Vol.2012-CE-113 No.11
2012/2/5

ブロックエディタ方式による プログラミング構造化教育支援システム

保井　元[†1]　松澤芳昭[†2]　酒井三四郎[†2]

近年，ブロック型言語による初等プログラミング教育の環境の提案と改善が行われている．その代表として Squeak や Scratch などがある．しかしながら，それらは構造化技法をサポートしておらず，C,Java などのテキスト記述型言語への移行も考慮されていないため，次のステップへの発展が難しいという問題があった．そこで我々は，OpenBlocks フレームワークを利用して，構造化技法をサポートし，Java 言語と相互変換できるブロック型言語を開発した．文科系の大学生 1 年生 107 名に本システムの使用実験を実施し効果の検証を行った．

A Block Editing System to Support Structured Programming in Introductory Programming Education

Hajime Yasui,[†1] Yoshiaki Matsuzawa[†2] and Sanshiro Sakai[†2]

In the past decade, improvements of the environment of an introductory programming education by block-based programming language has been made by Squeak, Scratch, etc. However, there are two problems for them. One is they do not support structured programming. Another is a lack of consideration for a migration and an improvement to text-based programming languages, such as C and Java. Hence, using the OpenBlocks framework proposed by MIT, we developed a system which has functions to support structured programming and a converter to change OpenBlock and Java in both ways. We conducted the use experiment of this system on 107 university students first graders liberal arts and succeeded.

†1 静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University
†2 静岡大学情報学部
Faculty of Informatics, Shizuoka University

1. はじめに

近年，Squeak[1] や Scratch[2] などブロック型言語が注目され始め，実際に教育現場にも取り入れられるようになってきた．ブロック型言語はテキスト記述型言語に見られる文法エラーがないので，初学者にとって使いやすいエディタになっている．しかしながら，それらのブロック型言語はプログラムの構造化技法をサポートしていないため，プログラムの構造を考える事を目的とした教育を行うことができない．また，テキスト記述型言語への移行も考慮していないため，将来的にテキスト記述型言語を必要とする受講者にとって，次のステップへの発展が難しいという問題がある．

本研究では，プログラミングの構造化教育支援をテーマに，MIT(Massachusetts Institute of Technology) で開発された OpenBlocks フレームワーク[3] を利用し，教育支援システムの開発を行った．そして，開発したシステムの仕様実験を実施し効果の検証を行った．

本稿では以下，2 章で関連研究について述べる．3 章でプログラムの構造化教育を支援する方法と，本システムに行った実装を述べる．5 章で本システムの使用実験を実施し，得られた結果を報告する．この結果を踏まえて 6 章でこの試みの考察を行う．

2. 先行研究

本研究のようなタイルスクリプティングとテキストスクリプティングを併用した学習，教育支援に関しての先行研究がある．石田ら[4] は PAD(Problem Analysis Diagram) と C 言語を併用した学習支援システムを提案している．このシステムは，PAD エディタと C 言語（独自のエディタで編集を行う）のどちらからでもプログラムの編集を行うことが可能で，編集した内容はもう片方のエディタに転送を行うことによって，反映される．石田らはこのシステムを教育現場に取り入れることによって，受講者のアルゴリズム作成技術の向上を目指している．しかし，このシステムは未だ完成には到っておらず，評価実験も行われていない．

また，岡田ら[5] は日本語プログラミング環境「ことだま on Squeak」を用いて，プログラミング入門教育を行ったあと，Java で同じ内容を行うという授業を実施した．岡田らは，最初に論理的に考える力を養った後，プログラムを作成する力を養うという授業を目指した．しかし，この授業は「ことだま on Squeak」を用いたプログラミング学習から Java に

