初級者向けプログラミング言語のための
オブジェクト指向学習支援教材

Learning Support Tool for Object-Oriented Programming in a Programming Language for Beginners

丸田明日美，　塩澤秀和

Asumi Maruta and Hidekazu Shiozawa

玉川大学工学部ソフトウェアサイエンス学科，　194-8610　東京都町田市玉川学園6-1-1

Department of Software Science, College of Engineering, Tamagawa University,

6-1-1 Tamagawagakuen, Machida, Tokyo 194-8610

Abstract

Computer programming is going to be introduced in elementary and secondary education, and the concept of modeling events is in the next high school curriculum. Although the object-oriented method is widely used for modeling software structure, programming beginners are often facing difficulties to learn its basic idea. So, we have developed a learning support tool for object-oriented programming. The software visualizes relations among objects in a program made in a blocks-based programming language for programming beginners.

Keywords: programming education, object-oriented programming, software visualization, learning support system

1. 背景と目的

2020 年に向けて，初等中等教育でも情報化が進められており [1]，その中には情報教育の推進としてプログラミング教育の充実が含まれている．

小学校では，新たにプログラミング教育が必修化される．これは，「プログラミング的思考」[2]を育むことが目的とされており．プログラミング的思考とは，「コンピュータに一連の処理を行うように指示し，自分の意図した一連の活動を表現できるという体験を通して必要なアルゴリズムや文法，記号の組み合わせなどを論理的に考えていく力」と定義されている．世界的に，このような初級のプログラミング教育では，ドラッグアンドドロップ操作でプログラムを作成できるブロック型言語などのビジュアルなプログラミング言語が利用されている．

高等学校の情報科も，学習指導要領の改訂によって大きく変更される．新たな「情報 I 」の単元の 1 つである「コンピュータとプログラミング」では，プログラミグによりコンピュータを活用する力や，事象をモデル化して問題を発見したり，シミュレーションを通してモデルを評価したりする力を育むことが目的とされている [3]．

