オブジェクトストーム：オブジェクト指向言語による初中等プログラミング教育の提案

中谷 多哉子[†] 兼宗 進[††] 御手洗 理英[†††]
福井 眞吾[††] 久野 靖[††]

初中等教育での情報教育の準備が進められている．オブジェクトストームとは，コンピュータへの興味を持たせつつ，楽しませながら，オブジェクト指向プログラミングを介してコンピュータの基本的な原理を理解させる情報教育コンセプトである．ドリトルは，オブジェクトストームに基づいて開発された日本語オブジェクト指向プログラミング言語である．ドリトルは，オブジェクトの効果的な見せ方の１つとして，タートルグラフィックスを含む図形環境を提供しており，簡潔で分かりやすい文法を持っている．本稿では，中高校生を対象に行った実験授業の成果を示すとともに，オブジェクトストームの概念を紹介し，ドリトルの教育効果と可能性について議論する．

Objectstorms: An Idea for K12 Programming Education in Object-oriented Language

Takako Nakatani,[†] Susumu Kanemune,[††] Rie Mitarai,[†††]
Shingo Fukui[††] and Yasushi Kuno[††]

Information education for K-12 (kindergarten and 12-year-education) will start from 2002. In this paper, we propose *Objectstorms*, the concept for K-12 programming education, and evaluate its educational effectiveness. The method is supported by object oriented programming language named *Dolittle*. Through *Dolittle* programming experiences, students learn the basic concepts of computer. Besides information education, *Dolittle* can be applied to various subjects on phenomena with rules, such as physics and mathematics.

1. はじめに

教育課程審議会が 1998 年 7 月に発表した教育課程の基準の改善に関する答申には，「小学校，中学校および高等学校を通じ，各教科などの学習においてコンピュータなどの積極的な活用を図ること」とし，各学校段階ごとに，以下の学習の指針が記されている[14]．

- 小学校
  様々な時間でコンピュータなどを適切に活用することを通して，情報化に対応する教育を展開する．
- 中学校
  技術・家庭科の中でコンピュータの基礎的な活用技術の習得など，情報に関する基礎的内容を必修とする．
- 高等学校
  情報手段の活用を図りながら情報を適切に判断・分析するための知識・技能を習得させ，情報社会と主体的に対応する態度を育てることなどを内容とする教科「情報」を新設し必修とする．

ある国立大学教育学部附属中学の調査によると，1 年生 120 人のうち，キーボードに初めて触れるという生徒も皆無ではなかったが，インターネットは 4 割，ワープロについては 8 割が経験者であったという．すでにコンピュータを利用すること自体は，家庭生活にも深く入り込んでいると考えてよいだろう．このような状況では，より高度にコンピュータと関われる授業を企画してもよいはずである．コンピュータとの高度な関わりとは，プログラミングのように，コンピュータが動作する基本原理に直接触れ，コンピュータを操作し発展させることを指す．

シーモア・パパートは，マインドストーム[15] の中で，「（CAI のようなシステムを使って）コンピュータ

† 有限会社エス・ラグーン
SLagoon Co., Ltd.
†† 筑波大学大学院経営・政策科学研究科
Graduate School of Systems Management, University of Tsukuba
††† 株式会社アーマット
Armat Corporation

に子供をプログラムさせたいのか，それとも子供にコンピュータをプログラムさせたいのか」という問いを投げかけている．もちろん，我々がやるべきことは子供にコンピュータをプログラムさせることである．

一般的な教育の方法には，話す，見せる，自分でやらせるという3つの方法があるといわれている．これを情報教育に適用すると，教師が何かのソフトウェアについて話し，そのソフトウェアを使ってみせ，生徒たちがそれを使ってみるという教育になろう．このような教育の目的には，コンピュータの最も基本的な原理，すなわちコンピュータがプログラムによって初めて意味を持つ機械であるということを理解させることは含まれないかもしれない．しかし，この基本原理こそ，コンピュータが他の家電製品と異なる点である．

コンピュータは，それを分解して中を見たり，CPUに触れたりしたところで，その中に組み込まれている論理回路やマイクロチップの動作，アルゴリズムなどの存在を直感的に理解することはできない．さらに，ソフトウェアの多くは，これらの仕組みを利用者に意識させないように作られている．ここに，コンピュータの基本原理を理解する難しさがある．

プログラミングは，情報教育における「自分でやらせる」という教育の3番目の方法に該当する．我々は，プログラミングという体験を通してのみ，コンピュータの基本原理を理解できると考える．

それでは，大学などで行われている手続き型言語によるプログラミング教育を，初中等教育に適用することは可能だろうか．身近なプログラミング言語の中には，プログラミングを始める前に言語の複雑な文法を理解し，アルゴリズムを組み立てるといった前提知識や準備が必要であるものが多い．さらにプログラミングに英字が使われている点も，初中等教育の学習者にとっては高いハードルとなる可能性がある．

そこで我々は，初中等教育における情報教育のために，「適切に設計されたオブジェクト指向言語を用い，生徒たちが関わる様々な現実の問題とリンクされたオブジェクトを取り扱うことを通じて，コンピュータの基本原理やその問題解決への適用について学ぶ」という情報教育コンセプトを考案し，プログラミング言語の検討，開発を行った．この情報教育コンセプトを「オブジェクトストーム」と名付ける．この名前は，パパートの「マインドストーム」のオブジェクト指向版，というつもりである．オブジェクトストームを実現するプログラミング言語は，次のような条件を備えている必要があろう．

- プログラミングを行って楽しさを味わえるために，たとえば，プログラミング後，速やかに自分のプログラミングの成果を確認できる仕組みを持っていること．
- 複雑なプログラミング言語の概念を学ばずに言語を理解でき，プログラミングが始められること．
- 現実世界の事物や事象に対応させたメタファとしてプログラムの構成要素を理解しやすいこと．
- プログラムの個々の構成要素に対して，自由に新しい動作を与えることができて，プログラムの世界を拡張できるようにするために，オブジェクト指向言語であること．
- 親近感を持たせるために，アルファベットではなく，かな，漢字によるプログラミングが可能であること．

本稿では，次の章で，オブジェクトストームをマインドストームと比較して議論し，3章で，オブジェクトストームを実現する教育用言語として開発したドリトルの特徴を，他のプログラミング言語と比較して議論する．4章では，プログラム例を示しながらドリトルの言語仕様の一部を紹介する．そして，5章では高校1年生と中学2年生を対象に行った実験授業の内容と成果を報告し，6章でプログラミング教育におけるドリトルの優位性について議論する．

2. マインドストームからの発展

パパートはマインドストーム[15]の中で，「子供がコンピュータをプログラムし，そうする過程で，最も進んだ強力な科学技術の産物を統御するという実感を得るとともに，科学，数学，そして知性のモデルを作る学問などからくる深遠な理念と密接な関係を確立する」ことを目指した．また，「普通の子供なら誰でも話すことを覚える．それならば，子供がコンピュータと話すことを覚えてはならない理由はあるまい」とも主張している．

パパートの教育理念を引き継ぐオブジェクトストームは，子供たちに，コンピュータの内部の基本原理を論理的，体感的に理解させることに重点を置く．ただし，ここで理解されるコンピュータの基本原理とは，ハードウェア構成や各構成要素の役割ではなく，プログラムが上から下へ実行されるといった，決められた約束に従って統一的にコンピュータが動作することを指す．子供たちには，コンピュータ内部のオブジェクトと話し，オブジェクトに言葉を教え，あるいは新しいオブジェクトを作る楽しみを体験しながら，コンピュータの基本原理を理解してもらいたい．

また，ゲーム機，自動洗濯機，ビデオ機器など我々

の身の回りにある家電製品にどのようにコンピュータが関与しているのかを理解するためには，自分たちがプログラミングによって創り出した作品を「もの」として操作する体験も必要である．そのためにも，言語はオブジェクト指向である必要がある．

