初中等教育におけるオブジェクト指向プログラミングの実践と評価

著者	兼宗 進, 中谷 多哉子, 御手洗 理英, 福井 眞吾, 久野 靖
雑誌名	情報処理学会論文誌. プログラミング
巻	44
号	SIG_13(PRO_18)
ページ	58-71
発行年	2003-10
権利	情報処理学会
その他のタイトル	K12 Education and Evaluation Using an Object-oriented Programming Language
URL	http://hdl.handle.net/2241/119193

初中等教育におけるオブジェクト指向プログラミングの実践と評価

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

近年，教育課程の改訂により，小学校から高等学校までの初中等教育において情報教育の導入が進められている．筆者らは,「情報教育の中でプログラミングを体験することは計算機の動作原理を学ぶうえで効果的である」という考えから，初中等教育で活用可能なプログラミング言語である「ドリトル」を開発し，それを用いたプログラミング教育について研究を行ってきた．本稿では，中学校においてドリトルを用いて実施したプログラミング教育とその評価について報告する．行った授業では，プログラミング経験のない生徒を対象に，タートルグラフィックスと GUI 部品を用いたプログラミングを扱った．また，アンケートと 2 回の定期試験の結果に基づき，オブジェクト指向を含むプログラミングの概念が生徒にどのように理解されるかを分析した．

K12 Education and Evaluation Using an Object-oriented Programming Language

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

Recently, IT education curriculum at K12 (kindergarten and 12-year-education) schools are being started in Japan. Programming experiences will be useful for students to learn essence and principle of computers. However, we insist that the use of “modern” languages is indispensable to achieve the goal. In this paper we describe our experiences of using “Dolittle” object-oriented programming language in junior high school classes. Turtle graphics, figure objects, and GUI objects were taught in the class, and paper tests and enquiries were conducted for evaluation. As the result, majority of the students understood basic concepts of programming and object-orientation, and enjoyed the classes.

1. はじめに

近年，教育課程の改訂により，小学校から高等学校までの初中等教育において情報教育の導入が進められている．2002 年度から，小中学校において情報教育の内容を含む新教育課程が実施された[20),21)]．また，2003 年度からは，高等学校において普通教科「情報」が実施される予定である[19)]．これらに象徴されるように，情報技術に関する適切な理解を持たせることは，我が国の初中等教育における重要な課題の 1 つとなっている．

プログラミングを体験することは，計算機の働きについて学ぶための効果的な手段の 1 つである[12)]．しかし，これまで教育用プログラミング言語として広く使われてきた BASIC[10)] や LOGO[12)] は 20 年以上前に設計された言語であり，これらを用いたプログラミングを体験しても，ふだん生徒が使っている現代的なソフトウェアの理解には結び付きにくいという問題があった．

このため筆者らは，今日のソフトウェア技術において主要な位置づけを占めるオブジェクト指向[11)] の考え方に基づいた教育用オブジェクト指向言語「ドリトル」を設計・開発し，これを用いた情報教育の可能性について研究を行っている．

本稿では，ドリトルを中学校技術・家庭科の情報基礎領域において活用した授業の事例を報告し，アンケートと定期試験による評価を通して学校教育におけるプログラミング言語の利用を考察する．

† 筑波大学大学院ビジネス科学研究科
Graduate School of Systems Management, University of Tsukuba
†† 有限会社エス・ラグーン
S-Lagoon Co., Ltd.
††† 株式会社アーマット
Armat Corporation

2. 情報教育とプログラミング

初中等教育で開始された情報教育では，計算機の操作だけでなく，情報の持つ意味や計算機の動作原理についての理解が扱われる．計算機の原理として，ハードウェアの物理的な特性や CPU の仕組みを理解することは重要であるが，ソフトウェアがオペレーティングシステムや日常使用するアプリケーションソフトウェアという形で計算機の中で重要な役割を果たしていることの理解も重要である．

ソフトウェアを理解するために，ソフトウェアの存在を知識として説明することは可能であるが，実感として理解するためには，プログラムを作ってみることが効果的である．初心者がプログラムを作るためのアプローチとしては，画面上の図形で行う方法[8] や例示による方法[1] などが知られている．しかし，次のような理由から，我々はテキストによりプログラムを記述するアプローチを採用した．

- コードを解釈して動作する計算機の原理との一致．
- 用途をシミュレーションなどに限定しない汎用的な記述が可能．
- 制御構造やオブジェクトなどの基礎概念を直接扱える．
- 他のプログラミング言語へ発展させやすい．

本稿で扱った実験では，中学生がプログラミングを通して次の概念を段階的に学習することを目的としている（本文中では次の (1)，(2) を合わせてプログラミングの基礎概念と呼ぶ）．

(1) 手続き的プログラミングの基礎概念
手続き的なプログラミングの基礎概念として，逐次実行，代入による値の変更，制御構造による実行制御などを学ぶ．

(2) オブジェクト指向プログラミングの基礎概念
オブジェクトに頼む実行モデル，値と手続きを内包する自律性などを学ぶ．

(3) 計算機の動作原理
簡単なアプリケーションプログラムを作る体験を通して，アプリケーションソフトウェア，オペレーティングシステム，CPU の内部動作など，計算機の中でソフトウェアの果たす役割を学ぶ．