今日，ソフトウェア開発で用いられている代表的なモデル化手法は，オブジェクト指向である．オブジェクト指向は，複雑なシステムを構築する

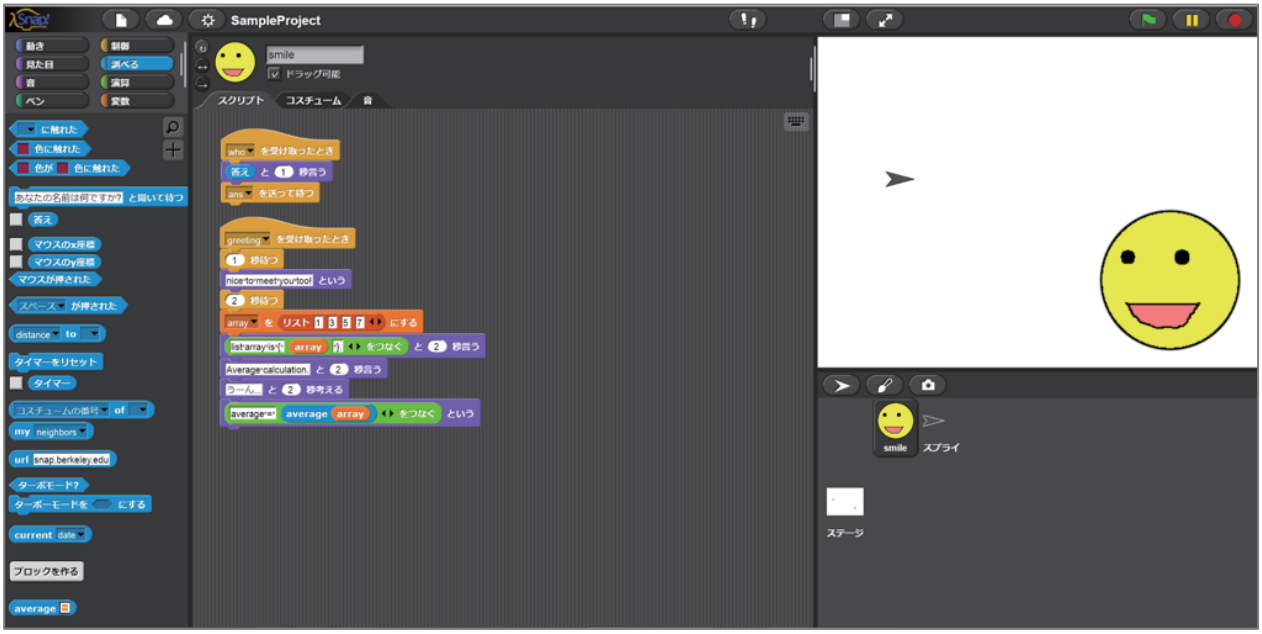


図 1　Snap! Build Your Own Blocks

上で有効なモデル化手法であり，今日のプログラミングには必要不可欠な考え方である．よって，プログラミング教育においても，学習者の発達段階に応じた形で，オブジェクト指向の要素を扱うべきであろう．しかし，プログラミング初級者の多くにとって，オブジェクト指向の考え方は習得が難しく，苦手意識を持ってしまう傾向がある．

そこで本研究では，プログラミング初級者が，プログラミング的思考を身につけ，代表的なモデル化手法であるオブジェクト指向への理解を深めることを目的として，Snap!（図 1）という初級者向けのブロック型言語で作成されたプログラムにおけるオブジェクトの関係を分かりやすく可視化する学習支援ソフトウェアを開発する．

2. 関連研究

プログラム内蔵型のコンピュータが発明され，各種産業や軍事のシステムにプログラムが必要になったころから，それを開発するためのプログラマーの養成は重要な社会問題となった．そこで，最初は大人の技術者向け，さらには子供向けとして，プログラミングの学習を支援するためのさまざまなシステムが提案された[4]．

画面上で図形を組み合わせるようなビジュアルプログラミング言語[5]もこのような要求の中から生まれた．ビジュアルプログラミング言語は，まだ大規模なソフトウェアを開発できるほどの実用性には欠けるが，プログラムの構造を図解によって表すので，子供などの初級者にも分かりやすいという利点がある．

その中でもブロック型プログラミング言語は，分岐，反復，表示，演算などの基本的な要素を，それぞれ特徴的な形のブロックで表現するものである．プログラミング初級者は，細かい文法の理解と活用に苦しむことなく，ブロックを組み立てるようにプログラムを作成することができるので，近年その開発と学習に関心が高まっている．今後，初等教育でも大半の学校がブロック型言語であるScratch[6,7]を採用すると予想される．

文献8)では，Scratchのようなブロック型言語とJavaのようなテキスト型言語を組み合わせた講義を設定し行った調査が紹介されている．この講義を受けた学生は，従来のテキスト型言語だけの講義を受けた学生に対して，Java言語の最終試験で平均10パーセント以上成績を向上させたことが報告されている．この結果により，テキスト型言

語の学習を始める前にブロック型言語による学習の時間を設けることで，式評価，制御構造，配列，クラス定義の操作などの理解をより深めることができると考えられる．

また，オブジェクト指向の学習に注目すると，図解によってオブジェクト指向の理解を支援するソフトウェアの例としては，Anchor Garden[9]が報告されている．これは，変数の型とその型を表した視覚モデルを学習者に提供し，学習者はその視覚モデルを組み合わせて，オブジェクトの構造図を作成するシステムである．作成された構造図の隣には，それに対応したソースコードが提示されるので，学習者は実際のプログラミングに対する理解を深めることができる．Anchor Gardenの評価実験では，構造図とソースコードを対比させることにより，学習者はプログラムの構造をより正確に把握することができたと報告されている．

3. 本研究の提案

プログラミング初心者に対して，ブロック型言語からプログラミングの学習を始めることの効果が認められていることから，我々は，オブジェクト指向の学習においても，テキスト型言語の学習を始める前にブロック型言語による適切な学習を行うことによって，学習者の理解がより深まるのではないかと考えた．

そこで本研究では，プログラミング初級者向けのブロック型プログラミング言語であるSnap!で作成されたプログラムのオブジェクト同士の関係を，学習者にわかりやすく可視化するソフトウェアを開発する．