我々は，オブジェクトストームの目的を達成するためのプログラミング言語として,「ドリトル」と名付けた日本語オブジェクト指向プログラミング言語を開発した．ドリトルは，様々なオブジェクトを創り，集わせ，オブジェクトの個性を育てるという現代的なプログラミング手法を用いて，子供たちが個別にオブジェクトの世界を構築できる仕組みを提供する[6),7)]．

3. ドリトルの開発

3.1 他のプログラミング言語の検討

情報教育用言語を開発するにあたり，既存言語に対してオブジェクトストームの教育用言語という観点から評価を行った．たとえば VisualBasic では，簡単に GUI を構築することができ，また，その GUI 部品の動作内容を記述する際には，様々なオブジェクトを参照し活用することができる．しかし，これらのプログラミングの内容は既存のオブジェクトや既存のメソッドを利用するだけであり，新しい種類のオブジェクトを作ったり，既存のオブジェクトにメソッドを追加したりすることは熟練者でなければ困難である．これでは,「初心者がオブジェクトを自由に成長させていく」というオブジェクトストームの目標を達せない．

例示プログラミング（Programming by Demonstration）では，ユーザがプログラムを明示的にコンピュータに与えるのではなく，入力データとそれを処理した結果をコンピュータに示す．コンピュータは入力されたデータの組みから，処理に使われるアルゴリズムを推論し，プログラムを自動的に生成する[1)]．このようなプログラミングの方法は，コンピュータにものを教える楽しみを味わえるかもしれないが，コンピュータの基本原理を学ぶというオブジェクトストームの目的を達成できない．

多くの既存言語では，そのプログラムに用いることができる文字が原則としてアルファベットである点にも問題がある．アルファベットは小学生には不向きである．日本語プログラミング言語については，1980 年代中頃から数多く研究されてきた[11)]．MIND[10)] はスタックを用いた日本語プログラミング言語であり，実用システムをワープロ感覚で記述できる．たとえば，文のキーとなる主語，引数，述語の語尾には，自由な語を付加することが許されており，結果的に通常の日本語と無理なく対応付けることができる文法を持つ．しかし，MIND は手続き型言語であり，オブジェクトを扱うという概念を持っていない．

3.2 タートルグラフィックスからの出発

タートルグラフィックスで有名な LOGO も教育用言語である．LOGO は，タートルグラフィックスを通して，子供が図形や空間などの知識を自発的に発見することを狙った言語であり，体験を通して子供たちの理論的な思考を育んだ[15)]．しかし，タートルが動いた軌跡は単なる描画された線にすぎない．ここに LOGO の限界があったように思う．我々は，タートルが歩いた軌跡もオブジェクトにすることを考えた．

この機構を取り入れると，タートルの活動を，視覚的なオブジェクトを新しく生成するための手段として使えるようになる．たとえば，作られた図形オブジェクトに，マウスクリックのイベントを受け付けられる機能と，そのときの動作を与えれば，図形オブジェクトにボタンオブジェクトと同様の動作をさせることができるようになる．

3.3 言語の検討

オブジェクトストームを実現する言語として，ドリトルが提供する機構と言語の特徴を以下に示す．

- 子供がコンピュータに指示して動作させることを視覚的に表現する機構：オブジェクトの効果的な見せ方の 1 つとして，タートルグラフィックスを含む図形環境を提供する．
- 世界が「もの」から成り立っていることを理解する機構：ドリトルの基本概念にオブジェクト指向を採用し，タートルが描いた図形をオブジェクトとして取り扱う仕組みを提供する．
- 日本語を使える機構：プログラミングに用いる文字をかな，漢字としただけでなく，プログラミング言語の構文を「主語 引数… 述語☆」という語順にすることで日本語の語順に近付ける．また,「」や『』など，プログラムで使う記号も日本語文章で馴染みのあるものを使う．
- 予約語，複雑な構造の排除：オブジェクトストームの目的は，プログラミング言語そのものを教えることではないから，言語の文法に多くの教育の時間が割かれるのは避けたい．そこで，ドリトルは予約語をできるだけ排除する方針を貫いている．たとえば，Java[3)] や C++[16)] には，this，super，if，while などの予約語が使われているが，ドリ

☆ オブジェクト指向では，主語はオブジェクトに，述語はメッセージ名にそれぞれ対応する．

トルでは，これらの予約語は使わない．また，分岐や繰返しといったプログラム構造は，プログラムの構造を複雑にする．そこで，これを避けるために，Smalltalk[2] で提案されたブロックを導入し，分岐や繰返しの意味をメッセージ送信式で表現できるようにした．ブロックを導入することによって，分岐や繰返しの意味を表現するための予約語や，そのための特別な構文を排除することもできた．

- プロトタイプ方式[17] に基づくオブジェクト指向の導入：オブジェクトストームを実現するための言語は，画面上や現実世界の「もの」に直接対応した概念を扱えるようなものである必要がある．これはすなわち，オブジェクト指向言語を意味する．C++，Java をはじめとした多くのオブジェクト指向言語はクラス方式であるが，クラス方式の言語を使いこなすにはクラス，インスタンス，継承などの概念を理解する必要がある．これらの概念を，プログラミングを始める前に理解しなければならないのでは，生徒にとっても辛いだろう．オブジェクト指向言語の方式には，Self 言語[17] などに代表されるプロトタイプ方式と呼ばれるものもある．プロトタイプ方式の言語にはクラスがなく，継承はオブジェクト間の委譲関係として定義される．委譲関係はオブジェクト生成時に自動的に定まるようにできるため，プログラマが別途継承を定義しなくても，その機構を使うことができる．そのため，プログラマが学ぶ概念を少なくでき，オブジェクトストームの目的にもよく合致する．これらの分かりやすさという利点から，ドリトルはクラス方式ではなく，プロトタイプ方式のオブジェクト指向言語として設計された．
- 簡潔であること：プログラミングを始める際に，様々な変数を定義するといった準備が必要な言語では，プログラミングを行い，すぐに動かして成果を確認するというプログラミングスタイルや，楽しみを味わいながらプログラミングを行うというオブジェクトストームの目標を達成できない．ドリトルは簡潔（terse）である．
- OS 非依存であること：様々な学校環境で使われることを想定すると，教育用プログラミング言語はマシン非依存，OS 非依存でなければならない．ドリトルは Java 上で実現されているため，多様な稼働環境に適用させることができる．

```
プログラム ::= (文 '。')…
文 ::= [ 変数 ' =' ] 式
変数 ::= [ 項 ' :' ] 名前
式 ::= 単純式 | 送信
送信 ::= [ 項 ]'!' 電文
電文 ::= 単純式… 名前 (['';'] 単純式… 名前 )…
括弧 ::=' (' 中置式 ')'|'(' 送信 ')'
単純式 ::= 数値リテラル | 文字列リテラル | 括弧 | ブロック
ブロック ::='「'['|' 名前… '|'] 文 ('。' 文 )… '」'
中置式 ::= 中置式 演算子 中置式 | 項
項 ::= 単純式 | 名前
数値リテラル ::= [ ± ] 数字… [. 数字…]
文字列リテラル ::="文字…"| 『文字…』
```

```
註 1：'[' と '「','」' と '」','。' と '. ' の各文字は
同一の意味をもつ文字として解釈される。
なお，ドリトルでは，16 ビットコードの文字 (いわゆる全角) と
8 ビットコードの文字 (いわゆる半角) とを区別しない。
註 2：そのほかに初期値を持つ変数として下記を使える。
真，偽，黒，赤，緑，青，黄，黄色，紫，水，水色，白
```

図 1 ドリトルの構文
Fig. 1 Syntax of *Dolittle*.

```
1: カメ太 = タートル! 作る。
2: カメ太! 100 歩く 120 右回り。
3: 赤三角形 = カメ太! 100 歩く 120 右回り 100 歩く 図形にする。
4: 赤三角形! (赤) 塗る。
5: 赤三角形：ぐるぐる =「! 15 右回り」。
6: 赤時計 = タイマー! 作る。
7: 赤時計! 1 間隔 10 時間。
8: 青三角形 = 赤三角形! 作る。
9: 青三角形! (青) 塗る 30 100 移動する。
10: 青時計 = 赤時計! 作る。
11: 青時計! 0.5 間隔。
12: 赤時計! 「赤三角形! ぐるぐる」 実行。
13: 青時計! 「青三角形! ぐるぐる」 実行。
```

図 2 ドリトルのプログラム例
Fig. 2 An example program in *Dolittle*.