今回行った授業では，時間的な制約から，(1) から (3) の前半である「簡単なアプリケーションプログラムの作成」までを扱った．次の段階の授業では，今回のプログラミング体験を基に,「計算機で重要な役割を担う各種ソフトウェアの理解」に発展させていくことが可能と考えている．

これらの目的を達成するために，次の方針で授業を設計した．

(1) 中学生が理解し，記述できる言語設計
簡潔な構文を採用することで，構文の理解でのつまずきを少なくする．変数名や命令語，記号に日本語で使われている記法を採用することで，英語の得意・不得意によらず学習できる言語とする．

(2) 画面上のオブジェクトを操作する学習モデル
図形やボタンなど，目に見えるオブジェクトに命令を送り操作するプログラミングを基本とする．このことにより，オブジェクトの存在を実感することができ，自分の書いたプログラムの動作を確認しながら学習を進めることができる．

(3) 生徒の興味を継続させる題材
描いた図形が動き出すアニメーションや，ボタンを押すことによる対話的な操作など，生徒の興味を継続する題材を扱う．また，適度な難易度を設定することで，課題をクリアする達成感を継続しつつ，難しすぎないカリキュラムとする．

3. プログラミング言語「ドリトル」

ドリトル[16)~18)] は教育用に設計されたオブジェクト指向言語である☆．ドリトルは簡潔な構文を採用しており，オブジェクトに呼び掛ける形でプログラムを記述する．また，プロトタイプ方式のオブジェクト指向[2),14)] を採用し，オブジェクトは複製により生成する．このためクラス定義が不要であり，プログラムを簡潔に記述できる．

3.1 言語仕様

図 1 にドリトルの構文を，**図 2** にドリトルのプログラム例を示す．以下ではこの例を使い，ドリトルの構文を解説する．

3.1.1 メッセージ送信

```
カメ太 = タートル ！ 作る。
```

タートルオブジェクトの複製を作り，“カメ太” という変数に格納している．画面にはカメの姿をしたタートルオブジェクトが現れる．タートルオブジェクトはタートルグラフィックスを実現するオブジェクトであり，軌跡の線を描きながら画面を移動する．

3.1.2 カスケード送信

```
カメ太 ！ 100 歩く 120 左回り 100 歩く
```

☆ 最新版はプログラミング言語「ドリトル」のサイト（http://www.logob.com/dolittle/）から入手できる．

```
プログラム ::= (文 ‘。’)…
文 ::= 代入文
  | メソッド定義
  | 式
代入文 ::= 変数 ‘=’ 式
メソッド定義 ::= 変数 ‘=’ ブロック
変数 ::= [項 ‘:’]  名前
式 ::= 単純式 | メッセージ送信
メッセージ送信 ::= [レシーバ] ‘!’  メッセージ
レシーバ ::= 項
メッセージ ::=
  引数… メソッド名 ([‘;’] 引数… メソッド名)…
引数 ::= 単純式
メソッド名 ::= 名前
括弧 ::= ‘(’ 中置式 ‘)’ | ‘(’ メッセージ送信 ‘)’
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
演算子 ::= ‘+’ | ‘-’ | ‘*’ | ‘/’ | ‘>’ | ‘<’
  | ‘>=’ |‘<=’ | ‘==’ | ‘!=’
```

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

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

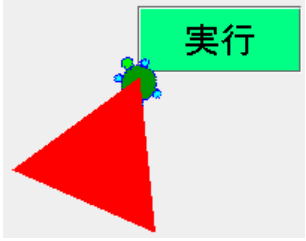

図 2 サンプルプログラムと実行例
Fig. 2 An example program in Dolittle.

閉じる。

ドリトルでは，“!” でオブジェクトに呼び掛ける形でプログラムを記述する．この部分は，『カメ太! 100 歩歩いて，120 度左に回って，100 歩歩いて，元の場所に戻って』と読み下すことができる．実行すると画面に正三角形が描かれる．

3.1.3 図形オブジェクト

三角形 = カメ太 ! 図形にする（赤）塗る。

カメ太に “図形にする” を送り図形オブジェクトを作る．タートルが描く線はタートルオブジェクトの一部であり☆，明示的に “図形にする” を実行することで，線をタートルから切り放して独立した図形オブジェクトを生成できる．続いて返された図形オブジェクトに “塗る” を送り色を塗る．最後に，図形オブジェクトを “三角形” に代入し名前を付けている．

3.1.4 タイマー

時計 = タイマー ! 作る 1秒 間隔 10秒 時間。

タイマーは決められた時間の間，一定間隔でプログラムを実行するオブジェクトである．ここでは “時計” という名前のタイマーオブジェクトを作り，実行時間と実行間隔をセットしている．

3.1.5 メソッド定義

三角形:ぐるぐる=「|x| 時計!「!（x）右回り」実行」。

オブジェクトには，プロパティにブロックを代入する形でメソッドを定義できる．ここでは三角形に，“x” という引数を 1 個とる “ぐるぐる” という名前のメソッドを定義している．このメソッドが実行されると，時計は与えられたブロック “「!（x）右回り」” を 1 秒間隔で 10 秒間（すなわち 10 回）繰り返し実行する．