Snap!は，画面構成上，プログラムの全体構造をユーザが一目で把握することが難しいが，本研究の可視化によって，プログラミング学習者が，オブジェクト指向の考え方で構成されたプログラムの構造を把握しやすくなり，その考え方につまづきにくくなることが期待される．

4. 対象とするブロック型言語

本アプリケーションは，Snap! Build your Own Blocks [10]のプログラムを対象とした．これは，Scratch の操作性や各種機能をそのまま保持した上で，自由にブロック（関数）が自作できるように拡張されたブロック型言語である．

また Snap!は，プロトタイプクラスの概念が拡張されており，オブジェクト指向によるプログラミングが可能なので，本研究の目的に適している．

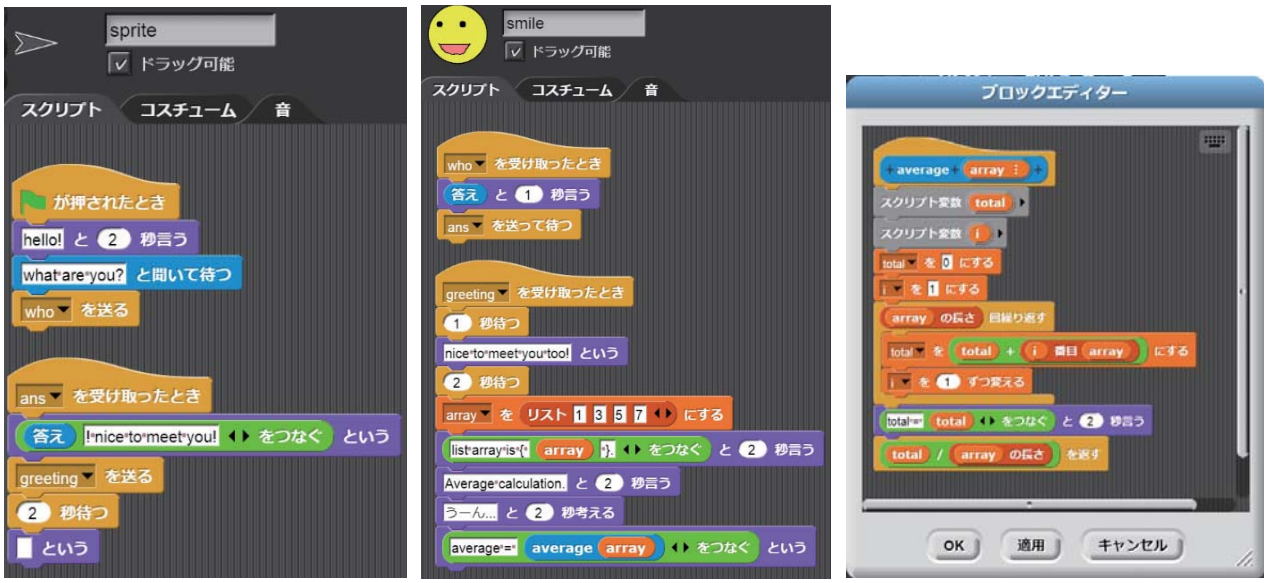
Snap!のプロジェクトは，「ステージ」（舞台）と「スプライト」（キャラクター等）から構成される．また，ステージおよびスプライトに属するプログラムコードのことを「スクリプト」と呼ぶ．さらに，ステージおよびスプライトは，相手のスクリプトを実行するために相互に「メッセージ」を送受信することができる．

Snap!のプロジェクトやその構成要素は，XMLファイル（図 2）として保存されており，ユーザはダウンロードして保存することができる．また，Snap!は Scratch をもとに拡張された言語であるため，Scratch で作成したプロジェクトを Snap! に読み込んで実行することもできる．