Vol.2012-CE-113 No.11
2012/2/5

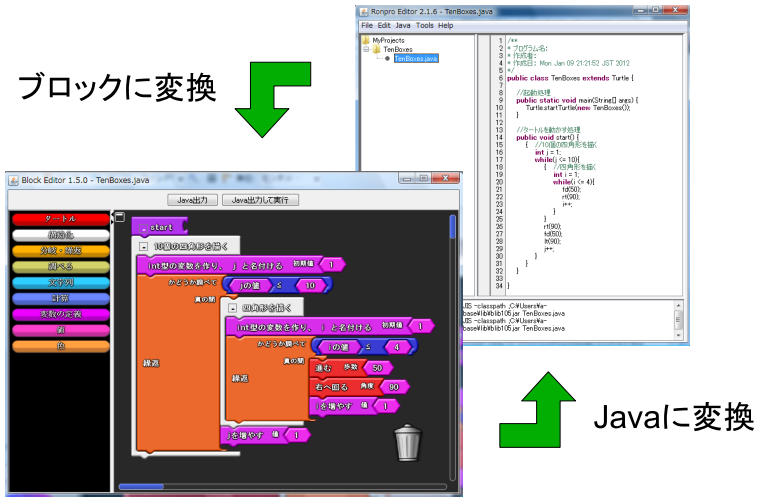

図 1 BlockEditor の外観

よるプログラミング学習へ，シームレスな移行になっていない．

プログラムの構造化手法は 30 年以上も前から利用されている．花田[6] は構造化手法の特徴として，A. データ構造の階層化 B. 処理構造の記述 C. 制御構造の記述 D. データと処理との関係を明記する，の 4 点を挙げている．これらの特徴を理解することによってプログラムを容易に理解できると川島[7] は述べている．

3. BlockEditor の設計

本章では，開発した「BlockEditor」と Java 言語の相互変換，プログラムの構造化支援，システムのユースケースの 3 点について述べる．

3.1 BlockEditor と Java 言語の相互変換

開発した「BlockEditor」の外観を**図 1** に示す．図 1 の左下の画像が BlockEditor を使用している様子である．BlockEditor はエディタの右下の作業ウィンドウでプログラムを作る．プログラムを作るときは，左のウィンドウにある各カテゴリからブロックを取り出し，作業ウィンドウでブロックを組み立てる．Java 言語に変換するときは，エディタの上部にある「Java 出力」というボタンを押すと，BlockEditor で組み立てたプログラムが Java 言語に変換され，初学者向けに開発された Java 開発環境（以下，Java エディタ）に出力される．図 1 の右上の画像が Java エディタである．「Java 出力して実行」を押すと，Java 言語が出力され，組み立てたプログラムが実行される．Java 言語から BlockEditor に変換するときは，Java でプログラムを実装し，Java エディタで Java ソースコードを保存すると，BlockEditor に変換される仕組みになっている．

```
1  public class TenBoxes extends Turtle {
2
3      public void start() {
4          { //10 個の四角形を描く
5              int j = 1;
6              while(j <= 10){
7                  { //四角形を描く
8                      int i = 1;
9                      while(i <= 4){
10                         fd(50);
11                         rt(90);
12                         i++;
13                     }
14                 }
15                 rt(90);
16                 fd(50);
17                 lt(90);
18                 j++;
19             }
20         }
21     }
22
23 }
```

図 2 変換された Java ソース例

3.1.1 ブロックから Java 言語への変換

BlockEditor で組み立てられたプログラムを Java 言語形式のソースコードに変換する．この変換システムによって，BlockEditor で構築したプログラムが Java 言語ではどのように記述されるかが受講者に提示される．内部的に，BlockEditor で組み立てられたプログラムは XML 形式で出力される．本システムではこの出力された XML 形式を変換することで，Java 言語のソースコードを出力する．

例として「10 個の四角を描くプログラム」の BlockEditor の変換例を示す．**図 2** に示された Java 言語のソースコードは図 1 で示されている BlockEditor で実装されたプログラム

を変換したものである．

図 1 の BlockEditor で実装されたプログラム中に使われているブロックで,「繰返」ブロックは while 文に変換される．プログラミング教育カリキュラムの前半は Java で実装されたタートルグラフィックス[8] を採用している．タートルグラフィックスで実行するために，Java 言語において (1)「右へ回る」ブロックは rt，(2)「進む」ブロックは fd と変換される．

3.1.2 Java 言語からブロックへの変換

Java 言語で実装されたソースコードを BlockEditor のプログラムに変換する．この変換システムによって，Java 言語で構築したプログラムが BlockEditor ではどのようなプログラムに実装されるのかが受講者に提示される．内部的には，Java 言語形式のソースコードを一旦抽象構文木に変換し，そこから BlockEditor 形式の XML ファイルに変換している．