3.1.6 イベント処理

実行ボタン = ボタン ! "実行" 作る。
実行ボタン:動作=「三角形! 36 ぐるぐる」。

“実行ボタン” という名前のボタンオブジェクトを作り，押されたときの動作を定義している．

ドリトルでは，ボタンオブジェクトの “動作” やタートルオブジェクトと図形オブジェクトの “衝突” など，特定の名前のメソッドを定義することで，そのオブジェクトのイベントを受け取ることができる．衝突メソッドは，タートルや図形オブジェクトが他のオブジェクトと重なったときに実行される．

このプログラムでは，画面に表示された “実行” ボタンを押すことにより，三角形に定義された “ぐるぐる” というメソッドが実行される．その結果，三角形が画面上で 1 秒ごとに 36 度ずつ回転するアニメーションが表示される．

3.2 実装と動作環境

ドリトルの処理系は Java 2[6] により記述されており，Java 2 が動くさまざまな環境で動作する．字句解析，構文解析は SableCC[3] による生成コードを利用し，これに加えて意味解析，インタプリタ，ユーザインタフェース，および標準オブジェクト群を実装して

☆ カメの尻尾が長く伸びた姿を想像すると理解しやすい．

図 3 ドリトルの実行画面
Fig. 3 Execution of Dolittle.

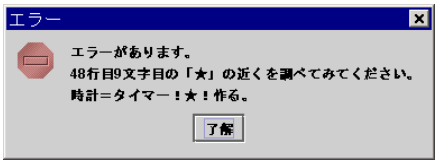

図 4 エラーダイアログ
Fig. 4 Error dialog.

いる．

今回実施した授業では，中学校のパソコン教室の端末（Windows98）に Java 2 の実行環境とドリトルをインストールすることで授業環境を構築した．

3.3 ユーザインタフェース

図 3 にドリトルの実行画面を示す．実行画面と編集画面，マニュアルを画面上部で切り替えて操作する．画面下には実行などのボタンが並んでいる．

プログラムは編集画面で記述する．実行時のエラーはダイアログでエラーの位置を表示するようにした．**図 4** にダイアログの例を示す．

4. 授業での活用

設計した言語が実際の学校教育で活用できることを検証する目的から，実験授業を行った．この授業は中学校において行われ，複数のクラスで実施されたものである．講義は学校教員がプログラミング経験のない生徒に対して行った．

4.1 授業準備

以下は，授業を実施するために我々が行った準備である．

(1) 参考資料の提供
授業で使用するテキストを作成してもらうために，ドリトルのマニュアル類と以前我々が実施した実験授業[16] のテキストを提供した．

(2) 講習会の実施
ドリトルとオブジェクト指向の概念を説明するために，2 時間の講習会を実施した．テキストはドリトルのチュートリアルマニュアルを使用し，スライドを使いながら説明を行った．

(3) 電子メールによるサポート
教員との連絡は電子メールを用いて行った．ドリトルを開発しているメンバと，学校での授業を準備している教員でメーリングリストを作り，テキストを作成する際のサンプルプログラムの相談や，質問への回答などを行った．

(4) 定期試験問題とアンケートの作成
定期試験の問題とアンケートを教員と相談しながら作成した．

担当教員は次の作業を行った．

(1) 毎時間の授業計画の策定
(2) 生徒に配布するテキストの作成
(3) 授業の実施
(4) 定期試験とアンケートの実施

4.2 実験対象

授業は三重県松阪市立鎌田中学校で実施した．教科は技術・家庭科の情報基礎領域の一環として行った．期間は 2001 年度の 2 学期後半から 3 学期に行われ，合計 11 時間の授業となった．この授業は必修科目であり，第 2 学年の生徒全員（132 人）が受講した．この授業以前にプログラミングの授業を行っていないため，ほとんどすべての生徒にとって，初めてのプログラミング体験となった．

4.3 行われた授業

授業は担当教員が黒板でその日のポイントを解説してから実習形式で行われた．中学校における 1 時間は 50 分だが，計算機の起動に 5 分，授業ごとのアンケートと回収に 5 分が使われるため，毎回の実質的な授業時間は 40 分前後であった．

テキストとして，毎時間の最初にその日の資料を配布した．テキストには，説明用の例題，新しい構文の説明，実習用の例題が書かれている．付録に，授業で使用したテキストの例を示す．

表 1 に，実施した授業のカリキュラムを示す．内容に書かれた毎時間のタイトルは，その時間の授業で使用した資料のタイトルである．

1，2 時間目は手続き的なプログラミングの基礎概念を理解する授業である．生徒はタートルグラフィックスを使い画面に三角形や四角形などの図形を描くことでプログラミングを体験した．入門段階でタートルグラフィックスを用いる利点は，プログラムが正しく動作しているかを本人が視覚的に判断できるため，正

表 1 授業カリキュラム
Table 1 Curriculum of experimental lesson.

