Vol.2015-IS-132 No.3
2015/6/13

道路標識をメタファーにしたプログラミング支援システムの提案

桑原礼子[†1]　伊藤永悟[†2]　藤本貴之[†3]

近年，日本の義務教育期間中でのプログラミング教育を推進する動きが見られる．しかしながら，情報教育一般とは異なり，一定の専門が求められるため，教員や開発環境などの教育環境が整っているとは言い難い．また，最低限の専門的知識が前提となるため，生徒達のプログラミングにおける基本的な理解が追いついていないという問題もある．そこで本研究では，小・中学生でも手軽に取り組め，且つプログラミングに関する基本的な理解を促進するプログラミング教育システムを提案し，実装する．

Proposal of programming creation application using road signs by smartphones

REIKO KUWABARA[†1]　EIGO ITO[†2]
TAKAYUKI FUJIMOTO[†3]

In recent years, in Japan, programming education during compulsory education has been promoted. However, it is difficult to say that the educational environment, such as teachers and development environment is in place because programming skills are required a certain amount of expertise unlike general education information. In addition, since the minimum of technical knowledge is assumed, there is a problem that students can't caught up with a basic understanding of programming. In this study, we proposed and implement a programming education system that tackle with ease even in elementary and junior high school students and facilitates a basic understanding of programming.

1. 研究の背景

近年日本ではプログラミング教育を推進している．2013年6月，日本政府が発表した成長戦略の素案の中でも，義務教育でのプログラミング教育について盛り込まれていた[1]．しかしながら，プログラミングを教える人材は不足することが懸念されている [2]．プログラミングスキルを習得しても，教職ではなく本職のプログラマーになる人もいれば，逆に情報系の学生でなく，文系の学生が教職課程として情報系の教職を取る可能性もあるという [3]．また，子供たちの能力や興味関心は多様であり，プログラム指導が可能な少数の教員で多くの生徒に対応していくのは困難な状況となっている [4]．よって，今後，義務教育の現場でプログラミング教育を促進してゆくためには，効率的な教材，特にノンプログラマーでも使用できる教材が必要になる．

寺元貴幸の調査によれば，基本的なアルゴリズム全般の理解が難しく感じるなど，プログラミングをかなり難しいと感じている学生も多い[5]．プログラミングを難しいと答えた学生は全体の約7割，アルゴリズム全般が苦手であると答えた学生は全体の３１％に及んでいた．

早い段階におけるプログラミング教育には，教育環境が整っていない，どのように指導して行くかなど，いくつか課題がある．

2. 既存のプログラミング教育/支援システム

簡単な操作によってプログラミング教育を支援するシステムは，はこれまでにもいくつか開発されている．

例えば，文部科学省が WEB 上で公開している「プログラミン」がある．「プログラミン」[6]では予め役割を持ったオブジェクトを組み合わせることで絵を動かすなどの様々な動作を行うことが出来る．本アプリケーションを通じて創ることの楽しさや方法論を学ぶことを目的とし，子供たちが自発的に触れ，楽しくプログラミングを知れるように設計されている．

MIT メディアロボが開発した「スクラッチ」[7]はブロックと呼ばれるフォームを積んでいくことでプログラムの作成を行うアプリケーションである．正しい構文の書き方が分からずともブロックの凹凸がはまるかどうかでプログラムが組めるかどうかを判断できる．

このようなビジュアルプログラミング教材では，コンピュータに関して特別な知識がなくとも視覚的に操作してプログラミングすることが可能である．この特徴から経験的なプログラミング学習を実現する．

†1 東洋大学大学院 理工学研究科
Toyo University
†2 東洋大学大学院 工学研究科
Toyo University
†3 東洋大学大学院 工学研究科
Toyo University

しかし，ビジュアルプログラミングはコマンドやソースコードを取り扱う場合と比べ楽しく直感的に操作できるが，プログラムの中身そのものはブラックボックスになってしまう.その結果,手順だけは簡単に覚えることができるが，分岐や変数などの論理構造は見えにくく理解しにくくなっており，本当の理解を妨げている面も否定できない．そこで，本研究では，そのような既存のビジュアルプログラミングによるプログラム教育支援システムに対し，論理部分を意識させることに重きに置いたシステム設計を行う．

