして触れることによってオブジェクトが移動する．本システムのマーカブロックの認識，実行フィールドの座標取得については Vuforia を用い，ユーザによるプログラミングのバックエンド側のプログラムは C#を用いて開発した．

4.　本システムを用いたプログラミング

本章では，本システムを用いたプログラミングの方法について詳述する．

4.1 マーカブロックの種類

本システムにおけるプログラミングはマーカブロックを組み合わせることで行われる．マーカブロックは Scratch を参考にして制作したものであり，Scratch と同様に命令のジャンルによって色分けされている．現段階では，4 種類のジャンルで 12 種類の命令についてのマーカブロックを制作している．制作した 4 種類の命令ジャンルは，「イベント」，「制御」，「動き」，「見た目」に関するマーカブロックであり，これらはすべて Scratch にも存在する分類である．今後，マーカブロックの種類は Scratch と同等またはそれ以上に数を増やしていく予定で

図 2　本システムを用いてプログラミングをした際のプログラミング画面（左），実行フィールド画面（中央），実行結果（右）

ある．以下に，現時点で制作，実装済みのマーカブロックの詳細について述べる．

まず，「イベント」マーカブロックは，プログラムを実行させるきっかけを与えるための命令ジャンルである．Scratch では，「緑の旗がクリックされたとき」というような命令として実装されている．本システムでは以下の 2 種類の「イベント」マーカブロックを用意している．

・もしオブジェクトにふれたら
・もしオブジェクトにふれたらくりかえし

これらは，実行フィールド上に配置したプログラミング済みのオブジェクトに対して，命令を実行させるトリガーとなる命令である．「もしオブジェクトにふれたら」をプログラムの先頭に配置することで，実行フィールド上でオブジェクトに触れた際にプログラムが実行される．「もしオブジェクトにふれたらくりかえし」は，プログラムを繰り返し実行する形で実行するための命令である．

「動き」マーカブロックは，オブジェクトの動きに関する命令を表すものであり，「まえにうごく」，「カメラにむける」，「音を出す」の 3 種類のマーカブロックを実装している．「まえにうごく」は，オブジェクトが向いている方向に進むという命令であり，オブジェクトは，初期状態では最初に実行フィールド画面に切り替えた際の Z 軸方向を正面に向いている．「カメラにむける」はユーザが持っているスマートフォンのカメラの方向をオブジェクトが向くという命令である．「音を出す」は，あらかじめ決められた音をスマートフォンのスピーカから鳴らす命令である．現時点ではユーザによる音の変更はできないが，今後に実装ではそのようなカスタマイズも可能とする予定である．

「見た目」マーカブロックは，オブジェクトの見た目に関する命令を表すものであり，「いろを変える」，「きえる」の 2 種類がある．「いろをかえる」はオブジェクトをあらかじめ決められた色（現状の実装では青色）に変えるという命令である．「きえる」はオブジェクトを消去する命令である．

「制御」マーカブロックは，プログラミングにおいて重要な，様々な制御を行うための命令を表すものであり，if 文，for 文に対応するようなマーカブロックが用意されている．if 文に対応する制御マーカブロックとして，「もし地面にふれたら｛」，「もしカメラにふれたら｛」，「｝」の 3 種類のブロックを実装した．「もし地面にふれたら｛」は，オブジェクトが地面に触れた際に「｛｝」の中の命令が実行されるというもので，「もしカメラにふれたら｛」は同様にオブジェクトがカメラに触れた際に実行されるというものである．for 文に対応する制御マーカブロックとして，「○回くりかえす｛」と「｝」がある．これは，「｛｝」内の命令を 3 回繰り返すためのマーカブロックである．ここで，「｝」のマーカブロックに関しては，if 文及び for 文の開始ブロックとは独立しているため，異なる色の「｝」を追加した場合や，「｝」を追加し忘れた場合などには，誤ったプログラムとなってしまう．この場合，プログラムのロードをした際にエラーメッセージが表示されるようになっている．

音の種類や，変更する色，if 文の条件の種類，for 文の繰り返し回数などをユーザがカスタマイズするための機能は，現段階では未実装ではあるが，今後のアップデートで機能を追加していく予定である．