3.2 プログラムの構造化支援システム

3.2.1 抽象化ブロックによるプログラムの構造化

我々は，プログラムを構造的なまとまりごとに分けてその構造の目的を記述することができるブロック[9]（以下，抽象ブロック）を提案している．Scratch に実装したものと同様のものを本システムにも実装した．図 1 の BlockEditor のプログラムには抽象化ブロックが使用されている．抽象化ブロックの中に他の命令ブロックを構築することが出来る．ブロックの上バーに自然言語でブロックの中に構築したプログラムの目的を記述することができる．

3.2.2 抽象化ブロックの折りたたみ機能

抽象化ブロックは，折りたたみ中に構築したブロックを隠すことが出来る．抽象ブロックの中に構築した命令ブロックが多くなった場合や，段階的にプログラムを構築するために抽象化ブロックを入れ子にすることになったとき，抽象化ブロックを折りたたむことが出来る．**図 3** は抽象化ブロックを折りたたみ機能を利用したときの様子である．図 3 のプログラムで「四角形を描く」の目的の抽象化ブロックを折りたたんでいる．

3.2.3 メソッドの学習支援

抽象化ブロックは，ある目的のために作られたプログラムをひとまとめにするためのブロックである．抽象化ブロックには折りたたむことによって，抽象化ブロックの中に実装したブロックを隠す機能がある．この機能を利用することで，抽象化ブロックは関数と同じ働きをすることになる．BlockEditor を使った構造化教育支援はテキスト型言語のメソッドの概念の学習を支援できる．

3.2.4 Java ソースコードにおける表現

抽象化ブロックを Java ソースへの展開は，空ブロックを使って実現した．空ブロックの中にそれぞれの構造のソースコードを記述し，その空ブロックの始まりの括弧の右隣に構造の目的をコメントで記述する．空ブロックの中に別の空ブロックを記述できるようにすることで，入れ子構造を表現している．

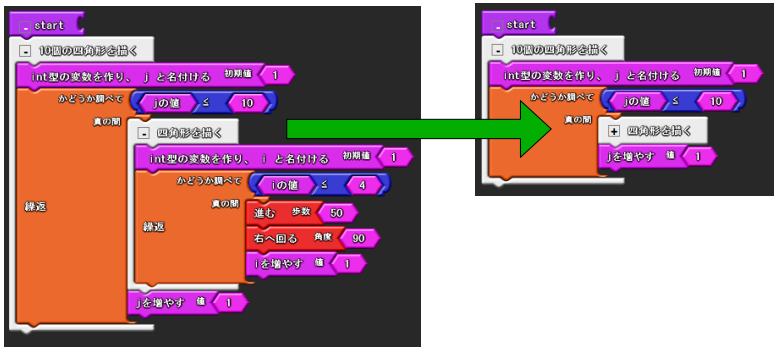

図 3 抽象化ブロックの折りたたみ機能

3.3 BlockEditor の日本語化

OpenBlocks 標準のブロックは全て英語表記であった．日本人にとってプログラムが読解しやすくなるようにするため，ブロックの日本語化を行った．

3.4 ユースケース

本システムは以下の 3 つのユースケース（以下 UC: Usecase）を想定している．

UC1 ブロック型言語から Java 言語への移行を目的とする支援 : アルゴリズムの基本構造といったプログラミングの基礎知識を学習し理解するまでは，BlockEditor を利用してプログラムを実装し，基礎知識を理解した後は徐所にユーザの任意で Java でのプログラムの実装へと移行する．

UC2 プログラムの構造化を目的とした学習支援 : Java 言語でプログラムの実装を行うと，プログラムの構造化を考慮しながら，実装することに不慣れであると感じているユーザが，プログラムの構造化を目的に学習を行う時に使用する．

UC3 プログラムの構造を確認するための支援 : Java 言語でプログラムを実装している最中，もしくは一通りプログラムを実装し終えた後，Java ではプログラムの構造を把握するのが困難であるとき，BlockEdiotor を利用して実装したプログラムの構造を把握する．

UC1 は，Java によるプログラムでは，初学者がコンパイルエラーに惑わされ，基本知識

表 1 実験の実施環境

対象学部	情報学部社会学科 (文科系)
対象授業	プログラミング (第 8 回)
実験実施日	2011 年 12 月 1 日
受講者数と学年	107 人（1 年 : 106 名 2 年 : 1 名）
授業時間数	90 分× 2 コマ
アシスタント	7 名