3. スマートフォンを用いたプログラミング支援システム

本研究で提案するシステムは，義務教育でのプログラミング教育の補助を行いたいため，小・中学生を主たる対象として，簡易な操作によってプログラムを作成することができ，且つプログラミングの仕組みを理解することを目的とする．特に，本システムでは，より身近で手軽なツールを用いてプログラミングを体験することができるようにするため，スマートフォンを用いてプログラミングを行うことができるアプリケーションとして開発する．本アプリケーションは目的に合わせて選択肢を選んでいくことでプログラミングを行う．アイコンやフォームを組み合わせる触覚重視の分かりやすさではなく，簡単なボタン操作での論理部分の分かりやすさを追求している．

ビジュアルプログラミングは見た目で分かり易くすることを目的にアイコンやフォームがベースとなっており，それらに対する視覚的操作と処理が紐付けがされるようになっている．そのため実行される命令の順番が分かりにくい，入力から出力までの処理の過程が見えにくいといった問題がある．提案アプリケーションはアルゴリズムを分かり易くすることを目的とし，逐次処理を視覚的に表現・操作できるように設計する．選択肢を処理の順番に選んで命令をいれていくため過程が見えにくいといった問題はない．

旧来の文字列によるプログラミングでは，全体を見渡す時にとても見づらく，分かりづらい．そこで本アプリケーションでは，アイコンとして道路標識を用いて認識性を高める.アイコンは様々な情報を圧縮して有したものである.そのためビジュアルプログラミングとはコンセプトは異なる.道路標識という身近で,しかも,特別な記号ではなく,共通して意味を理解できる記号を用いることでその意味を覚える手間を省き，道路標識が元来持つ視認性の高さ，直観性の高さを取り入れることが可能である．道路交通標識は，画像・記号とそれが意味する内容に乖離が少なく，普遍的である．多くの場合は少しの変更で，日本以外でも応用が容易となる．

4. 提案アプリケーションの設計概要

試作アプリケーションは Mac OS 10.8.5 環境下の Xcode 6.1.1 で Objective-C を使い, iOS 用アプリケーションとして開発した．

4.1 アプリケーションの起動

本アプリケーションを起動すると，タイトルと「プログラムを作る」「プログラムを見る」「使い方」の選択肢が表示される.「プログラムを作る」を選択し，名前を付けてプログラムを作成すると，作成画面に移る．作成画面では，実行時表示される画面とプログラミングに必要な動作を行う「入力」,「挿入」,「管理」,「実行」の四つの選択肢が配置されている．実行し，保存されたプログラムはメニュー画面の「プログラムを見る」で確認することが出来る．

4.2 プログラム作成画面

作成画面では，実行時表示される画面と「入力」,「挿入」,「管理」,「実行」の四つの選択肢が配置されている．入力はプログラミングの命令を入れていく作業，挿入は作成画面上に文字や絵の挿入をする作業，管理は現在作業を行っているページや変数，画像の管理を行う作業，実行は作ったプログラムを実行するものである．

a) 入力: プログラミングの命令を入れる作業を行う．命令の種類はプログラムを作るのに最低限必要であると思われる命令「位置移動」「演算する」「回転する」「条件分岐」「表示する」「ページ移動」の 6 つである.「位置移動」は画像や文字の x 座標や y 座標，高さや広さを指定した値に変えることが出来る.「演算する」は変数の代入,加算代入,減算代入，乗算代入，除算代入，余剰代入の設定を行う.「回転する」は回転させる文字や画像と回転させる角度を設定することで画像や文字を回転させる.「条件分岐」は if 文を示す．変数などが特定の条件に当てはまった時に実行される．オブジェクトの場合は例えば押された時に実行されるように指定，変数の場合は論理演算（〜かつ〜），比較演算（〜より小さい）などの条件を設定し，変数がその条件に当てはまる数値に達している時実行されるように指定を行う.「表示する」は何を何処に表示するか設定する.「ページ移動」はどのページに移動するか設定する．

b) 挿入: 文章や絵を画面上に挿入する．初めに文字と絵のいずれかを選択すると，名前の設定と，画面上に表示されるものの設定を行うことができる．設定後作成ボタンを押すと，編集画面上に絵や文字が表示され，ドラッグして配置を行う．