本論文ではアプリケーションによる可視化例を示すにあたり，図3のプログラムを用いた．

```
<?xml version="1.0"?>
<project version="1" app="Snap! 4.1, http://snap.berkeley.edu" name="SampleProject">
    <notes/>
    <thumbnail>
  - <stage name="ステージ" id="1" scheduled="false" sublistIDs="false" inheritance="true" codify="
    width="480">
        <pentrails>
      + <costumes>
      + <sounds>
        <variables/>
        <blocks/>
        <scripts/>
      - <sprites>
          - <sprite draggable="true" name="スプライト" id="8" costume="0" pen="tip" color="80,80,
              + <costumes>
              + <sounds>
                <blocks/>
                <variables/>
              - <scripts>
                  - <script y="25" x="22">
                      <block s="receiveGo"/>
                    - <block s="doSayFor">
                        <l>hello!</l>
                        <l>2</l>
                      </block>
                    - <block s="doAsk">
                        <l>what are you?</l>
                      </block>
                    - <block s="doBroadcast">
                        <l>who</l>
                      </block>
                    </script>
                  - <script y="146" x="19">
                      - <block s="receiveMessage">
                          <l>ans</l>
                        </block>
                      - <block s="bubble">
                          - <block s="reportJoinWords">
                              - <list>
```

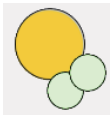
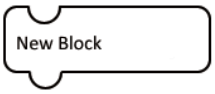
図 2　プロジェクトの XML ファイル（一部）



(a) sprite　　(b) smile　　(c) block

図 3　本論文の例で用いた Snap!のサンプルプログラム

表 1　Snap!のオブジェクトと可視化例

オブジェクト	メソッド	可視化例
ステージ	スクリプト	
スプライト	スクリプト	
ブロック	スクリプト	

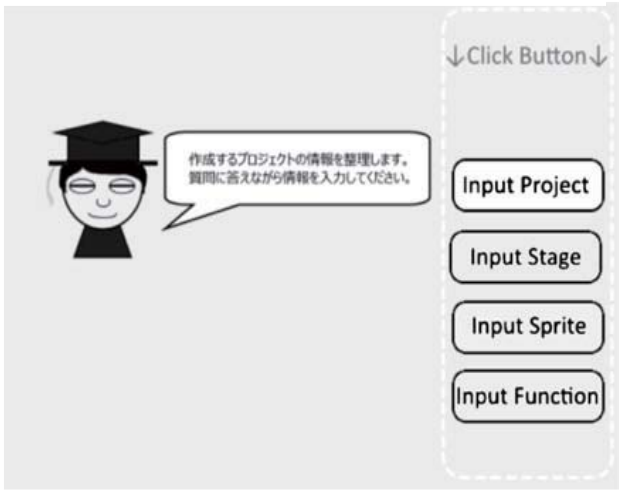


図 4　本システムの実行画面

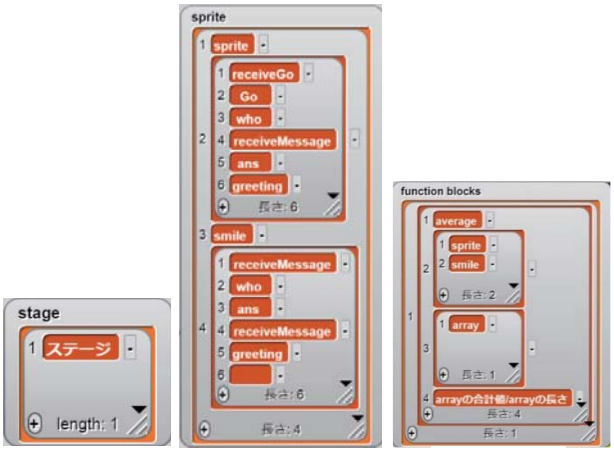
5. アプリケーションの構成と操作

本研究では, Snap!ユーザである学習者の利便性を考慮し, 教材である本アプリケーションもSnap!で作成した.

Snap!のプログラムでは, ステージ, スプライト, 自作ブロックは，オブジェクト指向におけるオブジェクトに相当し，スクリプトは，メソッドに相当するものであると考えられる．そこで，本アプリケーションでは，表 1 のような対応関係と可視化方法を採用することとした.

本アプリケーションの使用の流れは，学習者による Snap!プログラムの情報入力，アプリケーションによるプログラムの構造の可視化，メッセージの送受信関係の可視化の手順で進む.