表 2 カリキュラム

授業回	比較対象	内容
第 1 回		環境設定
第 2 回		タートルグラフィックスの基本的な命令
第 3 回		変数，条件分岐
第 4 回	○	繰り返し，ブロックの入れ子
第 5 回		コンソール入出力プログラム
第 6 回-第 7 回		アニメーション作品の制作
第 8 回	○	BlockEditor 使用実験
第 9 回-第 11 回		メソッド（引数，戻り値），再帰
第 12 回-第 15 回		集合データ構造，アルゴリズム

を学ぶことが出来ず，プログラミングに対して苦手意識を持ってしまうという仮説に基づくものである．ユーザがプログラムの基本構造を学習し，プログラミングに対する苦手意識が徐々に解消され，Java でのプログラムの実装へと移行することを目的に支援を行う．

UC2 は，ユーザが BlockEditor にある抽象化ブロックを利用してプログラムの実装を行うことで，プログラムの構造を考慮しながら実装を行い，その手段を確立するという仮説に基づくものである．プログラムの各所の目的を正確に把握出来るようになることを目的に支援を行う．

UC3 は，ユーザが Java で実装したプログラムをブロックに変換することで，プログラムの構造を抽象化ブロックを利用しながら構造を把握し，誤りがないか確認を行うという仮説に基づくものである．Java を利用してプログラムを実装しているユーザが，実装途中でも容易にプログラムの構造を把握することが出来ることを目標とする．

4. 実 験 方 法

4.1 実 施 環 境

著者所属学部で 2011 年度 1 年次後期に行われている，文科系の学生を対象にしたプログラミング教育を行う授業「プログラミング」のうち，1 回分 (2 コマ=180 分) の授業を使って実験を行った．実施環境の詳細を**表 1** に示す．この講義は 107 名の学生が受講しており，受講者のサポートをする 7 名のティーチングアシスタントが配置されている．利用する言語は Java 言語である．

次に,「プログラミング」の全体のカリキュラムを**表 2** に示す．表 2 には授業回とその回に行ったプログラミングの内容が記されている．第 1 回から第 4 回の授業ではタートルグラフィックスを利用してプログラミングの基礎を学習する．特に第 4 回の授業では，繰り返しの入れ子構造について学習する．第 5 回でコンソールの入出力を学習する．第 6 回と第 7 回の授業でアニメーション作品の制作を行う．

実験は「プログラミング」の第 8 回の授業で行った．講義の冒頭で BlockEditor の使用方法やユースケースの解説を行い，課題を提示した．課題は繰り返しの入れ子構造になるプログラムの BlockEditor を利用して実装する問題である．第 8 回の授業までの回で学習した事柄を利用すれば解決できる課題である．

本実験では，第 4 回の講義と第 8 回の講義を比較し，実験の検証を行った．

4.2 第 4 回授業で出題した課題

第 4 回授業では，受講者に 6 つの課題を提示した．課題の内容の詳細は付録に記す．受講者は課題を Java 言語を利用して実装した．

課題 1 と課題 2 は繰り返しを使って実装する課題である．繰り返しの入れ子構造になるプログラムではない．課題 3〜課題 6 はプログラムが繰り返しの入れ子構造になるプログラムである．

4.3 第 8 回の授業で出題した課題

第 8 回授業では，受講者に 4 つの課題を提示した．課題の内容の詳細は付録に記す．受講者は課題を BlockEditor を利用して実装した．

課題 1 は受講者に BlockEditor に慣れてもらうための問題である．繰り返しの入れ子構造になるプログラムではなく条件分岐を利用したプログラムとなるため，検証対象の課題ではない．課題 2〜課題 4 は繰り返しの入れ子構造になるプログラムになる．これらの課題を検証対象の課題とした．

課題 2〜課題 4 で抽象化ブロックを利用してプログラムを実装することを想定している．かつ，プログラムが抽象化ブロックの入れ子構造になり，プログラムの目的が明確になることで，プログラムの構造把握が容易になり，プログラムの実装時間が第 4 回のプログラムの実装時間よりも短縮される事を想定している．

プログラミングに対する意識	JavaとBlockEditorではどちらが理解しやすいか 無回答	5. BlockEditor	4. やや BlockEditor	3. どちらでもない	2. ややJava	1. Java	総計
1. 得意				1		2	3
2. やや得意						1	1
3. 普通	1	3	7	3	5	2	21
4. やや苦手		1	5	2	4	4	16
5. 苦手		4	9	8	6	1	28
無回答				1		1	2
総計	1	8	21	15	15	11	71