c) 管理: 作業を行っているページや変数，画像の管理を行う．ページ，変数，画像の三つの選択肢があり，ページを選択すると現在編集を行っているページとそのページに入力されている命令の手順を表示し，変数を選択するとプログラミングで使う変数の登録や管理が出来る．画像を選択すると編集ページ上で配置されている画像の管理，使用したい画像の取り込みなどを行うことが出来る．

d) 実行: 編集したプログラムを試しに実行させるとどのような動作を起こすのかを確認する．デバッグでプログ

ラムの再編集，保存で作成したプログラムが保存される．

e)　プログラムを見る: 作成し保存されたプログラムの一覧が表示され，実行したいプログラムの横にある「起動する」ボタンを押して，起動する．

5. プログラムの作成例

画面上にある図形を押すとその図形が 90 度回転し，その隣に押された回数が表示されるプログラムを作成するとする．

- [Step 1] プログラムの作成: タイトル画面で「プログラムを作る」を選択し，四角の枠の中に名前を入力し編集ボタンを押すと，編集画面と 4 つのボタンが表れプログラムの作成を開始する．

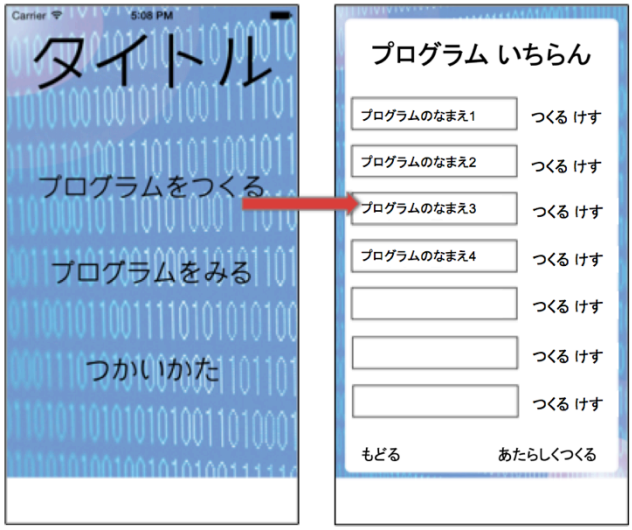

図 1　タイトル画面

Figure 1　Title screen

- [Step 2] 図形の配置: まず，画面上に命令を行う相手，図形を配置するため，挿入ボタンを押し画像を選択する．画像に名前を付け，表示する画像を選択して作成ボタンを押すと，編集画面上に設定した画像が表れる．ここでは「Pict1」という名前を付け，星の画像を表示させるものとする．

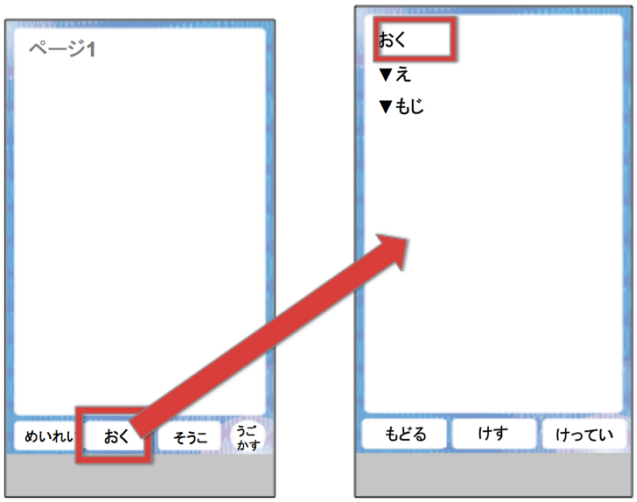

図 2　図形の挿入

Figure2　Inserting shape

図 3　画像配置の詳細設定

Figure3　Advanced image placement

- [Step 3] 文字の配置: 押した回数を表示させるには予め文字を配置しておく必要がある．ステップ２の画像の配置と同様に挿入ボタンを押し，文字を選択，名前と表示させる文字を設定，配置する．ここでは「letter 1 」という名前を付け，letter 1 と表示させるようにする．
- [Step 4] 変数の設定: 押した回数の記録をとるために管理ボタンを押し，変数を選択する．変数名欄に変数の名前を入力し，初期値を設定する．ここでは変数名を「押した回数」とし，初期値を 0 に設定する．