学期	時間	内容
2		《タートルグラフィックス》
	1	使ってみよう
	2	三角形，四角形
		《図形オブジェクトの生成と操作》
	3	星・円・塗る
	4	図形を移動する
	5	いろいろな図形を描く
3		《タイマーによるアニメーション》
	6	タイマー
	7	作品にタイマーを入れる
	8	作品作り
		《GUI 部品（ボタン）の活用》
	9	ボタン
	10	ボタン
	11	作品作り

しい動作に至るまで，自分で実行結果をフィードバックしてデバッグ作業を行えることである．

3〜5 時間目はオブジェクト指向の基礎概念を理解する授業である．生徒はタートルオブジェクトに“図形にする”を送り，タートルが移動により描いた線を図形オブジェクトにすることで，オブジェクトの生成を学んだ．次に，画面上の図形オブジェクトに色を塗るメッセージを送ることで,「特定のオブジェクトにメッセージを送る」というモデルを学んだ．

5 時間目までの内容で，2 学期の期末テストを行った．

6〜8 時間目はタイマーを理解する授業である．生徒は画面上の図形オブジェクトを一定の速度で移動させるアニメーションを通して，タイマーオブジェクトを体験した．

9〜11 時間目は GUI 部品を体験する授業である．今回は代表的な GUI 部品としてボタンを扱った．生徒はボタンが押されたときの動作をボタンオブジェクトに定義することを通して，メソッドの利用とイベントを処理するプログラミングを体験した．11 時間目の最後には，プログラミング概念の理解度を調査するアンケートを行った．

11 時間目までの内容で，3 学期の期末テストを行った．

5. 授業内容の評価

言語の評価を行う目的から，担当教員と相談しながらアンケートとテスト問題を作成し，それらの結果と生徒作品を分析することで次の評価を行った．

- 授業ごとのアンケート
- 定期試験
- 理解度アンケート

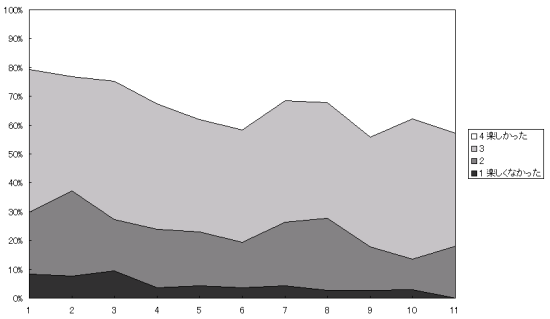

図 5 アンケート結果（楽しさ）
Fig. 5 Questionnaire (pleasant).

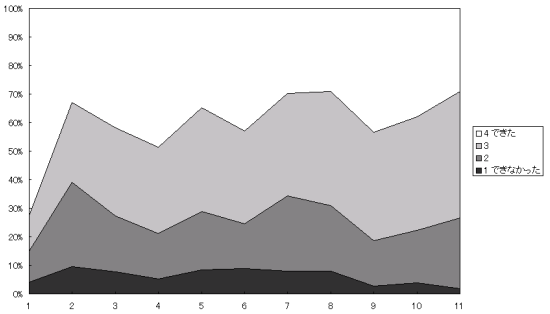

図 6 アンケート結果（達成度）
Fig. 6 Questionnaire (achievement).

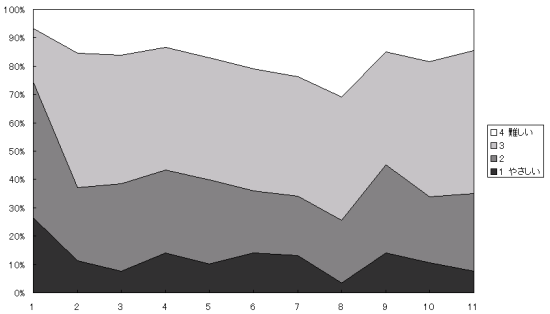

図 7 アンケート結果（難しさ）
Fig. 7 Questionnaire (difficulty).

- 生徒作品の分析

5.1 授業ごとのアンケート

授業の中でアンケートを実施した．アンケートは毎時間の最後に行い，数行の感想を記述するほか,「楽しさ」,「課題の達成度」,「難しさ」を 4 段階で記入させた．**図 5**，**6**，**7** に集計結果を示す．

アンケートから読み取れる結果を示す．

- 無理なく授業が成立した
  達成度を見ると,「その時間の課題を達成できなかった（達成度が 1）」と回答した生徒の割合はつねに 10%未満と低く，最後まで授業が成立して

いたことが分かる．

- 授業が進むにつれて楽しさが増した
  楽しさを見ると,「楽しかった（楽しさが 3 か 4)」と回答した生徒の割合は，授業が進むにつれて 20%から 40%に増加した．これは，授業を前半と後半に分けて行った符合検定の結果からも有意な変化である．一方,「楽しくなかった（楽しさが 1)」と回答した生徒の割合はつねに 10%未満で推移し，最後は 0%になった．
- 達成したときの喜びが大きかった
  難しさと楽しさを見ると，授業が進むにつれて難しいと感じる割合が増えているが，楽しさを感じる生徒の割合も増えている．最後の授業について達成度と楽しさの相関を求めたところ 0.55 であることから，課題の達成度が楽しさに結び付いていることが分かる．授業を担当した井戸坂教諭は「難しさを感じて，それを乗り越えた喜びを感じている生徒が多かった」[15] とコメントしている．中学生にとって，適度な難易度と達成感のある授業を構築することができた．
- 難易度の高い時間があった
  難しさを見ると，タイマーを扱った 6〜8 時間目の難易度が高くなっている．6 時間目でタイマーの意味を学んだ後，7，8 時間目の応用に進むにつれて難しさを感じる生徒が増えていることから，タイマーを自分の作品に応用することが難しかったことが分かる．