図 4 BlockEditor によるプログラムの理解度

現在利用したいのは？	将来利用したいのは？ 4. 両方	3. どちらでもない	2. BlockEditor	1. Java	総計
1. Java	2			34	36
2. BlockEditor	1		6	15	22
3. どちらでもない		1	1		2
4. 両方	7		1	3	11
総計	10	1	8	52	71

図 5 Java vs BlockEditor

4.4 アンケート調査

実験終了後，アンケート調査を実施した．実施したアンケートは以下の設問からなる．

- プログラミングに対する意識（問 1）
- Java と BlockEditor で現在利用したいのはどちらか（問 2-1）
- Java と BlockEditor で将来利用したいのはどちらか（問 2-2）
- Java と BlockEditor のどちらを利用したほうがプログラムが理解しやすかったか．(問 3)
- BlockEditor の使用性とその理由（問 4）

5. 実験結果

5.1 アンケート調査結果

受講者 107 名中 71 名から回答を得た．本節ではその結果を述べる．

5.1.1 BlockEditor によるプログラムの理解度

アンケートの問 1 と問 3 をクロス集計することで，受講者のプログラミングに対する意識別の BlockEditor の理解度を得た．その結果を**図 4** に示す．

表 3 より，受講者の全体的なプログラムの理解度について，「BlockEditor」もしくは「やや BlockEditor」の方が理解しやすいと答えた受講者の数と，「Java」もしくは「やや Java」の方が理解しやすいと答えた受講者の数はほぼ同数であった．

プログラミングが「得意」もしくは「やや得意」だと答え，「Java」もしくは「やや Java」の方が理解しにくいと答えた受講者は 3 名 (4.2%) であった．このように答えた受講者の 1 名の問 4 の回答に「最初からこれだったらよかった．直感的に操作できない．思ったことをやる際に手間がかかる．可視性は良いと思う.」というものがあった．この受講者は Java 言語によるプログラムの実装の方が，プログラムを理解しやすいが，BlockEditor による可視性の良さについても評価していると解釈できる．

表 3 より，プログラミングが「苦手」もしくは「やや苦手」だと答え，「BlockEditor」もしくは「やや BlockEditor」の方が理解しやすいと答えた受講者が 19 名 (26.8%) であった．このように答えた受講者の，問 4 の回答に「記述のコンパイルエラーがでないので，構造で間違えていることがわかり，また構造も色分けされていて見やすくわかりやすかったから」，「視覚的に構造が見やすいので分かりやすかったです」というものがあった．そのように回答した受講者は，BlockEditor の構造把握の容易性を評価していると解釈できる．

5.1.2 どちらの言語を使用するか

アンケートの問 2-1 と問 2-2 をクロス集計することで，現在と将来で Java と BlockEditor のどちらを使用してプログラムを実装したいか，その受講者の数を得た．その結果を表 4 に示す．

図 5 より，現在かつ将来で Java を使用してプログラムを実装したいと回答した受講者の数は 34 名 (47.9%) であった．このように答えた受講者の，問 4 の回答に「文字で入力する Java になれていたから」，「思ったことをすぐに書ける Java が自分的にはやりやすい」というものがあった．その一方で，問 4 の回答に，「プログラムの視覚化という意味では大変よいツールであると感じた」，「アルゴリズムはよく理解できた」と BlockEditor に対してポジティブな意見もあった．また，自由記述の回答に「構造化の仕様は便利だと思った」，「導入的に使用すると，アルゴリズムを視覚的に理解しやすく，良いのではないかと思いました」，「プログラムの構造の可視化はありがたいと思った」，「Java の導入部分で紹介があると、構造的なプログラミングのつかみがわかりやすかったと思う。」と BlockEditor に対してポジティブな意見が 4 件あった．現在も将来も Java を使用してプログラムを実装したいと回答している受講者の中にも，BlockEditor はプログラムの構造を視覚的に理解しやすいと感じている受講者がいると解釈できる．更に，「プログラミング」の導入時のときであれば，BlockEditor を利用してプログラムを実装したいと感じている受講者もいると解釈で