まず，学習者がアプリケーションを起動すると図4のような実行画面が表示され，右側の「Input Project」「Input Stage」「Input Sprite」「Input Function」の4つのボタンをクリックすると各項目に応じた質問が表示される．学習者は表示された質問に答えながら, Snap!で作成したプロジェクトに関する情報を入力する．現状では，対象となるプロジェクトの構造の情報は，アプリケーションから指示された手順にそって学習者に入力してもらう.



（a） stage　　（b） sprite　　（c） block

図 5　Snap!のリストに格納された情報

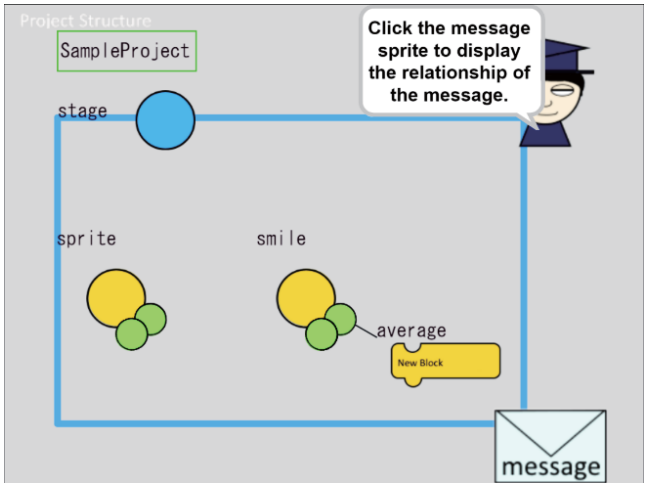


図 6　プログラムの構造の可視化

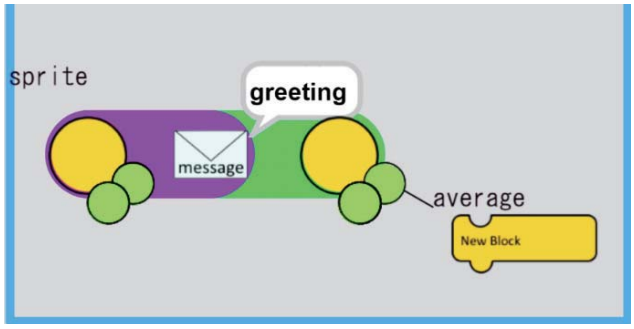


図 7　メッセージの送受信関係の可視化

入力する情報は次の通りである．

（1）　プロジェクト名

（2）　ステージ名とスクリプト

（3）　スプライト名とスクリプト

（4）　学習者が新たに作成したブロック

入力された情報は，Snap!の階層的なリスト構造で格納される．図5は，図3のサンプルプログラムをもとに入力した情報である．

情報の入力後，学習者が実行画面（図4）の左上に表示されているキャラクターのアイコンをクリックすると，入力された情報から図6のようなプロジェクト全体の可視化が表示される．ここで，オブジェクトは名前つきの大きい円で表現され，各オブジェクトの周囲にはメソッドがその名前とともに小さな円で表示される．自作ブロックは，それを持つオブジェクトと線でつなげられ，ブロック型のアイコンで表示される．

最後に，学習者が図6の右下の封筒のアイコンをクリックすると，図7のように送信オブジェクトから受信オブジェクトへのメッセージを表す封筒のアイコンが動き，学習者はメッセージの送受信関係を理解することができる．

6. 考察と課題

学習者は，Snap!で作成されたプログラムを見ながら本アプリケーションを使用することにより，そのプログラムが複数のオブジェクトで成り立っていることが一目で分かる．また，メッセージがオブジェクト内で送信される様子を見ることができるので，その役割と送信・受信の関係を直感的にとらえることができる．

よって，ブロック型言語である Snap!に本アプリケーションを補助教材として組み合わせてオブジェクト指向について学んでから，Java などのテキスト型言語へ移行するようにすれば，最初からテキスト型言語のみを学ぶよりもオブジェクト指向への抵抗感が抑えられると期待される．

今後の課題としては，オブジェクト指向の重要な概念である継承関係（クローン関係）の可視化がある．Snap!では，定義されたスプライトをクローン（再利用）し，新しいスプライトを作成することができる．本アプリケーションでは，現状ではプロジェクトの全体構造のみを可視化しているが，オブジェクト指向の学習を支援する意味で