4. ドリトルのプログラム例

4.1 構文規則

図 1 にドリトルの構文規則を示した．ここで，**図 2** に示したプログラムの例を順に追いながら，ドリトルの構文規則を紹介する．

1: "!" は，その前に示されたオブジェクトへの呼びかけを表し，ここで呼びかけられたオブジェクトは，続くメッセージの受け手となる．"!" に引き続く最初の（丸括弧などに入っていない）名前がメッセージ名であり，"!" とメッセージ名の間にあるもの（数値リテラル，文字列リテラル，丸括弧に囲まれた式など）がそのメッセージの引数となる．名前とは，英字で始まり，英字または数字の並んだものである．ただし，英字には漢字，ひらがな，カタカナが含まれる．ドリトルの文は "。" で終わる．オブジェクトに "作る" と

いうメッセージを送ると，そのオブジェクトの性質を引き継いだ新しいオブジェクトが作られ，返される．1:では，新しいオブジェクトとして，カメ太がタートルから作られる．以降，カメ太にいろいろなことを教えることができるようになる．

2: この式では，“歩く”，“右回り” がカメ太に送られたメッセージである．ドリトルのコンパイラは，16ビット文字の数字などの単純式をメッセージの引数として識別する．したがって，100，120 は続くメッセージの引数である．2: は，カメ太に対して，今向いている方向に 100 歩くようにメッセージを送り，120 右に転回させている．

3: この式では，カメ太に対して，2: に引き続き “歩く”，“右回り” のメッセージが送られ，最後に “図形にする” がカメ太に送られる．“図形にする” というメッセージで，カメ太は自分が歩いた軌跡を図形オブジェクトにして返してくれる．この結果は変数 “赤三角形” に代入される．

4: では，赤三角形に赤という色を引数とする “塗る” というメッセージを送っている．“赤” という変数には赤に対応する色が格納されているが，これを引数としてコンパイラに参照させるためには，メッセージ名として解釈されないように括弧で囲んで “(赤)” のように記す．括弧内には変数参照のほかに任意の演算式やメッセージ送信式を書くことができる．

5: 赤三角形が保持しているオブジェクトに，新しいプロパティとして “ぐるぐる” を定義する．その内容である右辺は一連の動作を表すブロックなので，以後，このオブジェクトは “ぐるぐる” というメッセージを受け取れるようになる．オブジェクト指向では，メッセージに対する手続きの本体をメソッドというが，“ぐるぐる” というメッセージに対するメソッドは，このオブジェクト自身に対して，“15 右回り” というメッセージを送るというものである．ブロックをオブジェクトのプロパティに入れると，オブジェクトが受け取るメッセージに対するメソッドとなる．“!” の左側に何も書かれていないときは，そのメソッドを保持しているオブジェクト自身がメッセージの受け手となる．このように，メッセージの受け手が明示されない “!” を導入して，self や this に相当する予約語を排除している．

6: アニメーションを制御するために，タイマーに “作る” というメッセージを送って新しい時計を作り，それを変数 “赤時計” に代入する．

7: 赤時計に対して，動作を実行する “時間” と “間隔” を規定する．

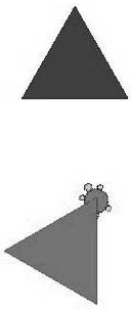

図 3 ドリトルによるプログラムの実行結果
Fig. 3 Execution results of *Dolittle* example program.

8: この “作る” では変数 “赤三角形” に格納されているオブジェクトの性質を引き継いだオブジェクトを作る．変数 “青三角形” には，その結果が代入される．青三角形は赤三角形の性質を引き継いでいるため，このオブジェクトの形も三角形であり，“ぐるぐる” というメッセージも理解する．

9: 青三角形に，“(青) 塗る” というメッセージを送って色を青く塗るように指示する．続いて x 軸方向に 30，y 軸方向に 100 移動するようにメッセージを送っている．

10: 赤時計に “作る” というメッセージを送って，赤時計の性質を引き継ぐ新しい時計を作り，それを変数 “青時計” に代入する．

11: 青時計が制御する時間間隔を 0.5 に設定する．

12:，13: 赤時計，青時計それぞれに，赤三角形，青三角形に対して “ぐるぐる” を命令する内容のブロックを “実行” の引数として渡している．各時計は，それぞれの時間だけ，規定の間隔でこのブロックを実行し続ける．

このプログラムの実行結果を**図 3** に示した．結果のアニメーションは，赤い三角形が回転し，同時に青い三角形がその倍の速さで回転し，しばらくたつと両方とも停止する．

4.2 基本オブジェクト

ドリトルではあらかじめ定義されているオブジェクトを「基本オブジェクト」と呼んでいる[6),7)]．**表 1** に基本オブジェクトを示し，以下にそれらの概要を述べる．

以下に，ドリトルの基本オブジェクトと，それらを用いたプログラムの例を紹介する．

- 数値オブジェクト
  数値を表すオブジェクトであり，四則演算や大小比較などのメソッドを持つ．数値リテラルによる表記も行える．
- 文字列オブジェクト

表 1 基本オブジェクトの一覧
Table 1 Library objects of *Dolittle*.

オブジェクト	説明
数値	数値を表すオブジェクト
文字列	文字の並びを表すオブジェクト
ブロック	動作の列を表すオブジェクト
論理値	真偽値を表すオブジェクト
配列	一連の値の並びを格納するオブジェクト
タートル	タートルグラフィックスの機能を提供するオブジェクト
図形（パス）	タートルのメソッド“図形にする”を呼び出すと，タートルがその時点で持っている軌跡から生成されるオブジェクト
色	色を表すオブジェクト
タイマー	ブロックを指定した“実行”メソッドにより，一定間隔で一定時間，操作を実行できるオブジェクト
GUI 部品	ボタン，ラベル，フィールド，選択メニューを表すオブジェクト．

```
物質 = タートル! 作る。
物質 : 重力 = 9.8。
物質 : 横速度 = 20。
物質 : 縦速度 = 50。
物質 : 質量 = 100。
物質 : 落下 =「|t| ! ((横速度)*t) ((縦速度)*t−(重力)/2*t*t)
移動する。
縦速度 =(縦速度)−(重力)*t。」。
実行間隔 =0.2。
時計 = タイマー! 作る。
時計! (実行間隔) 間隔 10 時間。
時計! 「物質! (実行間隔) 落下。」実行。
```

図 4 ブロックオブジェクトによる自由落下の公式の実装
Fig. 4 Implementation of a formula of free falls by block object.