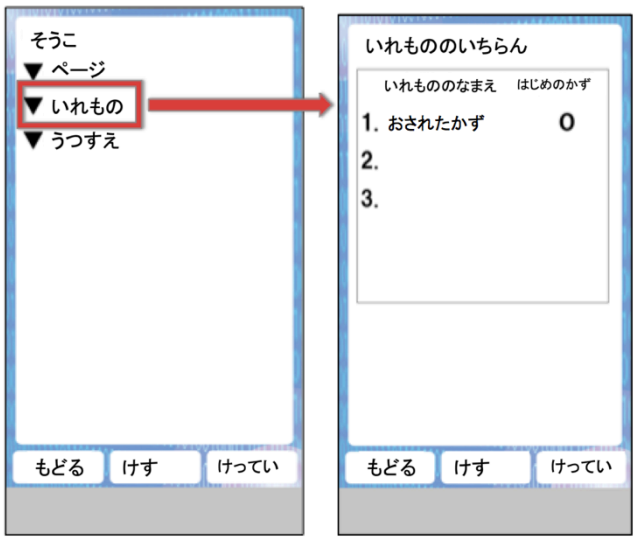

図 4　変数の設定
Figure4　Setting variable

- [Step 5] プログラミング: プログラム全体は前章で述べたように道路標識を用いて表示される．図５のような車のマークがある場所が現在編集を行っている部分を示しており，ここを押すことで命令文を入れていく．

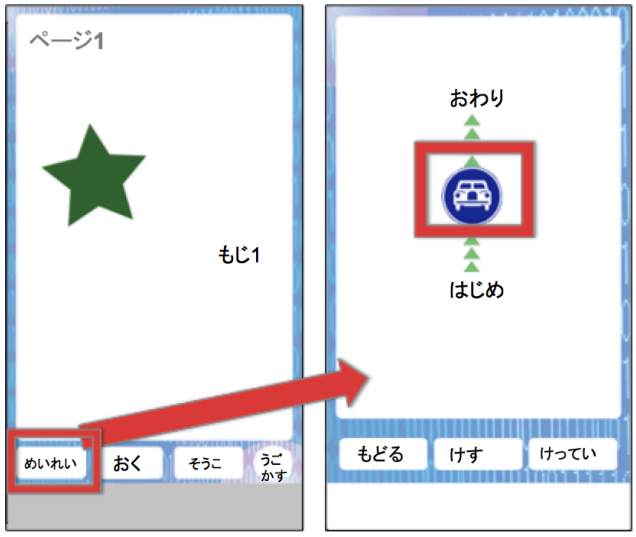

図 5　プログラミング開始画面
Figure5　Programing start screen

- [Step 6] 図形の表示: 車のマークを押すと６つの命令の選択肢が表れる．今回は図形を表示したいので，「表示」を選択する．すると何処に何を表示するかの設定画面と，プログラミング全体を見通すときに表示される道路標識の絵が表示される．設定を終えて作成ボタンを押すと，表示の命令が追加される．ここで表示する画像を四角に設定した場合，編集中はステップ２で行った画像配置時の設定を変えない限り星型で表示されるが，プログラムが実行される際には四角の図形が表示される．