きる．

現在は BlockEditor を利用してプログラムを実装したいが，将来は Java を使用してプログラムを実装したいと回答した受講者の数は 15 名 (21.1%) であった．このように答えた受講者の，問 4 の回答に「記述のコンパイルエラーがでないので、構造で間違えていることがわかり、また構造も色分けされていて見やすくわかりやすかったから」,「構成が分かりやすかったから」というものがあった．そのように回答した受講者は，BlockEditor による構造把握の容易性を評価していると解釈できる．

現在かつ将来で Java を使用してプログラムを実装したいと回答した受講者の数は 6 名 (8.5%) であった．このように答えた受講者の，問 4 の回答に「色わけがしてあったのと、たためたりしたのが便利だった」というものがあった．その学習者は抽象化ブロックの折りたたみ機能を有効に使い，その有効性を理解できたと解釈できる．

現在かつ将来で Java と BlockEditor の両方を使用してプログラムを実装したいと回答した受講者の数は 7 名 (8.5%) であった．このように答えた受講者の 1 名の問 4 の回答に「使いにくい時と、使いやすいときがあったから」というものがあった．その受講者は BlockEditor の長所と短所の両方を理解していると解釈できる．

5.2 受講者からのコメントの抜粋

受講者からのコメントを 7 件抜粋した．抜粋したものを**表 3** にまとめる．表 3 には授業で提示された課題を受講者が提出する際，任意で求められるコメントのうち，第 4 回の授業と第 8 回の授業のそれぞれにおいて特徴のあったコメントと受講者を抜粋し記載したものである．受講者は以下の 3 つのグループに分類してある．

Group1 学習者 A,B,C からなる．第 4 回のコメントが取り組んだ課題に対してネガティブなコメントを述べており，第 8 回のコメントが BlockEditor に対してポジティブなコメントを述べているものである．

Group2 学習者 D,E からなる．第 4 回のコメントが取り組んだ課題に対してポジティブなコメントを述べており，第 8 回のコメントが BlockEditor に対してネガティブなコメントを述べているものである．

Group3 学習者 F,G からなる．第 8 回のコメントで BlockEditor に対してポジティブな意見とネガティブな意見を述べているものである．

Group1 において，受講者 B の第 8 回のコメントを見ると，受講者 B は抽象化ブロックによってプログラムの構造をより理解することが出来たと解釈できる．

Group2 において，受講者 E の第 8 回のコメントを見ると，受講者 E は Java よりも BlockEditor でプログラムを実装したほうが可視性が悪いと感じたと解釈できる．

受講者 G,H のコメントで，受講者 G は第 8 回のコメントで BlockEditor が日本語化されていたことで，Java よりも BlockEditor の方を良しとしていることが解釈できる．また，

表 3 受講者のコメント

受講者	第 4 回	第 8 回
A	最後の方はプログラミングを書いているうちに，どこがどうなっているのか自分でもわからなくなってきて大変でした．自分で整理しつつ，書いていけるようになりたいと思いました．でも，頑張ってできた時はうれしいです．	やっぱりエディタを使うと，順番を考えるだけでいいし遊び感覚で楽だなと思った．java の打ち間違いとかもないし，間違いも見つかりやすくてよかった．
B	四角形や三角形を 100 個描くプログラムでは，遠回りをして考えてしまい，描き方が悪かったので奇数と偶数の分岐条件を使うことができなかった．いかに簡単にプログラムを組むかことは，非常に難しい課題だと思った．	BlockEditor を使うと，どのブロック {} に何が入っているかというのがとても分かりやすかった．長いプログラムを作っているときに BlockEditor が役に立つかもしれないと思った．
C	四角形を１０個描くプログラムと四角形を１００個描くプログラムが難しかったです．途中で嫌になりかけました．でも，四角形と三角形を交互に描くプログラムは，意外と早くできてすっきりしました．	自分で書かなくてもよかったので，自分で書くよりも修正するのが楽だった気がします．でも，なかなかできなかったです．
D	入れ子構造などどんどん難しくなっていくが，それ相応のやり甲斐もあってとても楽しく感じている．	使い方に慣れるまでブロックエディタは使いづらかった．
E	頭の中で整理するのが難しかったです．でも自分で書いたプログラムが実際に動いてくれるのは，何度見ても楽しいです．	ブロックエディタも楽しかったのですが，普通にプログラミングするより自分にはごちゃごちゃして見えたので，導入にはいいかもしれませんが，ふつうの Java に少し慣れると逆に使いづらいかなと感じました．以上です．
F	最後の三角と四角を交互に並べる課題がすごく難しかったです．	ブロックは日本語なので今までよりもわかりやすかった．while だとよく分からないけど繰り返しと書かれると自分が今どこら辺を書いているのかとかよく分かった．
G	今回の課題は１００個の図形を描くプログラムにおいて，次の図形に進む指定が上手くいかず苦労しました．繰り返しプログラムは時間短縮のためにとても便利なので，今後も繰り返しの指定ができる時は忘れないようにようにしたいと思います．	今回の課題の内容は比較的簡単なものでしたが，BlockEditor を使うと細かい修正点がパッと見て分からず少し戸惑いました．しかし，入力の手間は省けるのでその点はとても便利でした．