5.2 定期試験による評価

授業の内容を 2 回の定期試験に出題し，評価した．

試験は技術・家庭科の技術科の部分（25 分間，50 点満点）として行われた．家庭科と合わせて実施したため，テキストやノートは持ち込まない形とした．試験の結果は，担当教諭により成績評価のための採点が行われた.「計算機による構文チェックを活用し，デバッグを行いながら開発を進めていく」というプログラミングの特性を考慮し，採点は細かい記述ミスに対しては部分点を与える形で行った．

今回の研究では，プログラミング概念の理解度をより細かく分析する目的から，問題ごとにチェックポイントを設定し，回答の内容を検討することで，出題時に意図した概念を生徒が理解しているかどうかを判定した（概念理解率)．

概念理解率では，採点上は不正解とされた場合でも，次の条件にあてはまる場合は「正解ではないが，概念を理解している」と判定している．

- プログラミングの概念は理解しているが，図形の数学的な知識など，他の概念の誤りで誤答となった場合
- プログラミングの概念は理解しているが，記号の抜けや書き間違いなど細かなミスにより構文が不完全であった場合

表 2 2 学期末テストの問題と概念理解率
Table 2 The rate of understanding (2nd term-end examination).

問題		概念理解率
1	逐次実行の理解	89.6%
2	プログラムの記述と繰返し	84.0%
3	構文エラーの発見	92.0%
4	プログラムの読解	91.2%
5	絶対座標の理解	73.6%
	平均	86.1%

5.2.1 テスト 1（2 学期末試験）

5 時間目までの内容を 2 学期の期末試験として出題し，評価を行った．出題のねらいと概念理解率を**表 2** に，テスト問題を**図 8** に示す．

採点の結果は，平均点が 32.3 点であり，満点（50 点）が 4 人であった．概念理解率の平均が 86.1%であることから，構文誤りまたは図形の数学的な知識など他の概念の誤りによる減点が約 10 点あったことが分かる．

生徒の理解度については，基本問題である問題 1〜4 の概念理解率が 84%以上あることから，80%以上の生徒がプログラミングの基本概念を理解できていることが確認された．

以下に，各問題を考察する．

問 1 逐次実行の理解

画面上のタートルオブジェクトにメッセージを送るプログラムを示し，画面に描かれる線を図示する問題である．

結果は，89.6%の生徒が「タートルオブジェクトに前進と回転が交互に 3 回ずつ送られる」という図を描くことができた．約 90%の生徒が逐次的な実行の流れを理解していることが分かる．

問 2 プログラム記述と繰返し

タートルグラフィックスの実行結果を示し，その図形を描くためのプログラムを記述する問題である．図形は階段の形をしており，1 段ずつの描画を繰り返して描くことができる．問題では「できるだけ長くないプログラムをかきなさい.」という形で繰返しの存在を気付かせるようにした．