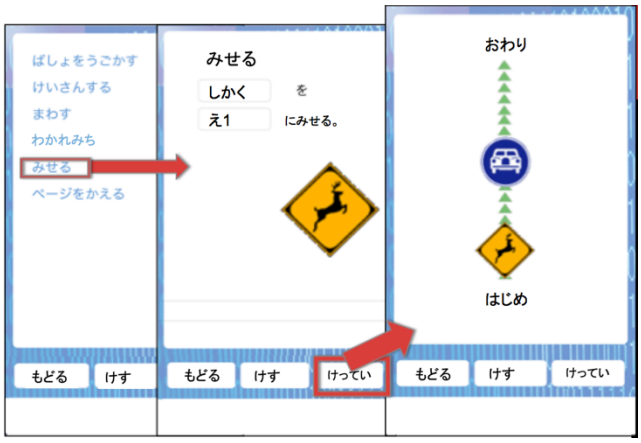

図 6　表示の命令
Figure6　Display of instruction

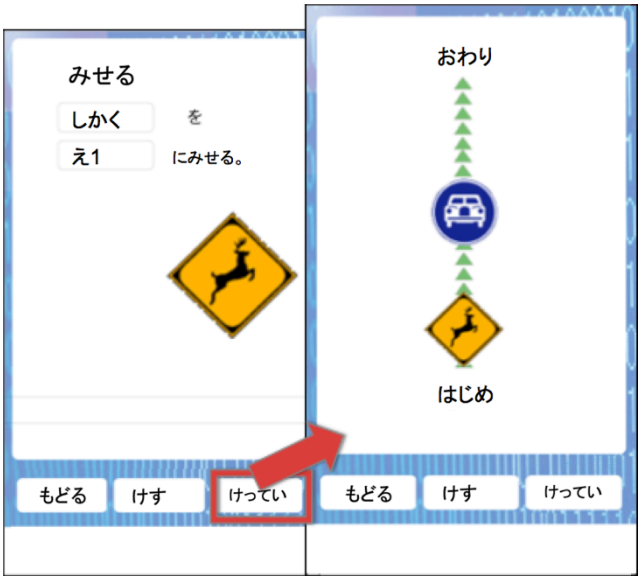

図 7　表示の命令 2
Figure7　Display of instruction 2

- [Step 7] 条件分岐の設定: 押された時に画像の回転や押した回数の演算が実行されるようにするため，条件分岐を選択する．実行される条件を設定すると，条件分岐内のプログラムの編集に入る．ここでは「絵１」が押された時を条件に設定する．条件に当てはまった場合の処理は左の道，当てはまらない場合の処理は右の道の車マークを押して命令を入れる．