受講者 H の第 8 回のコメントより，受講者 H は構造の間違いではなく，スペルミスや値の間違いといった細かな修正を行うことは，BlockEditor には不向きであると主張していると解釈できる．

6. 考　　察

6.1 「UC1:ブロック型言語から Java 言語への移行を目的とする支援」の効果

5.2 節の受講者からのコメントより、BlockEditor はプログラミング学習の導入にはいいというコメントが 1 件あった。5.1.2 項より，現在も将来も Java を利用してプログラムを実装したいと回答した受講者の内の 2 名が，アンケートの問 5 の意見で BlockEditor はプログラミング学習の導入にはいいとコメントした．

以上から、「プログラミング」の導入時期に BlockEditor を投入していたならば、導入段階では BlockEditor を使う人が多く、徐所にプログラミングに慣れてくると BlockEditor から Java に移行するようになるのではないかと推測できる．これにより，設計された「UC1:ブロック型言語から Java 言語への移行を目的とする支援」の妥当性を確認できた．

6.2 「UC2:プログラムの構造化を目的とした学習支援」の効果

5.2 節の受講者からのコメントより，BlockEditor はプログラムを構造化するのに役立ったとコメントしている受講者を 1 名確認できた．また 5.1.2 項より，現在は BlockEditor を利用してプログラムを実装したいが，将来は Java を使用してプログラムを実装したいと回答した受講者のコメントにも、BlockEditor はプログラムを構造化するのに役立ったとコメントしている受講者を 2 名確認できた．

以上のことより，抽象化ブロックを使用することによってプログラムの構造を把握することができ、プログラムをうまく実装することができた受講生がいることが確認できた．これにより，設計された「UC2:プログラムの構造化を目的とした学習支援」の妥当性を確認できた．

6.3 「UC3:プログラムの構造を確認するための支援」の効果

5.1.1 節より，プログラミングが得意だと答え，Java の方がプログラムを理解しやすかったと答えた受講者の中で「BlockEditor の可視性はいい」と答えた受講者が 1 名いた．また，5.1.2 項より，今も将来的にも BlockEditor を利用してプログラムを実装したいと回答した受講者の中で，プログラムの構造を視覚的に見るのに BlockEditor が役に立つとコメントする受講者が 4 名確認できた．

以上の 5 名は，Java でプログラムの実装を行うが，プログラムの構造を確認するときは BlockEditor の方が把握しやすいと考えているのだと解釈できる．これにより，設計された「UC3:プログラムの構造を確認するための支援」の妥当性を確認できた．

6.4 そ の 他

6.4.1 BlockEditor の日本語化の効果

5.2 節の受講者からのコメントより，BlockEditor の日本語化は効果があることが分かった．表 3 の被験者 G の第 8 回のコメントから，ユーザの母国語でプログラムを実装するとプログラムの可読性が向上し，プログラムを理解する上での手助けになっている．

6.4.2 実験における反省点

5.1.2 項より抽象化ブロックの折りたたみ機能を有効に利用した受講者が 1 名いたことが分かったが，その受講者以外の受講者から BlockEditor を有効に利用したコメントは得られなかった．抽象化ブロックの折りたたみ機能によって，構造が複雑なプログラムを単純に出来るという利点を第 8 回の冒頭でうまく説明することが出来なかったので，受講者に有効に利用してもらえなかった．

また，本システムの特徴である BlockEditor と Java のどちらを使ってプログラムを実装してもよいという点である．よってプログラムを実装するとき，ユーザはどちらの言語を使って実装するのか選択権がある．しかし今回の実験では，提示したすべての問題を BlockEditor だけを使用してプログラムを実装するようにと指導してしまい，BlockEditor と Java の関連性について深く指導することが出来なかった．