文字の並びを表すオブジェクトであり，連結や部分文字列の取り出しなどのメソッドを持つ．文字列リテラルによる表記も行える．

- ブロックオブジェクト
  動作の列を表すオブジェクトであり，メソッド本体，制御構造の範囲や条件部などに使われる．ブロックの範囲は，「...」，[...] のどちらで指定してもよい．ブロックを用いたプログラム例を**図 4** と図 7 に示した．図 4 は，物質オブジェクトの落下公式をメソッドとして定義したブロックオブジェクトの例である．このようにブロックオブジェクトを用いると簡単にオブジェクトのメソッドを追加できる．そのため，たとえば，if 文の入れ子の内側のプログラムをブロックオブジェクトとして分離して単独のメソッドにし，入れ子を使わないプログラミングスタイルを選択することもできる．図 7 に示した GUI のプログラム例では，選択されたメニュー項目の番号を添字として，別の配列に格納されたブロックオブジェクトの 1 つを取り出して実行する．これで，メニュー項目の動作を切り替えられるようにしている．これはプログラムをブロックオブジェクトというデータとして格納している例である．また，これは，プログラムの場合分けを避ける例にもなっている．
  ブロックオブジェクトは，“「カメ太! 10 歩く 10 右回り」! 10 繰り返す。” のように，指定した回数だけ実行を繰り返すときにも使われる．もし，繰返しごとに変わる値をブロックオブジェクトに与えたいのであれば，ブロックの引数を “|” の間に書き，“「| i | カメ太! (i) 歩く 10 右回り」! 10 繰り返す。” とする．これで，実行を繰り返すときの引数の値である “10” がブロックオブジェクトの “i” に代入される．このプログラムでは，繰り返す回数によって歩数が決められる．

表 2 他の言語とドリトルとの制御構造の比較
Table 2 Comparison of the control structure of O-O languages.

Java	ドリトル

if (**条件**) {
プログラム文; ··
}
	「**条件**」! なら
	「**プログラム文.** ··」実行.

if (**条件**) {
プログラム文; ··
} else {
プログラム文; ··
}
	「**条件**」! なら
	「**プログラム文.** ··」実行
	そうでなければ
	「**プログラム文.** ··」実行.

for（**繰り返し制御文**）{
プログラム文; ··
}
	「**プログラム文.** ··」!
	n 繰り返す.

while (**条件**) {
プログラム文; ··
}
	「**条件**」! の間
	「**プログラム文.** ··」実行.

- 論理値オブジェクト
  真偽値を表すオブジェクト．リテラルはなく，“真” および “偽” という名前の広域変数にそれぞれ true，false を表す論理値オブジェクトが格納されている．制御構造の条件部分の評価結果は論理値でなければならない．論理値とブロックを組み合わせることで，一般的なオブジェクト指向プログラミング言語に見られる if，for，while と同様の制御構造を実現している．**表 2** に，一般の言語との制御構造の違いを比較するため，Java とドリトルの制御構造を提示した．
- 配列オブジェクト
  一連の値の並びを格納するオブジェクト．順に要素を追加するメソッド，位置を整数で指定して要

```
カメゾウ = タートル! 作る。
カメ子 = タートル! 作る。
カメ子! 『ayumiAka.gif』 変身する。
カメゾウ：うろうろ =「「! 30 歩く 30 右回り」! 10 繰り返す」。
カメ子：うろうろ =「「! 10 歩く 90 右回り 10 歩く 90 左回り」! 10 繰り返す」。
全員 = 配列! 作る。
全員! (カメゾウ) 入れる。
全員! (カメ子! 90 左回り) 入れる。
全員!「! うろうろ」 それぞれ実行。
```

図 5 配列オブジェクトの使用例
Fig. 5 Example of array.

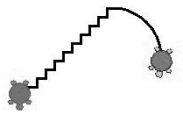

図 6 配列オブジェクトを用いたプログラムの実行結果
Fig. 6 Execution image of the array example.

素の格納をしたり，取り出しを行うメソッドを持つ．また，格納されている各オブジェクトに対してブロックで指定した動作をそれぞれ行わせるメソッド “それぞれ実行” も持つ．**図 5** に配列を用いたプログラム例を示す（実行結果は**図 6**）．2 つのタートルオブジェクトが “全員” と名付けられた配列に格納され,「! うろうろ」というブロックを指定した “それぞれ実行” メソッドにより，各々がメソッド “うろうろ” を実行する．

- タートルオブジェクト
  タートルグラフィックスの機能を提供するオブジェクトであり，“歩く”，“右回り” などのメソッドを持つ．また，メソッド “図形にする” を使うことで歩いた軌跡を図形オブジェクトにすることができる．
- 図形（パス）オブジェクト
  タートルのメソッド “図形にする” を呼び出すと，タートルがその時点で持っている軌跡が図形オブジェクトになる．図形オブジェクトは，線分の集合であり，閉じていてもいなくてもよい．図形オブジェクトが受け取ることのできるメッセージは次のとおりである．
  - 右回り，左回り
    引数で指定された角度ぶん右，または左に現在方向を変える．
  - 移動する
    現在方向に対して，引数で指定された相対座標ぶん移動する．
  - 位置
    引数で与えられた絶対座標へ移動する．
  - 塗る
    図形のパスで囲まれた領域に色を塗る．
  - 拡大する
    図形のパスの長さを拡大する．
- 色オブジェクト
  色を表すオブジェクトである．“色! 128 255 0 作る” のようにして RGB 値をそれぞれ 0〜255 の範囲で指定して色を作り出せるほか，広域変数の黒，赤，緑，青，黄，黄色，紫，水，水色，白には，それぞれその名前に対応する色オブジェクトが初期値として格納されている．
- タイマーオブジェクト
  ブロックを指定した “実行” メソッドにより，そのブロックの動作を一定時間，一定間隔で繰り返し実行させられる．その間隔や時間はあらかじめメソッド “間隔”，“時間” によって指定しておく．あるタイマーによる動作は他のタイマーの動作と並行に実行させることができる．また，指定された動作がすべて完了するのを待つメソッド “待つ” も用意されている．
- GUI 部品
  GUI 部品として，ボタン，ラベル，フィールド（テキスト入力欄），選択メニューが利用できる．これらはメソッド “位置” と “大きさ” によって画面上の位置と大きさを指定でき，またそれぞれの機能を利用するためのメソッドを持つ．ボタンが押されたときに実行したいコードを “動作” という名前のメソッドとして用意しておくことで，ボタンがクリックされたときに決まった動作を行わせることができる．**図 7** はボタン，選択メニュー，フィールドおよび配列を用いた四則演算プログラムである（実行結果は**図 8**）．

```
入力1= フィールド! 作る -250 100 位置。
入力2= フィールド! 作る -50 100 位置。
出力 = ラベル! 作る 150 100 位置 150 45 大きさ。
演算 = 選択メニュー! 作る -150 100 位置。
演算! 『+』 書く 『−』 書く 『×』 書く 『÷』 書く。
ボタン 1= ボタン! 『=』 作る 50 100 位置。
四則=配列! 作る「|x y| x+y」入れる「|x y| x−y」入れる「|x y| x*y」入れる「|x y| x/y」入れる。
ボタン1：動作 =「出力! ((四則! (演算! 何番目) 見る)! (入力1! 読む)(入力2! 読む) 実行) 書く」。
```

図 7 GUI 部品オブジェクトの使用例
Fig. 7 Example using a GUI object.

4.3 その他のドリトルのオブジェクト

教育審議会の答申[14] では，様々な教育の場でコンピュータを活用した授業を進めることを推奨している．

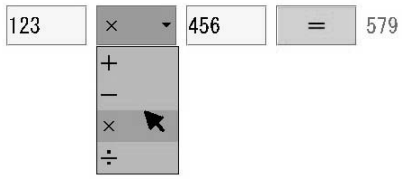

図 8 GUI オブジェクトを用いた電卓プログラムの実行結果
Fig. 8 Execution image of GUI objects example.