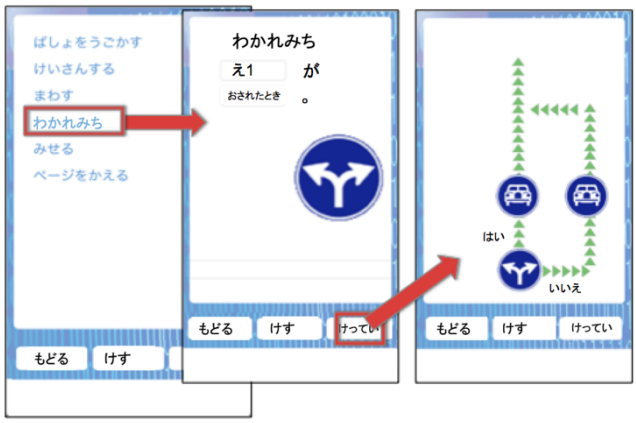

図 8　条件分岐の設定

Figure8　Setting of conditional branch

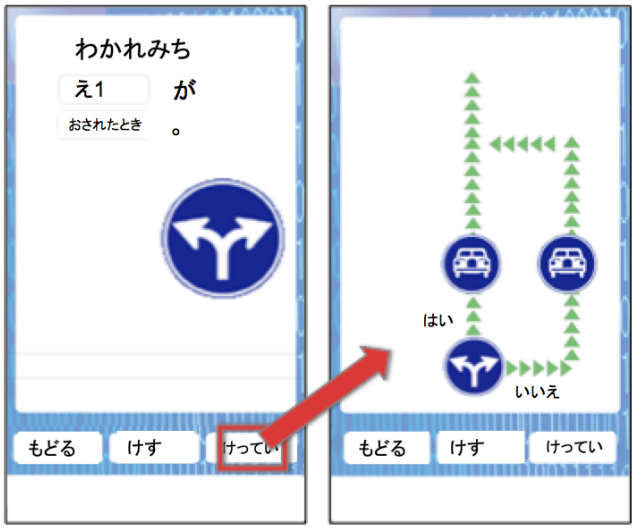

図 9　条件分岐の設定 2

Figure9　Setting of conditional branch2

- [Step 8] 条件分岐内の設定（回転する）：何が何度回転するかを設定する．ここでは「絵１」を９０度回転するよう設定する．

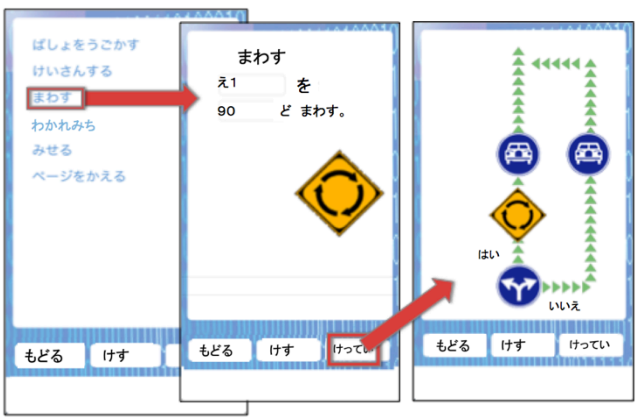

図 10　回転の命令

Figure10　Rotation of instruction

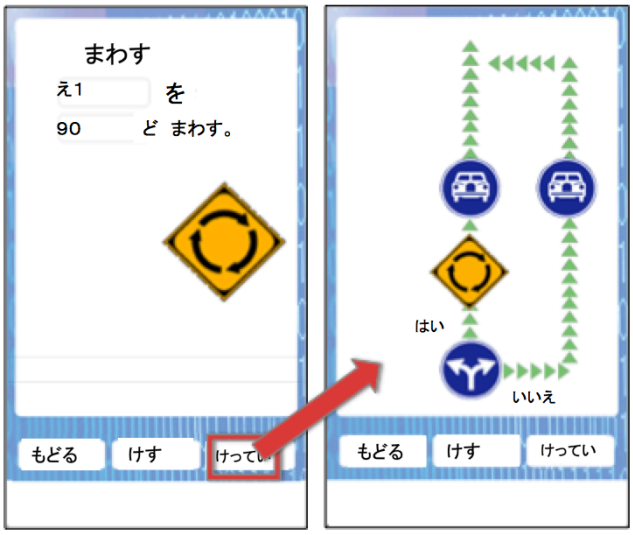

図 11　回転の命令 2

Figure11　Rotation of instruction2

- [Step 9] 条件分岐内の設定（演算する）：　押した回数を表示するために押した回数の演算を行う．ここでは変数名「押した回数」に 1 を加算代入する．変数は表示するように命令をしない限りは画面上には表示されない．これで条件分岐内の設定は終わったので，作成を押すと，プログラム全体に条件分岐の命令が入れこまれる．なお，処理手順が長引き画面外に出そうな場合はスクロールをして確認を行う．

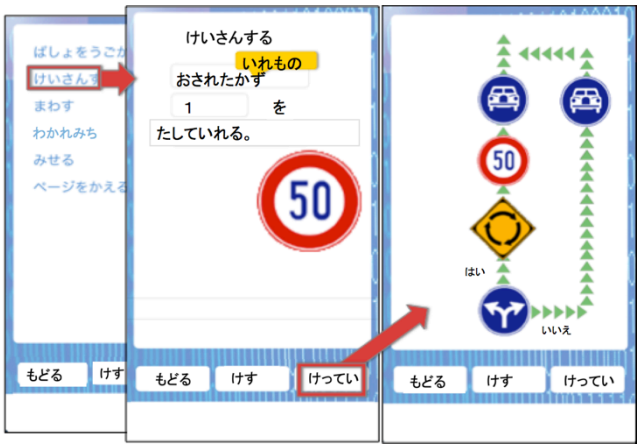

図 12　演算の命令

Figure12　Calculation of instruction

- [Step 10] 表示する: 最後に押した回数を表示させるために表示する命令文を入れる．表示を選択し，変数名「押した回数」を「文字１」に表示するよう設定すると，プログラムを実行し絵が押された際に押した回数が表示されるようになる．

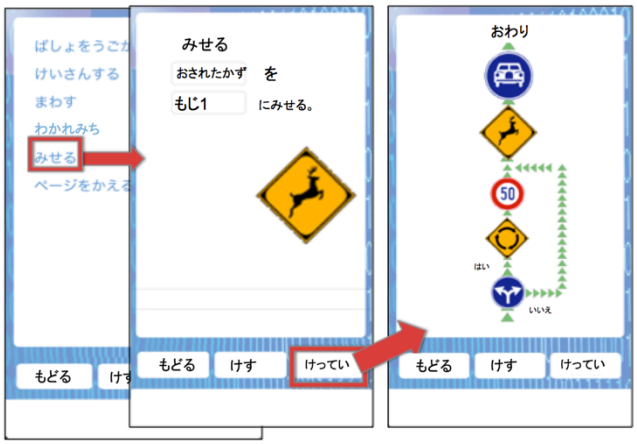

図 13 表示の設定
Figure13 Display of instruction 3

- [Step 11] 実行: 実行を押すと，作成されたプログラムが実行され，動作確認を行うことが出来る．保存ボタンを押すと，「プログラムを見る」から起動することが出来るようになる．