4. 2 プログラムと実行結果の例

現状のシステムは，まだプログラミング可能な命令の種類が限られているが，4.1 節で説明した 12 種類のマーカブロックを用いて基礎的な動作に関するプログラムを作成することができる．図 2 に，本システムのマーカブロックを用いて実際にプログラミングした際のマーカブロックと実行フィールド上での実行結果を示す．

4.3 AR を用いたことによる特徴的なプログラム

4.2 節に示したプログラムは，画面内の実行画面に対してプログラミングを行うようなビジュアルプログラミング言語でも実行可能なプログラムであるといえる．本節では，AR 技術を利用していることにより実現可能な，現実と CG オブジェクトとの関係性を利用した特徴的なプログラムについて紹介する．

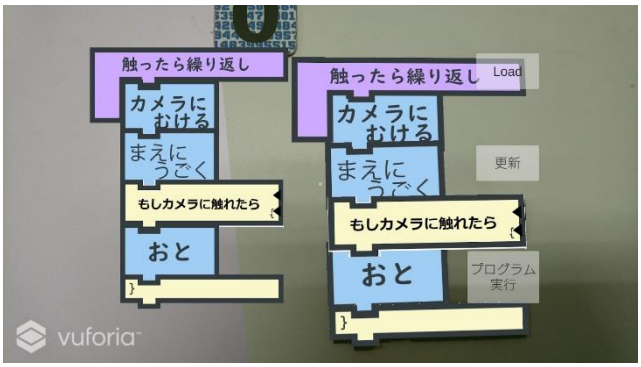

図３　ユーザとオブジェクトとの鬼ごっこを実現するプログラム

現状の 12 種類のマーカを用いて制作した特徴的なプログラムとして，CG オブジェクトとユーザとの鬼ごっこを実現するプログラムを図 3 に示す．

このプログラムでは，オブジェクトが常にカメラ（ユーザ）の方向を向いて前に進む．そのため，ユーザはオブジェクトから遠ざかる方向に動き続けなければならない．オブジェクトがカメラ（ユーザ）に触れたら，音が鳴ってオブジェクトは消失する．

4.4 想定される教材としての利用方法

本システムは，プログラミング教育を支援することを目的として開発しているため，プログラミング学習を支援する教材としてどのような利用方法を想定しているかを記述する．本システムは，Android 端末にアプリケーションの一つとしてインストール可能である．児童全員に Android OS のスマートフォンやタブレットを配布することができれば一人一人がプログラミング体験をすることができる．マーカブロックに関してもプログラミングをする児童の人数分必要であるが，これに関しては，紙のコピーであっても動作させることができるため，必要枚数をコピーするだけでよい．これは，全国すべての小学校で，クラスの人数分の PC を用意することに比べて低コストで導入ができるのではないかと考えられる．

また，仮に人数分の環境が用意できない場合でも，人数分のマーカブロックに対して，数人に一台の Android 端末で実行結果を確認させるという利用方法も考えられる．もしくは，数人のグループに端末とマーカブロックを 1 セットずつ配布し，グループでプログラミングをすることも可能である．本システムの特徴として，実体のあるマーカブロックを使うため，PC を使ったプログラミングと違い，他者と協力してのプログラミングが容易にできる点が挙げられる．そのため，ペアプログラミング等の複数人の協力によるプログラミング学習も比較的やりやすいものと考えている．

PC を利用した授業の問題点の一つとして，児童が PC を勝手に操作してしまうことを先生一人では完全に制御しきれない点が挙げられる．それに対して，マーカブロックを並べる作業は，先生一人でも児童の作業を全体的に観察できるため，児童の様子を見ながら授業を進めるなどといった点でも適していると考えられる．

今後，実際に児童を対象として，本システム体験するワークショップを開く予定であり，ワークショップを通して本システムのプログラミング教育における効果を検証していきたいと考えている

5. 評価実験

我々が開発したプログラミング環境を実際に体験してもらうことで，評価を行った．評価は，6 名のプログラミング初学者に実際にシステムを使用してもらった上でアンケートに回答してもらう形で行った．アンケートによる評価項目は，プログラミングの理解度，面白さ，操作性，AR の必要性，学習の継続性の 5 点である．回答結果及び実験中の被験者の様子を観察することで提案システムの評価を行う．