ドリトルを他の教科にも適用することを期待し，たとえば，ドリトルはロボットを操作したり[9),12),13)]，通信を介してオブジェクトを共有したりする機能を提供するオブジェクトも用意されている[8)]．

5. 実験授業

5.1 実験の評価基準

実験授業では，ドリトルがオブジェクトストームの目標を達成し，教育効果を上げることができたかを評価するために，授業中の生徒の発言，インタビュー，アンケートをまとめた．評価項目は，以下の点である．

(1) コンピュータの基本原理，すなわち，コンピュータはある決められた約束に従って統一的にプログラムを実行するものであるということは理解されたか．
(2) 現代的なプログラミングの方法を伝えられたか．
(3) 言語は分かりやすかったか．
(4) プログラミングの楽しみを教えられたか．

5.2 高等学校における実験授業

オブジェクトストームの目的は，プログラミングを通して生徒にコンピュータの基本原理を論理的，体感的に理解させることである．そこで，小中学生を対象とした授業を行う前に，日本語入力ができ，論理的な思考に比較的慣れていると考えられる高校 1 年生を対象に授業を行い，ドリトルによってどのようなコンピュータの基本原理を教えることができるかを実験した．

5.2.1 授業の進め方

授業は 1 人 1 台のコンピュータを用いて，1 時間ずつ 3 回に分けて行われた．各授業では，毎回 10 行程度のプログラムの解説を行い，そのプログラムを編集することから学習を始めてもらった．また，各回とも，授業の後半の 30 分程度の時間を使って課題のプログラムを作成してもらった．我々はプログラム作成中の生徒の発言や後で行ったアンケート結果，および作成されたプログラムから，生徒が獲得した知識を評価した．

授業の受講生は筑波大学附属高等学校の 1 年生 3 人である．3 人のうちの 1 人は VisualBasic のプログラミング経験者であったが，他の 2 人はワープロ程度の経験者だった．この 2 人のうちの一方は，プログラムとは何かを聞いて知っていたが，他の 1 人はまったくプログラムの存在を知らなかった．このように，プログラムに対して多少の経験の差があったり，学習の成果物にも差が生じると思われたため，個々のレベルに合った目標を設定して，プログラミングに挑戦してもらった．

```
カメ太 = タートル! 作る。
カメ太! ペンなし。
カメ太! 200 歩く。
カメ太! 120 右回り。
カメ太! ペンあり。
カメ太! 100 歩く。
カメ太! 120 右回り。
カメ太! 100 歩く。
カメ太! 閉じる。
三角 = カメ太! 図形にする。
三角! (青) 塗る。
三角! 0 200 移動する。
```

図 9 ドリトルの基本構文を学ぶためのプログラム例
Fig. 9 Introduction of basic grammar in *Dolittle*.

```
カメ太 = タートル! 作る。
カメ子 = タートル! 作る。
時計 = タイマー! 作る。
カメ太! 100 歩く 120 右回り 100 歩く。
三角形 = カメ太! 閉じる 図形にする。
三角形! 0 150 移動する。
時計! 1 間隔 10 時間。
時計! 「三角形! 36 右回り。カメ太! 15 歩く。カメ子! 36 右回り」実行。
```

図 10 タイマーを用いたプログラム例
Fig. 10 Example of Timer.

5.2.2 授業の内容

授業は次のように行われた．

1 回目：新しいオブジェクトを生成する方法，複製方法，オブジェクトとの対話の仕方を学ぶ．
図 9 に，第 1 回目の授業で解説したプログラムを示す．このプログラムはタートルグラフィックスを用いて，三角形を描くものである．このプログラムを参考にして自由な図形をドリトルで描画する課題を出し，生徒に挑戦してもらった．

2 回目：タイマーオブジェクトに繰り返し作業をさせる方法を学ぶ．
第 2 回目で用いたプログラムは，**図 10** に示すプログラムである．タイマーを用いて，2 匹のタートルと三角形を移動させるアニメーションの作り方が示されている．授業では，このプログラムを

```
カメ太 = タートル! 作る。
時計 = タイマー! 作る　1 間隔 10 時間。
三角形 = カメ太! 100 歩く 120 右回り 100 歩く 閉じる 図形にする。
三角形! (赤) 塗る。
三角形 : ぐるぐる =「時計! 「三角形! 36 右回り」実行」。
三角形! ぐるぐる。
```

図 11 メソッドの定義を含むプログラム例
Fig. 11 Example of method definition.

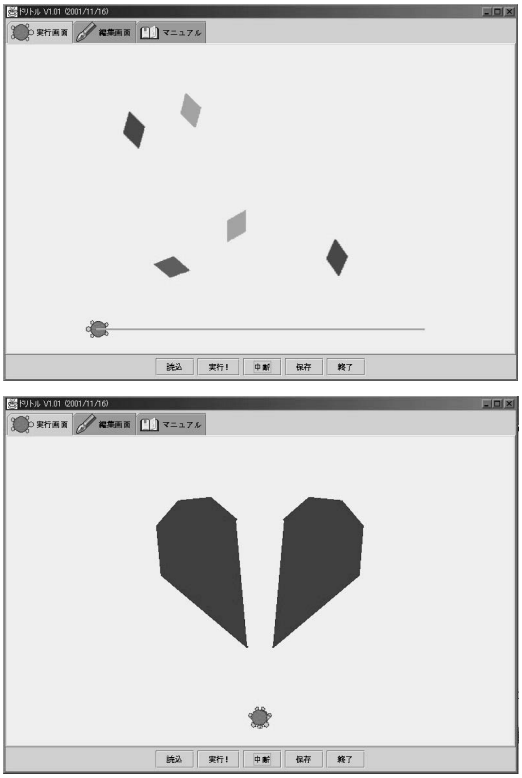

図 12 落ち葉（上）とハート（下）の実行結果：高校生の作品
Fig. 12 The source codes of the sample works “Falling Leaves” and “Broken Heart”.

参考にし，自由な図形のアニメーションを作ることを課題とした．

3 回目：オブジェクトにメソッドを教える方法を学ぶ．**図 11** に示した例を用いて，オブジェクトの動作をオブジェクトごとに定義する方法を解説した．また，複製元のオブジェクトの動作は，複製されたオブジェクトにも有効であることを解説した．**図 12**，**13**，**14** は，最終授業の自由課題で制作された生徒の作品である．図 13 のプログラムは VisualBasic の経験者によるものであり，図 14 のプログラムは，3 人による合作である．これらのプログラムは，オブジェクトの生成，継承，メソッドの定義，タイマーなどを使いこなした作品となっている．

```
カメ太 = タートル! 作る。
時計A= タイマー! 作る。
時計B= タイマー! 作る。
時計C= タイマー! 作る。
カメ太 : 落葉 =「! 30 歩く 60 右回り 30 歩く 120 右回り 30 歩く 閉じる 図形にする」。
赤1= カメ太! 落葉。
赤1! -100 150 移動する (赤) 塗る。
赤1 : 落ちる =「時計A! 0.1 間隔 7 時間 「! 18 右回り 5 − 5 移動する」実行」。
赤2= 赤 1! 作る。
赤2! -100 -20 移動する 45 右回り。
赤1! 落ちる。
赤2! 落ちる。
黄1= カメ太! 落葉。
黄1! -200 130 移動する (黄) 塗る。
黄1 : 落ちる =「時計B! 0.3 間隔 8 時間 「! 30 右回り 10 -13 移動する」実行」。
黄2= 黄1! 作る。
黄2! 100 -20 移動する 45 右回り。
黄1! 落ちる。
黄2! 落ちる。
緑1= カメ太! 落葉。
緑1! 110 110 移動する (緑) 塗る。
緑1 : 落ちる =「時計C! 0.2 間隔 6 時間 「! 40 右回り -10 -10 移動する」実行」。
緑1! 落ちる。
カメ太! ペンなし 90 左回り 200 歩く 90 右回り -250 歩く ペンあり。
カメ太! 500 歩く 図形にする (黄) 塗る。
```

図 13 落ち葉：高校生の作品 1
Fig. 13 Falling Leaves: sample work #1.

```
右 = タートル! 作る 45 左回り 50 歩く。
左 = タートル! 作る 135 左回り 50 歩く。
「右! 45 右回り 50 歩く」! 2 回 繰り返す。
「左! 45 左回り 50 歩く」! 2 回 繰り返す。
ハート1= 右! 45 右回り 75 歩く 45 右回り 171 歩く 図形にする (赤) 塗る。
ハート2= 左! 45 左回り 75 歩く 45 左回り 171 歩く 図形にする (赤) 塗る。
時計 = タイマー! 作る。
時計! 1 間隔 6 時間。
時計! 「ハート1! 0 20 移動する。ハート2! 0 20 移動する」実行。
時計! 待つ。
ハート1! (青) 塗る。
ハート2! (青) 塗る。
時計! 「ハート1! 5 右回り 20 -10 移動する。ハート2! 5 左回り -20 -10 移動する」実行。
```

図 14 ハート：高校生の作品 2
Fig. 14 Broken Heart: sample work #2.