結果は，84.0%の生徒が階段状の図形を描くプログラムを記述した．また，半数の 42.0%の生徒が繰返しを含むプログラムを記述した．約 85%の生

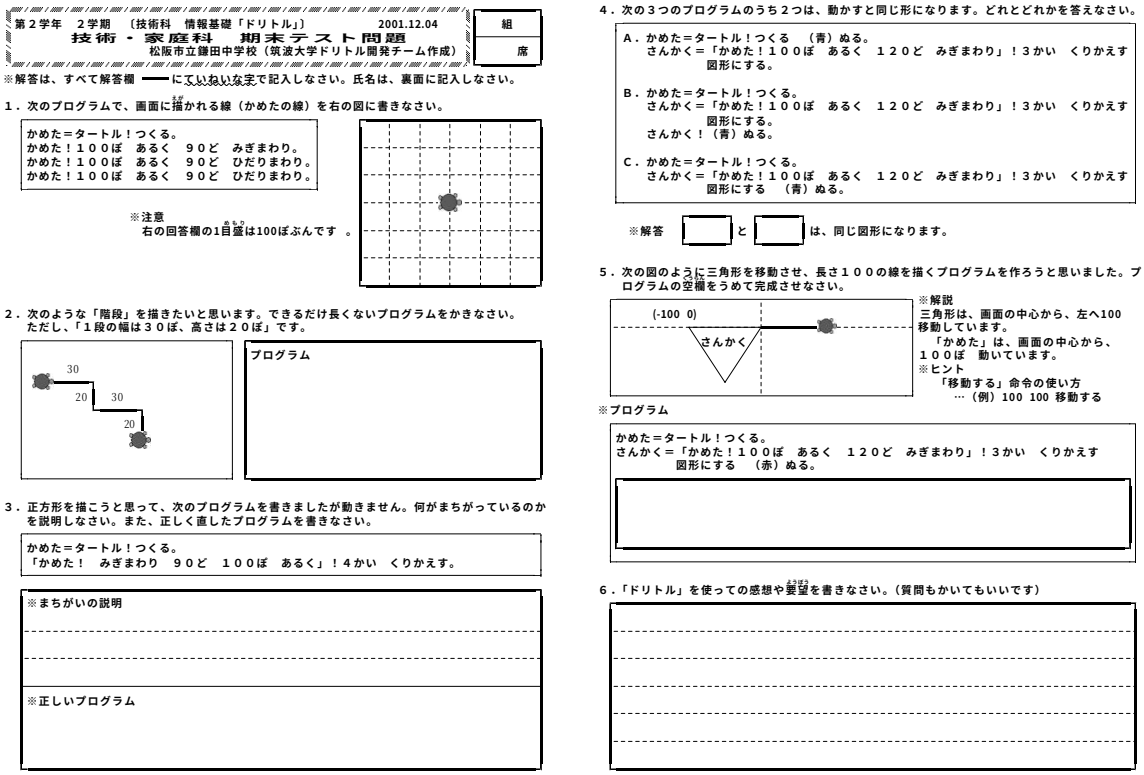

図 8 ２学期末のテスト問題
Fig. 8 Exercise (2nd term-end examination).

徒が基本的なプログラムを記述することができ，そのうち半数の生徒が繰返しを使いこなしていることが分かる．

問 3 構文エラーの発見

構文規則に適合しない記述を含むプログラムを示し，誤りの説明と修正後のプログラムを記述する問題である．誤った構文には，命令語と引数の順序が逆になった記述を使用した．

結果は，92.0%の生徒が “みぎまわり 90ど” が誤りであることを指摘し，“90ど みぎまわり” と正しく修正した☆．90%以上の生徒が基本的な構文の規則を把握していることが分かる．

問 4 プログラムの読解

3つのプログラムを比較し，結果が同じ動作になる2つのプログラムを選択する問題である．選択肢としたプログラム例のうち，プログラム A はタートルオブジェクトに図形用のメッセージ “ぬる” を送っているため，正しく動作しない．プログラム B とプログラム C はともにタートルグラフィックスの線分から作った図形オブジェクトにメッセージを送る．プログラム B は図形オブジェクトの生成と図形へのメッセージ送信を別の文に分けているが，プログラム C では 1 つの文の中で図形オブジェクトの生成とメッセージ送信を行っている．結果は，92.2%の生徒が正解である B と C を選択した．90%以上の生徒がオブジェクトごとに解釈できるメッセージが異なることを理解していることが分かる．

問 5 絶対座標の理解

実行結果の画面を示し，その結果を得るためのプログラムを完成させる問題である．この問題では画面に図形とタートルを生成するプログラムが与えられており，その後にタートルと図形をそれぞれ動かすプログラムを追加する．

結果は，73.6%の生徒がタートルと図形を移動するプログラムを記述した．

この問題では，「図形とタートルの移動」という基本的な内容を扱っているにもかかわらず，他の問題と比較して正答率が低い結果となった．これ

☆ ドリトルの日本語命令は，“右回り” のような漢字表記を標準としている．今回の実験授業を行うにあたり，担当教員から「“塗る” のように自分が読めない難しい漢字が出現すると，生徒によっては学習意欲をなくしてしまう可能性がある」という指摘があり，“みぎまわり” のような仮名のみの命令表記を追加した．

は，他の問題がタートルグラフィックスによる相対座標を扱っているのに対し，この問題では絶対座標を用いているという違いによるものと考えられる．このことから,「絶対座標を意識せず，現在の位置から相対的な移動を繰り返す」ことを基本としたタートルグラフィックスは，入門用の言語に適していることが分かる．

5.2.2 テスト 2（3 学期末試験）

11 時間目までの内容を 3 学期の期末テストとして出題し，評価を行った．出題のねらいと概念理解率を**表 3** に，テスト問題を**図 9** に示す．

採点の結果は，平均点が 27.7 点であり，満点（50 点）が 2 人であった．概念理解率の平均が 74.3%であることから，構文誤りまたは図形の数学的な知識など他の概念の誤りによる減点が約 10 点あったことが分かる．

問題 1，2，4，5 の概念理解率から，タイマーを除くプログラム上の概念は 70%以上の生徒が理解していることが確認された．

以下に，各問題を考察する．

表 3 3 学期末テストの問題と概念理解率
Table 3 The rate of understanding (3rd term-end examination).