7. お わ り に

本稿では，ブロック型言語と Java 言語のソースコードを相互に変換できるプログラミング初学者向けの教育支援システムについての概要とそれを用いたプログラムの構造化教育の学習効果について述べた．プログラミング初学者に対して実施した評価実験の結果，プログラミングに対して苦手意識を持つ学習者にはプログラムの構造化の手法を学習することにおいて有用であり，プログラミングに対して得意であると考えている学習者にはプログラムの構造を確認するために BlockEditor は有効なツールであるという感触を得た．

参 考 文 献

1) Dan Ingalls, Ted Kaehlei, John Maloney, Scott wallace and Alan Kay: Back to the Future: The Story of Squeak, A Practical Smalltalk Writtern in Itself `http://www.squeak.org`, Proc. of ACM OOPSLA ’97, p.318 (1997).

2) Scratch Team Lifelong Kindergarten Group MIT Media Lab: Scratch -imagine・program・share- http://scratch.mit.edu/.
3) Ricarose Vallarta Roque: OpenBlocks: An Extendable Framework for Graphical Block Programming Systems, *Master thesis at MIT* (2007).
4) 石田真樹，桑田正行：C プログラミングの学習支援に関する研究 PAD エディタを用いたアルゴリズム学習支援システムの構築-，コンピュータと教育，pp.41–48 (2000).
5) 岡田　健，杉浦　学，松澤芳昭，　大岩元：日本語プログラミングを用いた論理思考とプログラミングの教育，情報処理学会研究報告. コンピュータと教育研究会報，pp. 123–128 (2000).
6) 花田收悦：プログラム設計図法，企画センター (1983).
7) 川島秀樹：プログラミング教育における構造化チャート，保健医療経営大学紀要，No.03, pp.17–22 (2011).
8) Papert, S.: *Mindstorms: children, computers, and powerful ideas*, Basic Books, Inc., New York, NY, USA (1980).
9) 保井　元，酒井三四郎，松澤芳昭：コメントがプログラムコード実装時間と修正時間に及ぼす影響，教育システム情報学会第 35 回全国大会講演論文集，pp.167–168 (2010).

付　　録

A.1　第 4 回の課題

第 4 回授業時に学生に課した課題は以下のとおりである．(学生に示した仕上がりイメージを**図 6** に示す)

問 1　四角形を 10 個描くプログラムを作ろう．

問 2　四角形を 100 個描くプログラムを作ろう．

問 3　四角形と三角形を交互に 100 個描くプログラムを作ろう．

問 4　貝殻を描くプログラムを作ろう．

A.2　第 8 回の問題

第 8 回授業時 (BlockEditor 実験時) に学生に課した課題は以下のとおりである．(学生に示した仕上がりイメージを**図 7** に示す)

問 1　四角形を 100 個描くプログラムを作ろう．

問 2　100 個の四角形の中に三角形を描くプログラムを作ろう．

問 3　三角形のマトリョーシカを描こう．

問 4　貝殻を描くプログラムを作ろう．

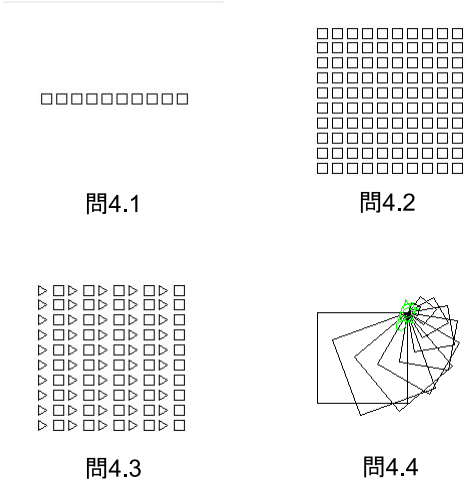

図 6　第 4 回の問題

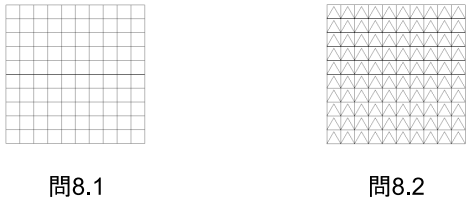

図 7　第 8 回の問題