5.2.3 授業の結果

(1) コンピュータの基本原理を理解する

ドリトルを体験した 3 人の生徒のうち，初めてプログラミングを行った生徒の発言から，以下

のことを学んだと評価できた．

- プログラムの実行のされ方に一定の規則があることを理解する
  プログラムに依存せず，プログラムは上から下へ実行されることを発見できた．
- 画面の仕組みを理解する
  1回目のプログラミング中，カメ太を5 cm程度移動させるのに100歩かせなければならないことを発見したことから，コンピュータの画面がcmやmmといった単位で取り扱われるものではなく，点の集まりとして取り扱われていることに気付くことができた．
- プログラムの役割を理解する
  プログラムで図形を描けたという体験から，画面上に現れているマウスが図形であり，プログラムによってカメが動いたように，マウスの動作にもプログラムが関与していることに気付くことができた．

(2) 現代的なプログラミングを体験できた
既存のオブジェクトに「作る」というメッセージを送ると元のオブジェクトの複製ができる体験をとおして，複製されたオブジェクトが元のオブジェクトの性質を引き継いでいることを学んだ．また，新しいオブジェクトを追加したり，オブジェクトにメソッドを追加したりすることによって，それ以降，使える語彙が増えていくことを体験した．

(3) 言語は理解しやすかった
ドリトルは簡潔であり，実行結果が視覚化されているため，数行の例題プログラムからプログラムと結果の動作の対応関係を理解した．特にアニメーションは，図形の作成，時間の操作方法の定義，動作のアルゴリズムの構築という複合した作業の成果として完成できるプログラムである．図13や図14のようなアニメーションプログラムを，3回の授業と30分のプログラミングだけで完成できたということは，ドリトルを理解するのに多くの時間が必要ないことを表していると評価できる．

(4) プログラミングを楽しめた
授業終了後，進化したドリトルを再び使ってみたいという要望が寄せられた．ドリトルを介して，プログラミングに興味を持ってもらえたようである．プログラミングの楽しみを教えることができたか否かの詳細な評価は，次項の中学校における授業の結果から評価を行うことにする．

5.3 中学校における実験授業

国内の複数の中学校の授業でドリトルの試験適用を行っている．

5.3.1 授業の進め方

三重県松阪市鎌田中学校では，技術・家庭科の情報基礎領域の必修授業をプログラミング未経験の2年生4クラス125人の生徒に対して行っている．2001年度は2年生の1学期から情報教育を始めているが，2学期の1時限（50分）の授業6回でドリトルを導入した．生徒には，毎回学習ノートを書いてもらい，そこに主な学習内容と授業の感想を10〜20文字程度で，さらに，難しさ，達成度，楽しさを1から4までの4段階で，評価4をとても簡単/達成できた/とても楽しい評価として記入してもらった．本節では，この授業の評価結果のうち楽しさに関する評価を行い，他の評価は別の機会に行うことにする．

5.3.2 授業の内容

2学期に行われた6回の授業の流れは以下のとおりである．

(1) 導入：プログラミングの概念を学ぶため，タートルを動かしてみる．
(2) タートルグラフィックス：タートルを動かして三角形，四角形，円，星を描く．
(3) 図形オブジェクトを作る：前回の復習からタートルが描いた軌跡を図形オブジェクトにし，色を塗る．
(4) 図形オブジェクトを操作する：図形オブジェクトに名前を付けて移動する．
(5) 自由課題：画面に作品を作る．
(6) アニメーションを作る：タイマーを使ったプログラムを作成する．

5.3.3 授業の結果

授業の楽しさに関するアンケート結果を**表3**と**図15**に示す．グラフの横軸は授業の回数を示し，縦軸は楽しさに関する4段階の各評価結果の生徒の割合を表す．評価によると，最終回の授業の評価で1，2と答

表3 楽しさに関する授業アンケート結果（単位：人）
Table 3 Result of questionnaire on delight.

回数	1	2	3	4	5	6
楽しくなかった	10	9	11	4	5	4
少し楽しくなかった	26	34	21	23	22	18
少し楽しかった	60	46	56	49	46	44
楽しかった	25	27	29	37	45	47

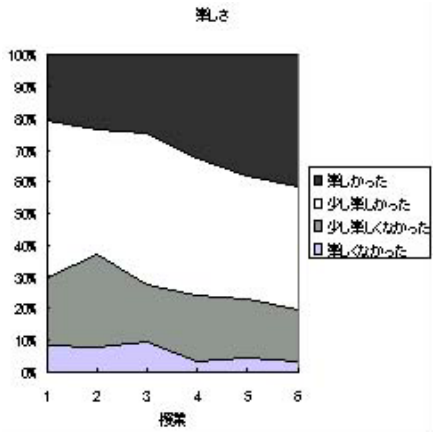

図 15 鎌田中学校におけるドリトルの楽しさの評価結果
Fig. 15 Evaluation results of delight on *Dolittle* at Kamata Junior High School.

えた生徒は 113 人☆中 22 人であったが，そのうち，過去 5 回の授業で少なくとも 1 回は評価 4 をつけて楽しみを味わうことができた生徒は，10 人であり，まったく楽しさを感じることができなかった生徒はわずか 3 人であった．

一方，最終回の授業で楽しさの評価が 3 以上であった生徒は 113 人中 91 人で，全体の 8 割を超えた．これらの生徒のうち，初回の評価が 2 以下であった生徒は 18 人で，初回に 4 の評価をつけた生徒は 19 人だった．各自の評価の変遷を見ると，回を重ねるごとに授業の内容が難しくなっていたにもかかわらず，徐々に楽しみを味わえるようになった傾向が見てとれる．以下に，実際の生徒が書いてくれた感想文を転載する．

- 生徒 A
  - 1 回目：楽しさの評価 3
    使いやすかったし，楽しかった．
  - 2 回目：楽しさの評価 2
    四角形はできたけど，三角形ができなかった．
  - 3 回目：楽しさの評価 3
    全部できた．色をぬるのが楽しかった．
  - 4 回目：楽しさの評価 4
    図形を動かすのが楽しかった．
  - 5 回目：楽しさの評価 4
    いろいろな図形を描くのは楽しかったけど難しかった．
  - 6 回目：楽しさの評価 4
    動かすのが楽しかった．

☆ 授業を受講した生徒である 125 人よりも少ないのは，最終回を欠席した生徒がいたからである．

- 生徒 B
  - 1 回目：楽しさの評価 3
    あまりむずかしくはなかった．
  - 2 回目：楽しさの評価 2
    三角形が少しむずかしかった．
  - 3 回目：楽しさの評価 2
    円がむずかしかった．
  - 4 回目：楽しさの評価 3
    図形を動かすことができなかった．
  - 5 回目：楽しさの評価 3
    文字でうつのは少しめんどうだったけどできた．
  - 6 回目：楽しさの評価 2
    図形を動かせなかった．