5.1 実験条件

被験者はプログラミング初学者である大学生 6 名と，プログラミング歴 1 年半の大学生 1 名の計 7 名である．被験者は全員，評価実験を行う以前に授業の一環で Scratch を使用した経験を有する．また，プログラミング歴 1 年半の大学生 1 名が習得している言語は C++, ruby, Java であり，具体的に習得した項目は条件文，反復文，変数，配列，画像処理であると回答している．評価実験は，一度に一名ずつ，一人 10 分程度で行った．最初の 5 分で操作方法を説明し，残りの 5 分間でシステムを使ったプログラミングをしてもらった．

5.2 評価実験におけるプログラミングの内容

評価実験では，我々が指示した数種類の課題を実現するプログラムを各被験者に作成してもらった．課題に取り組んでいる間，被験者には一切の説明やヒントを与えなかった．取り組んでもらった課題は以下のとおりである．

・画面上のオブジェクトをタップしたら前に進むプログラム
・画面上のオブジェクトをタップしたらオブジェクトの色が変わるプログラム
・画面上のオブジェクトをタップしたら，ずっと前に進み続けるプログラム
・画面上のオブジェクトをタップしたら，ずっと自分の方向に進みカメラに触れたら音を出して消えるプログラム

課題は，順番に難しくしていったが，一人の被験者を除いて全員が全課題を実行可能なプログラムを作成することに成功した．残りの一人の被験者に関しても，最後の課題以外は完成させることができた．

実験終了後，アンケートを通じてプログラミングの理解

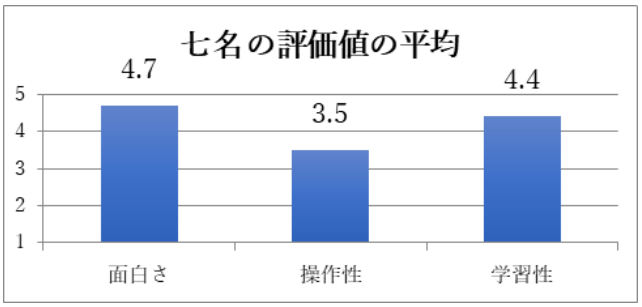

図 4　評価実験におけるアンケートの結果

度，面白さ，操作性の 3 項目を 5 段階で，本システムにおける AR の必要性と本システムを用いた学習の継続性に関する 2 項目については「はい」か「いいえ」のどちらかで評価してもらった．また，このツールの良い点，改善点に関しても自由記述欄を設けて記述してもらった．

5.3 実験結果

アンケート結果を図 4 に示す．評価値は，5 がその項目に関して最もポジティブな値で，評価値が低い場合はネガティブな評価ということになる．この結果から，面白さ，学習性の二点に関して高い評価が得られたことがわかる．AR の必要性に関する質問に対しては，全被験者が「はい」と回答し，本システムにおける AR の必要性が認められた．また，全被験者が今後もこのツールを使用して自主的にプログラミングを勉強したいと回答した．

評価実験の結果，本システムの操作性に関しての評価は低く，改善点に関する自由記述欄にも「マーカブロックを読み取り易くするように改善してほしい」という意見や，「ツールが重くなることがあるので軽くしてほしい」などの意見があった．ツールの良い点に関する自由記述欄には，「現実の世界に仮想の物体を現すことができる部分が面白いと感じた」との意見や，「マーカブロックを手に取ってプログラミングできる部分が良い点だ」との意見が得られた．実験中の被験者の様子の観察からは，マーカブロックを組み合わせずに距離を離して使用しようしている被験者がいたが，マーカ同士の距離が大きく離れていたわけではなかったため，プログラムの認識ができてしまっていた．このような点については，システムの実装によってユーザが正しくマーカブロックを扱えるようなガイドを出すなどの工夫の余地があると考えられる．

5.4 考察

アンケートの結果から，提案システムの面白さに関して高い評価が得られたが，自由記述欄への記述内容から推察できたこととして，プログラミングそのものの面白さではなく AR 技術の真新しさの部分が評価されていたような印象を受けている．操作性に関しては我々が説明をしなくても感覚的に操作ができるように誘導する必要があると感じた．また，「マーカブロックを手に取ってプログラミングできる部分が良い点だ」との回答から，物理プログラミングのように現実世界の物体を利用することによる直感的なプログラミングがある程度実現できたものと捉えられる．しかし，ビジュアルプログラミング言語のような自由なプログラミングが実現できたかを判断するには，まだマーカブロックの種類が少ないと考えられる．今後，マーカブロックの種類を増やすことで，ユーザが自由にプログラミングを行うことができるツールの実現を目指す．