問題	内容	概念理解率
1	オブジェクトの区別	90.2%
2	ボタンによる対話操作	92.7%
3	タイマーによる繰返し	42.3%
4	仕様を満すプログラミング作成	74.8%
5	メソッドの定義	71.5%
	平均	74.3%

問 1 オブジェクトの区別

オブジェクトの性質を表す 5 種類の説明文を示し，あてはまるオブジェクトを 5 種類の選択肢から選択する問題である．

結果は，90.2%の生徒が正しいオブジェクトを選択した．90%以上の生徒がオブジェクトごとの違いと基本的な性質を理解していることが分かる．

この問題では，複数個の選択を許したことから，厳密には性質を満さないオブジェクトを回答するケースがあった．たとえば（5）の設問では，“あるく”というメソッドを持つタートルオブジェクトが正解となる．しかし，図形オブジェクトも “移動する”といった類似のメソッドにより移動することが可

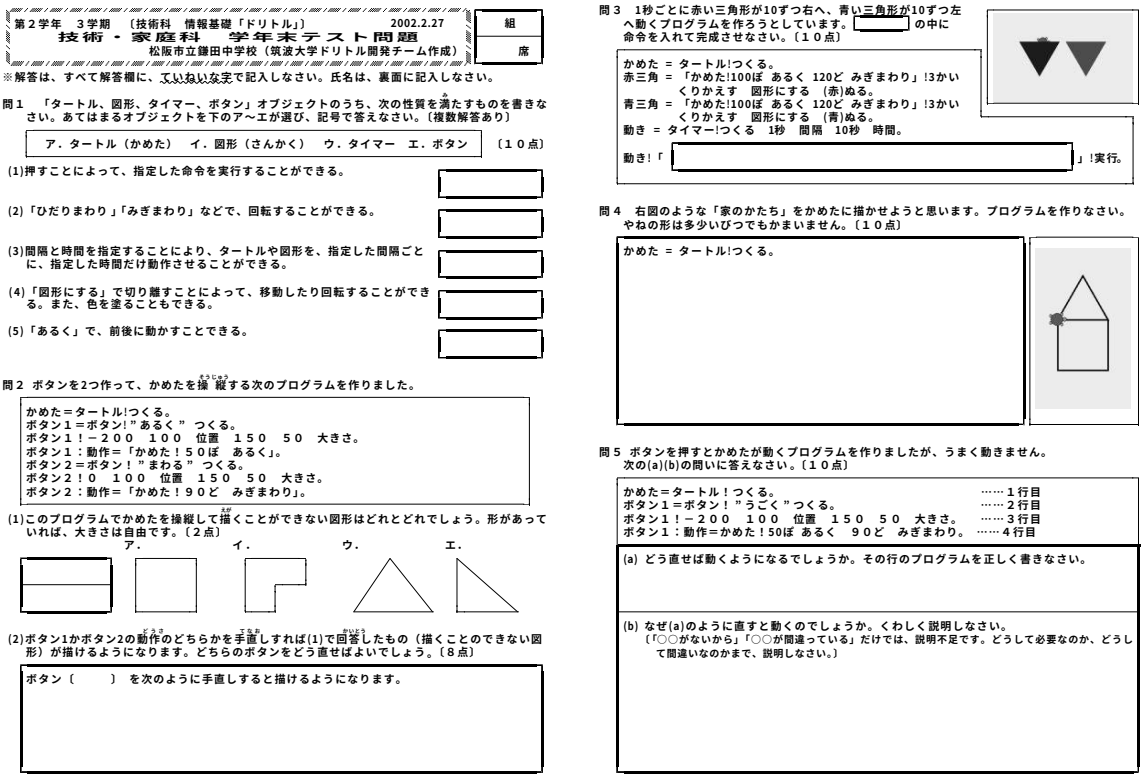

図 9 3 学期末のテスト問題
Fig. 9 Exercise (3rd term-end examination).

能であることから，今回の評価では，図形オブジェクトを含めて回答した場合でも「基本概念を理解している」と判断した．

問 2 ボタンによる対話操作

前進・回転という 2 つのボタンを操作して図形を描くプログラムを示し，描くことのできる図形を選択する問題と，そのままでは描けない図形を描くためにプログラムを修正する問題である．提示したプログラム例では，回転する角度は 90 度になっている．そのため，90 度の倍数以外の角度を持つ図形は描けない．

結果は，92.7%の生徒が描ける図形として 90 度の倍数の角度を持つ頂点から構成される図形を選び，三角形などを描くために回転するボタンのメソッドを 15 度など汎用性の高い角度に修正することができた．90%以上の生徒が，ボタンの機能を修正することでプログラムの汎用性を増やせることを理解していることが分かる．

問 3 タイマーによる繰返し

画面に描かれた 2 つの図形をタイマーにより移動するプログラムを完成させる問題である．

結果は，42.3%の生徒がタイマーにより図形を動かす記述を行った．他の問題と比較して正答率が悪かった理由としては，12 月にタイマーを扱う授業を行った後，1，2 月にボタンを扱う授業を行ったため，試験前の 2 カ月間に授業の中でタイマーを使う機会がなかったことが考えられる．

問 4 仕様を満すプログラミング作成

課題として与えられた図形を描くプログラムを作成する問題である．図形は三角形と四角形を組み合わせた家の形である．

結果は，74.8%の生徒が課題の図形を描くプログラムを作成できた．

問 5 メソッドの定義

メソッド定義の誤った構文を示し，修正後のプログラムと誤りの理由を記述する問題である．ドリトルのメソッドはブロックをプロパティに代入する形で定義するが，この問題ではブロックを示す記号（「…」）を記述していないため，メソッドが定義されない．