個別には，生徒 B のように，図形を動かすのが難しくて楽しさを失ってしまった生徒もいたが少数であった．感想にあるように，生徒 B は文字をうつというプログラミング活動に対しては達成感という楽しみを味わっていたといってよいだろう．

我々は，5 回目の授業を参観したが，生徒の 8 割が画面に図形を表示することができていた．これは，ドリトルが理解しやすい言語である根拠となろう．

集団教育では，すべての学習者を完全に理解できるまで指導するのは困難であるから，鎌田中学校の授業で，最終的に 8 割の生徒が「楽しい」と感じ，多くの作品を作成できたことは，授業の進め方とともに，ドリトルにも生徒に楽しさを与える魅力を持っていたといってよいだろう．さらに，ドリトルでプログラミングの楽しみを味わってもらえるようにするには，ドリトルの利用者インタフェースの評価や基本オブジェクトの追加および機能の見直しも必要となる．今後も引き続き評価と言語の見直しを継続していく予定である．

6. 評　　価

6.1 他言語との比較

初中等校におけるプログラミング教育では，LOGO や一般のオブジェクト指向言語の適用が考えられる．プログラミング言語は汎用的であれば実現できる機能は多くなるが，それだけ学ぶことも多くなり，最初のプログラムを動かすまでの壁は高くなる．これに対して，ある目的に特化した言語の場合，学ぶ項目は少なくて済むが，多くの機能を実現することは困難になる．たとえば Java は前者の代表であり，LOGO は後者の例といえるだろう．ドリトルでは，これら 2 つの目標をバランス良く達成する言語の開発を目指した．ここではドリトルと比較するプログラミング言語として，LOGO

表 4 プログラム比較：ロゴ坊，Java，ドリトル
Table 4 Program comparison between LOGO, Java and *Dolittle*.

	LOGO	ドリトル	Java
行数	22	6	40
変数の数	1	3	14
オブジェクトの数	—	3	12
概念	手続き 逐次処理 繰り返し 引数	タートル 図形 タイマー ブロック メソッド	継承 配列 スレッド 例外 Runnable interface public クラス 無名内部クラス ....

```
手順は ぐるぐる
    繰り返せ 10 [
        三角 7
        待て 1
        三角 0
        右へ 20
    ]
終わり
手順は 三角 :c
    色は :c
    繰り返せ 3 [
            –右欄へ続く–
```

```
            –左欄より続き–
        前へ 100
        左へ 120
    ]
    左へ 10
    ペンをあげろ
    前へ 10
    塗れ
    後ろへ 10
    ペンをおろせ
    右へ 10
終わり
```

図 16 ロゴ坊による三角形回転アニメーションプログラム
Fig. 16 Rotating triangles animation in LOGO-bou.

の日本語版であるロゴ坊[4)]☆と一般のオブジェクト指向言語の中から Java を選択し，図 11 に示したドリトルのプログラムと同様の機能をそれぞれの言語で実現し，教育言語としての学びやすさを比較評価した．プログラムの行数，変数の数，オブジェクトの数，使われている概念について比較したものを**表 4** に示す．

(1) LOGO との比較
図 11 に示したドリトルのアニメーションをロゴ坊で記述したプログラムを**図 16** に示す．LOGO ではタートルが三角形を何度も描き直しながらアニメーションを実現しなければならない．
市販の LOGO には，複数のタートルを定義して絵を描かせることができるものもあるが，基本的にすべての画面に表示される絵はタートルグラフィックスであるから，色を塗るにもタートルを図形の中に移動させる必要がある．したがって，LOGO では，たとえば図 10 に示したように複数の図形を動かすアニメーションを作るのは困難である．
表 4 に示したように，使われている概念の数はドリトルより LOGO の方がわずかに少ないが，プログラムの簡潔さ，明快さから見ればドリトルが優る．たとえば，三角形の数を増やすとドリトルでは，増えた三角形に対応してプログラム内の三角形オブジェクトの数も増えるが，LOGO では三角形の数を増やしても，プログラム上の三角形の数と対応することはない．LOGO の三角形とは図形を指しているのではなく，三角形を描くための手続きを表しているからである．このようにドリトルで，実際に目で確認できる「もの」とプログラムの対応関係がとれているのは，オブジェクト指向の採用により，抽象度の高いレベルでの記述を可能にしたことによっている．

(2) 一般のオブジェクト指向言語との比較
言語比較用に作成した Java プログラムを**図 17** に示す．このプログラムの三角形はアニメーション機能を持っている．表 4 に示すように，Java プログラムではきわめて多くの概念が現れており，行数も多く，変数の数も多い．これは，Java が汎用のプログラミング言語として大規模なソフトウェア作成に対応できるように設計されていることによる．このような言語を初中等教育に用いることはかなり困難をともなうものと考えられる．

(3) 教師による評価
実際にドリトルによる教育を中学校で試行した静岡県藤枝市立西益津中学校の紅林秀治教諭による評価[5)]を**図 18** に示す．これによると，LOGO や BASIC と比べて，ドリトルでは生徒たちが手続き中心ではなくオブジェクト中心のより抽象度の高いプログラミングスタイルが自然に取り入れられていることがうかがえる．

6.2 オブジェクト指向の効果

ドリトルはオブジェクト指向言語である．これまで多くのオブジェクト指向言語が開発され，使われてきたが，なぜ教育現場への適用が検討されてこなかったのであろうか．それには，以下のことが原因であると考えられる．

- オブジェクト指向は実務システム開発のために提案された考え方である．実システムへの適用が優先され，教育への適用の検討が遅れた．
- 現役システム開発技術者にとって，オブジェクト

☆ http://www.logob.com/logo/

```
//—- 三角形クラスを用いたプログラム —-
import java.awt.*;
public class Rot extends Frame {
 Triangle t1 = new Triangle(180, 200, 60, Color.red);
 public Rot(){ setSize(400, 400); }
 public void start() {
  (new Thread( new Runnable() {
   public void run() {
    while(true) {
     try {Thread.sleep(50);}
     catch(Exception ex) { }
     t1.addTheta(0.05);
     repaint();
    }
   }
  })).start();
 }
 public void paint(Graphics g){t1.draw(g);}
 public static void main(String[ ] arg){
  Rot app = new Rot();
  app.setVisible(true);
  app.start();
 }
 static class Triangle {
  double gx, gy, rad, theta = 0.0, d = Math.PI*2/3;
  Color col;
  public Triangle(double x, double y, double r, Color
c){
   gx = x; gy = y; rad = r; col = c;
  }
  public void addTheta(double dt){theta + = dt;}
  public void draw(Graphics g){
   int[ ] x = {(int)(gx+rad*Math.cos(t heta)),
               (int)(gx+rad*Math.cos(theta+d)),
               (int)(gx+rad*Math.cos(theta−d))};
   int[ ] y = {(int)(gx+rad*Math.sin(theta)),
               (int)(gx+rad*Math.sin(theta+d)),
               (int)(gx+rad*Math.sin(theta−d))};
   g.setColor(col); g.fillPolygon(x, y, 3);
  }
 }
}
```

図 17 Java による三角形回転アニメーションプログラム
Fig. 17 Triangles rotating animation in Java.

BASIC やロゴでのプログラミング学習との違いは，生徒の思考パターンのように思いました．BASIC やロゴでは，フローチャート図を想定したプログラム作りが生徒の思考の中心でしたが，ドリトルの場合は生徒が作ったオブジェクトどうしのかかわり合いを中心にプログラム作りをしています．それは，タイマープログラムにより複数のオブジェクトを同時に一画面上に動かすため，それぞれ作ったオブジェクトが「衝突したらどうするか」とか,「ボタンオブジェクトとどうかかわらせていくか」など，プログラムの流れが一筋ではなく，それぞれ異なるという観点から作品作りに取り組んでいるように思えました．これが，これからのコンピュータプログラミングに必要な基礎的な考え方とすれば，ドリトルの学習は今後の情報教育学習に大いに役立つのではないかと考えています．

（紅林秀治氏．文献 5) から引用）

図 18 教師による評価
Fig. 18 Evaluation at the education in practice.