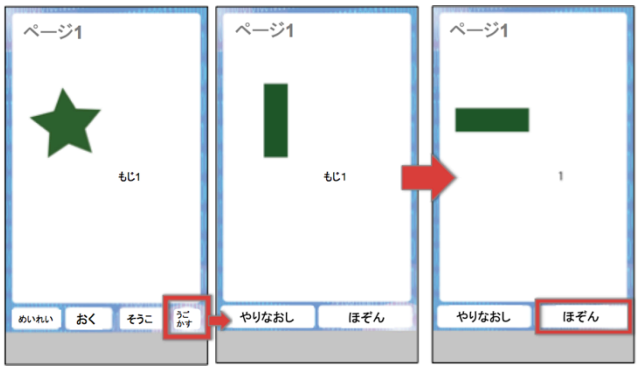

図 14 実行画面
Figure14 Execution screen

- [Step 12] プログラムを見る: タイトル画面に戻り，「プログラムを見る」を選択，作成したプログラムを選択すると，完成品がプログラム編集時より広い画面で起動される．

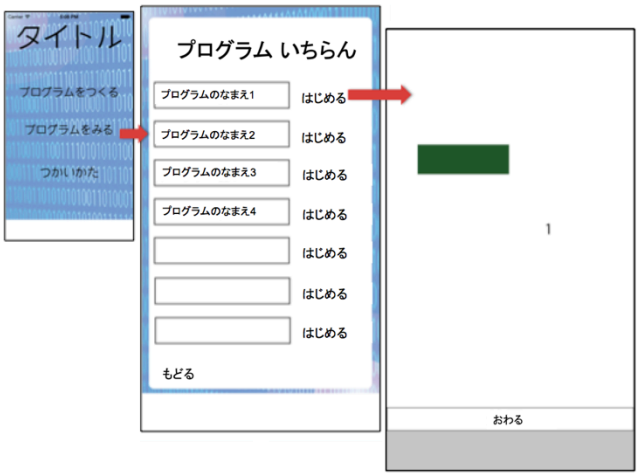

図 15 プログラムの閲覧
Figure15 Browsing program

6. 今後の課題

本論文で提案したアプリケーションは，日常生活で見かける道路標識を用いることで記号を新たに覚える必要を省き，プログラム全体を見やすくしている．しかしながら，何よりも重要なことは開発後の被験者実験・評価によって，本研究のコンセプトの有用性を検証することである．開発をすすめ，道路標識への認識の差から，年齢層別に被験者実験を行い，有用性を検討したい．

また，道路標識は国により異なるものを利用している．本論文では日本のものを使用したが，将来的には個々の国に対応した標識を利用可能にするべきだろう．

参考文献

1) 首相官邸, “成長戦略素案,” p.46 (2013).
http://office.microsoft.com/ja-jp/word-help/CH010097020.aspx
2) 宮脇亮, “エンタープライズ 0.2 - 進化を邪魔する社長たち -282 アベノミクスが失速している理由を見つけるプログラミング教育 0.2 (2014).
http://news.mynavi.jp/column/itshacho/282/
3) 清水亮, “義務教育におけるプログラミング教育の課題と可能性,” (2015).
http://wirelesswire.jp/2015/01/20173/
4) 神谷加代, “現代っ子の習いごとはプログラミングも当たり前――「TENTO」レポート (3/3),” (2013).
http://www.atmarkit.co.jp/ait/articles/1312/02/news026_3.html
5) 寺元貴幸, ”プログラミング教育を支援する問題解決環境に関する研究”, 宇都宮大学大学院工学研究科博士論文(2011).
6) 文部科学省, “プログラミン,”
http://www.mext.go.jp/programin/
7) MIT メディアラボ, “スクラッチ,” https://scratch.mit.edu/