は継承の可視化が是非とも必要であろう．

他にも，Anchor Gardenのように可視化に対応したテキスト型のコードを隣に表示する機能を追加し，学習者がそれらの関係を観察できるようにすれば，初級者向けのブロック型言語から中級者以上が用いるテキスト型言語への移行に対する学習支援になると考えている．

また，現在はプログラムの情報を手作業で入力する必要があるが，実用的には自動化することが望ましい．図8はProcessingを用いてXMLファイルから構造を読み出して可視化したものである．現在, Snap!の環境の中で同等の機能を実現できるように改良を試みている．

7. まとめ

本研究では，ブロック型言語 Snap!で作成したプロジェクトのデータ構造の可視化を行うアプリケーションと，オブジェクト指向の理解を支援するために Snap!でプロジェクトの構造を可視化するアプリケーションの開発を行った．学習支援を目的としたアプリケーションでは，現在，Snap!プロジェクトの構造とメッセージの送受信関係の可視化が実現されている．

今後は，Snap!プロジェクトの情報をより多く視覚的に表現できる方法を検討しアプリケーションを改良していきたい．また，アプリケーションの有用性とブロック型言語によるオブジェクト指向学習の学習効果を評価する実験を行っていきたいと考えている．

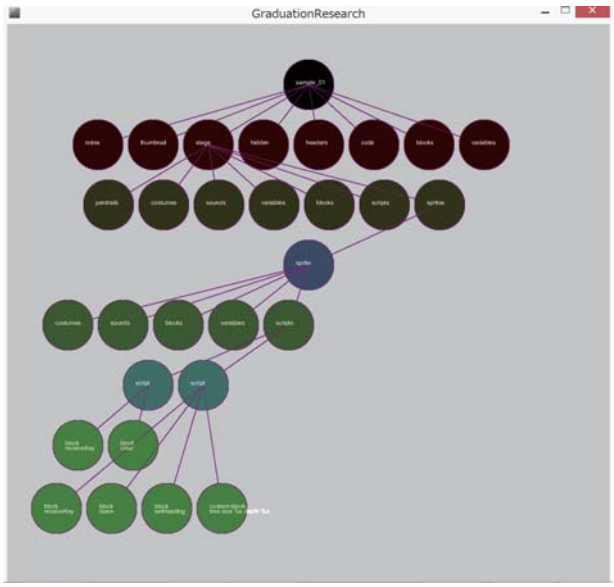


図 8　Processing による可視化

参考文献

1）文部科学省 2020年代に向けた教育の情報化に関する懇談会：中間とりまとめ，(2016).

2）文部科学省 小学校段階における有識者会議：小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ），(2016).

3）中央教育審議会：幼稚園，小学校，中学校，高等学校及び特別支援学校の学習指導要領等の改善及び必要な方策等について（答申），(2016).

4) E. Lemut, B. du Boulay, G. Dettori (Ed.): Cognitive Models and Intelligent Environments for Learning Programming, Springer-Verlag, (1993).

5) 田中二郎：ビジュアルプログラミング, bit別冊 ビジュアルインタフェース (平川正人, 安村通晃 編)，第2章 2.2, 共立出版, (1996).

6) NHK Eテレ：Why!? プログラミング, (2016より放送)．

7) Lifelong Kindergarten Group (MIT Media Lab): Scratch, (2006). https://scratch.mit.edu

8) D. Bau, J. Gray, C. Kelleher, J. Sheldon, F. Turbak: Learnable Programing: Blocks and Beyond, Communications of the ACM, Vol.60, No.6, pp.72-80, (2017).

9) 三浦元喜，杉原太郎：オブジェクト指向プログラミングの概念理解を支援する Anchor Garden システムの評価，情報処理学会インタラクション 2013 論文集, (2013).

10) M. Jens, B. Harvey: Snap! Build Your Own Blocks, (2011). http://snap.berkeley.edu

2018年2月28日原稿受付，2018年3月14日採録決定
Received, February 28, 2018; accepted, March 14, 2018