指向は難しく，ハードルが高いと当初いわれた．そのため，教育への適用は無理であるという先入観があった．

- 教育現場では多くの種類のマシンが使われる．マシン非依存のオブジェクト指向言語として Smalltalk[2] があったが非常に高価であった．そのため，Java の開発がなければ教育現場へオブジェクト指向を導入することは困難であった．
- 教師が旧来の BASIC や LOGO，FORTRAN で教育を受けており，それ以降のプログラミング言語に関する情報を得ていない．

コンピュータの低価格化，インターネットの普及，Java など，オブジェクト指向言語を教育現場へ適用する環境はやっと整ってきたといえるだろう．教師の教育の問題は，ドリトルを適用した授業の成果報告などを随時進めながら情報提供を行い，解決していきたい．

7. ま と め

本稿では，オブジェクトストームという情報教育の新しいコンセプトを紹介し，ドリトルの教育言語としての有効性について，実験授業の成果を示しながら議論した．現在,「ドリトル」はインターネットを介して配布されている☆．今後は，利用者を増やし多くの意見を採り入れて言語開発を進めるとともに，他の教科への適用事例の開発，さらに，学校への導入活動を進めていく予定である．

謝辞 本研究は，情報処理振興事業協会（IPA）の平成 12 年度未踏ソフトウェア創造事業の援助を受けました．また，電気通信大学の竹内郁雄氏から有益なコメントを多くいただきました．実験授業に協力いただいた筑波大学付属高等学校の矢野一幸教諭，三重県松阪市立鎌田中学校の井戸坂幸男教諭，静岡県藤枝市立西益津中学校の紅林秀治教諭，そして生徒の皆様に感謝いたします．

参 考 文 献

1) Cypher, A. (Ed.): *Watch What I Do: Programming by Demonstration*, MIT Press (1993).
2) Goldberg, A. and Robson, D.: *Smalltalk-80 The Language and Its Implementation*, Addison-Wesley (1983).
3) Gosling, J., Joy, B. and Steel, G.: *The Java Language Specification*, Second Edition, Addison-Wesley (2000).
4) 井戸坂幸男：コンピュータの楽しさを知るプロ

☆ http://www.logob.com/dolittle/

グラミングソフト「ロゴ坊」，NEW 教育とコンピュータ，1997 年 3 月号，pp.42–45, 学習研究社 (1997).
5) 井戸坂幸男，紅林秀治：プログラミングっておもしろい—中学校　技術・家庭科での授業実践（ドリトルではじめる情報教育（第 3 回）），NEW 教育とコンピュータ，2002 年 3 月号，pp.84–87, 学習研究社 (2002).
6) 兼宗　進，久野　靖：学校教育用オブジェクト指向言語 “Dolittle” の提案，第 42 回プログラミングシンポジウム，pp.11–20 (2001).
7) 兼宗　進，御手洗理英，中谷多哉子，福井眞吾，久野　靖：学校教育用オブジェクト指向言語「ドリトル」の設計と実装，情報処理学会論文誌：プログラミング，Vol.42, No.SIG11(PRO12), pp.78–90 (2001).
8) 兼宗　進，中谷多哉子，御手洗理英，福井眞吾，久野　靖：オブジェクト指向言語「ドリトル」を利用した情報教育について，情報教育シンポジウム論文集，B2-2f, 情報処理学会コンピュータと教育研究会 (2001/8).
9) 兼宗　進：「コンピュータっておもしろい！」を体験しよう（ドリトルではじめる情報教育（第 1 回）），NEW 教育とコンピュータ，2002 年 1 月号，pp.88–90, 学習研究社 (2002).
10) 片桐　明：日本語プログラミング言語 Mind，翔泳社 (1988).
11) 木村俊一：日本語プログラム，昭和 62 年度第一回人工知能学会全国大会論文集，pp.465–468 (1987).
12) 紅林秀治：プログラミングっておもしろい—ドリトルで制御の学習その 1（ドリトルではじめる情報教育（第 4 回）），NEW 教育とコンピュータ，2002 年 4 月号，pp.120–123, 学習研究社 (2002).
13) 紅林秀治：プログラミングっておもしろい—ドリトルで制御の学習その 2（ドリトルではじめる情報教育（第 5 回）），NEW 教育とコンピュータ，2002 年 5 月号，学習研究社（掲載予定）.
14) 文部科学省教育課程審議会：幼稚園，小学校，中学校，高等学校，盲学校，聾学校及び養護学校の教育課程の基準の改善について（答申），前文. http://www.mext.go.jp/b_menu/shingi/12/kyouiku/toushin/980703.htm#1 (1998/7/29).
15) Papert, S.: *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books (1980). 奥村貴世子（訳）：マインドストーム，未来社 (1982).
16) Stroustrup, B.: *The C++ Programming Language*, Third Edition, Addison-Wesley (1997).
17) Ungar, D. and Smith, R.: Self: The Power of Simplicity, *Proc. OOPSLA'87*, pp.227–242 (1987).

(平成 13 年 10 月 1 日受付)
(平成 14 年 3 月 14 日採録)

中谷多哉子（正会員）

1957 年生．1980 年東京理科大学理学部応用物理学科卒業．1994 年筑波大学大学院経営・政策科学研究科経営システム科学専攻修士課程修了．1998 年東京大学大学院総合文化研究科広域システム科学系博士課程修了．博士（学術）．富士ゼロックス情報システム（株）を経て，（有）エス・ラグーンを設立．和歌山大学客員教授．オブジェクト指向分析・設計方法論，要求工学に興味を持つ．共編著に「ソフトウェアパターン」（共立出版）等．電子情報通信学会，日本ソフトウェア科学会，ACM，IEEE CS 各会員．

兼宗　　進（正会員）

1963 年生．1988 年千葉大学工学部電子工学科卒業．1990 年筑波大学大学院理工学研究科理工学専攻修士課程修了．同年（株）リコー入社，現在に至る．2000 年から筑波大学大学院経営・政策科学研究科企業科学専攻博士課程在学中．ACM，IEEE 各会員．プログラミング言語，データベースシステム等に興味を持つ．

御手洗理英（正会員）

1960 年生．1978 年目黒星美学園高等学校卒業.（株）フジミックを経て，（株）アーマットを設立．ソフトウェア開発に従事し，現在に至る．1985 年放送大学教養学部作業と技術専攻卒業，2001 年筑波大学大学院経営・政策科学研究科経営システム科学専攻修士課程修了．放送大学非常勤講師．著書に「学習 BDS C・α-C」（工学図書，共著），「入門 Visual Basic6.0」（きんのくわがた社）等がある．日本ディスタンスラーニング学会，日本教育工学会，教育システム情報学会各会員．

福井　眞吾（正会員）

1959 年生．1984 年東京工業大学大学院理工学研究科情報科学専攻修士課程修了．同年日本電気（株）入社．現在，同戦略マーケティング本部エキスパート．1991 年〜1992 年イリノイ大学客員研究員．現在，筑波大学大学院経営・政策科学研究科博士課程在学中．プログラミング言語，分散システム，データベースシステム等に興味を持つ．ソフトウェア科学会，ACM 各会員．

久野　　靖（正会員）

1956 年生．1984 年東京工業大学大学院理工学研究科情報科学専攻博士後期課程単位取得退学．同年東京工業大学理学部情報科学科助手．筑波大学経営システム科学専攻講師，同助教授を経て現在同教授．プログラミング言語，プログラミング環境，ユーザインタフェース，情報教育に興味を持つ．著書に「UNIX による計算機科学入門」（丸善），「入門 WWW」（アスキー）等がある．日本ソフトウェア科学界，ACM，IEEE Computer Society 各会員．