6 まとめ

本稿では，プログラミング学習の支援を目的として，AR 技術を用いた物理/ビジュアルプログラミングのハイブリット型開発環境を提案した．本システムは，物理プログラミングのように現実世界の物体を使ってプログラミングを行い，その結果を AR 技術によって現実世界に重畳させることで直感的なプログラミングを可能としながら，拡張現実空間における CG アニメーションという形で物理法則に縛られない自由なプログラミングを実現するものである．評価実験を通じて，プログラミングの直感性についてはその有効性を示すことができた．

本研究の今後の課題として，操作性の向上やマーカブロックの種類を増やすことが挙げられる．UI デザインの再検討や，ユーザが直感的にプログラミングやプログラムの実行をできるようにするためのシステム設計の改善等を通して操作性の向上を図りたい．また，マーカブロックの種類を増やすことで，より自由なプログラミングを可能とするだけでなく，AR 技術を用いることに関連したユニークな命令やプログラミング体験のデザインを行っていきたい．

本稿では，AR とプログラミングとの関連性について，あまり深く議論できていなかったが，CG オブジェクトと現実の物体とのインタラクションをより多様に行えるようにすることで，AR 技術の必然性と，AR ならではの面白さを追究していきたい．将来的には，ユーザが作ったプログラムが現実の物体に干渉し，現実のオブジェクトも含めてプログラミングできるようなツールの実現を目指したい．それが実現できれば，生活において便利なものをユーザ自身が作れるようなツールとなる．拡張現実空間内の CG オブジェクトは，現実世界の物理的な制約を受けないものとしてプログラミング可能である．もしも CG オブジェクトが現実世界へと干渉可能となれば，本システムは，魔法をプログラミングすることができるツールとなる．このような可能性を示すことで，小学生だけでなく，多くの子供たちの心をつかむことができるプログラミングツールとしていきたい．

さらに，ユーザが作ったプログラムを拡張現実空間内で共有できるようにすることで，誰もがその結果を個々のデバイスから観測出来るようなシステムを作ることを目指している．それにより，現実世界という共通の実行画面を舞

台に様々な人が作った様々なプログラムが新たな世界を彩るような未来を夢見ている．

参考文献

1) Allison Linn: With Project Torino, Microsoft creates a physical programming language inclusive of visually impaired children, https://blogs.microsoft.com/ai/project-torino-microsoft-creates-physical-programming-language-inclusive-visually-impaired-children/ （2018 年 8 月 7 日アクセス）
2) 秋葉原リサーチセンター: MOONBlock, http://moonblock.jp/ （2018 年 8 月 7 日アクセス）
3) Anna Fusté, Judith Amores, David Ha, Jonas Jongejan, and Amit Pitaru: Paper Cubes: Evolving 3D characters in Augmented Reality using Recurrent Neural Networks, NIPS 2017 Art Gallery, pp.1-3, (2017)
4) Fisher-Price: Code-A-Pillar, https://www.fisher-price.com/ja_JP/product/98301 （2018 年 8 月 7 日アクセス）
5) Google: Google Blocky, https://developers.google.com/blockly/ （2018 年 8 月 7 日アクセス）
6) 原田康徳: 子供向けビジュアル言語 Viscuit とそのインタフェース，社団法人情報処理学会研究報告, Vol.2005-HI-116, No.7, pp.41-48, (2005).
7) Mitchel Resnick: Scratch: Programming for All, Vol.52, No.11, pp.60-67, (2009).
8) MIT Media Lab:Scratch.jr, https://www.scratchjr.org/
9) 文部科学省: 新学習指導要領 ～情報教育・ＩＣＴ活用関連部分のポイント～, http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/03/30/1375607_01.pdf （2018 年 8 月 7 日アクセス）
10) 清水亮: enchant.js と enchantMOON が目指すもの，映像情報メディア学会誌， 68, 2, pp. 131-135, (2014)
11) Ellis, T. O., J. F. Heafner, and W. L. Sibley, The Grail Project: An Experiment in Man-Machine Communications. Santa Monica, CA: RAND Corporation, (1969).
12) 八城朋仁，迎山和司: 物質プログラミング -物質によるプログラムの可視化と開発環境の制作-，情報処理学会インタラクション, Vol.C3, No.4, pp.647-650, (2014).