結果は，71.5%の生徒が次のように正しくプログラムを修正した．

```
ボタン1:動作=「かめた!50ぽ あるく
90ど みぎまわり」.
```

なぜ「…」を付けると動作するのかという説明は，授業の中でブロックについて説明しなかったこともあり，ほとんどの生徒は「括弧を付けないとボタンの動作を定義できないから」という構文上の規則として回答した．2 人だけ「括弧を付けないとその場で実行されてしまう」のように,「プログラムを内包するオブジェクト」であるブロックの存在に気付いていると思える回答を行った．

5.3 理解度アンケート

最終回の授業では，プログラミング概念の理解度を見るためのアンケートを実施した．このアンケートでは，プログラミングに関する 8 個の質問を行い，生徒はそれぞれについて理解している実感を回答した．質問には，授業で明示的に説明を行っていない概念を含め設問を設定した．結果を**表 4** に示す．表中で「不明」は「設問の意味が分からなかった」という選択肢である．

このアンケートの結果から，次のことが分かる．

- 授業の中で説明を行った設問（○）は，肯定的な回答（1 または 2）が 60%以上あり，プログラミングの基礎概念を理解できたと感じている生徒が多い．
- 授業で説明していない設問（×）のうち,(5),(6) については約 60%の生徒が「プログラムを正しく書けば自分の考えたとおりに動く．エラーがあれば動かない」と経験的に認識している．
- (7) については，授業で説明していないにもかかわらず,「タイマーが非同期に実行される」ことに気付いている生徒が約 50%いる．
- (8) については授業で説明していないこともあり，設問を理解できない生徒が 33%存在した．ブロックの存在は授業の中で説明しない限り理解することは難しいことが分かる．

5.4 生徒作品の分析

9〜11 時間目の授業では，学習内容をまとめる課題として，ボタン操作で線や図形を描くペイントソフトを作成した．

図 10 に，生徒作品の例を示す．このプログラムでは，画面左にボタンが配置されており，それぞれのボタンに次の種類のメソッドが定義されている．

- タートル操作ボタン．タートルを操作する（移動，回転，ペンの上げ下げ）．
- 図形描画ボタン．繰返し命令でタートルを移動し，図形オブジェクトを生成する．
- 色指定ボタン．ペンの色を変更する．
- 姿の指定．タートルの画像を変更する．

学んだ知識を活用することで，ほぼ全員が豊富な表現力を持つペイントソフトを作成することができた．

表 4 理解度アンケートの結果
Table 4 Questionnaire of understanding.

設問		授業での説明	肯定 ← 1	2	3	4	→ 否定 5	不明 X
(1)	プログラムの書き方はきっちり決まった規則があって，その規則を少しでもはずれていると動作しない．	○	60.0%	21.7%	9.6%	5.2%	1.7%	1.7%
(2)	「あるく」「移動する」「塗る」などの命令の前には，「どれくらい動くか」「どこに移動するか」「何色に塗るか」などの指定をつけることができる．	○	62.6%	14.8%	10.4%	2.6%	4.3%	5.2%
(3)	「タートル」「図形」「タイマー」「ボタン」など，ものの種類ごとに見え方やできることが違っている．	○	47.0%	23.5%	12.2%	3.5%	2.6%	11.3%
(4)	さまざまなものに命令するときは，そのものを指定した後に「!」をつけて，それから命令を書く．	○	38.9%	25.7%	18.6%	4.4%	7.1%	5.3%
(5)	プログラムは原則として書かれた順番どおり上から下，左から右の順番にきっちり実行されていく．	×	33.9%	26.1%	17.4%	6.1%	1.7%	14.8%
(6)	コンピュータは自分が書いたプログラムの通りに厳密に動作するようにできている．	×	27.0%	30.4%	26.1%	7.8%	3.5%	5.2%
(7)	タイマーによる動作だけは（5）と異なり，タイマーに指示したタイミングで実行される．	×	27.2%	21.1%	21.1%	7.0%	0.9%	22.8%
(8)	繰り返し実行させる動作，タイマーに実行させる動作，ボタンが押されたときの動作などは「…」という形で囲んだもの（ブロック）で指定する．	×	18.3%	17.4%	20.9%	4.3%	6.1%	33.0%

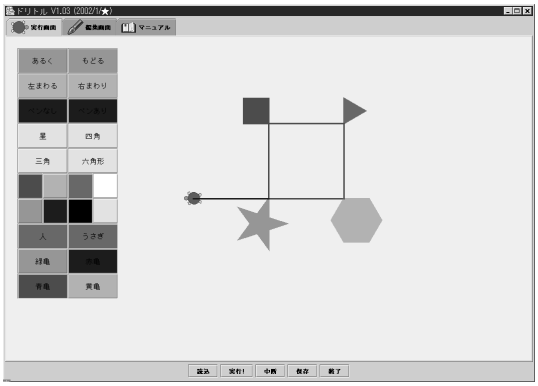

図 10 生徒作品（ペイントソフト）
Fig. 10 Student work (paint tool).

表 5 生徒作品の分析
Table 5 Analysis of student works.

機能	使用率
タートル	100%
図形オブジェクト	97%
繰返し	96%
ボタン	93%
タイマー	85%

表 5 に，「ペイント」を含む各種の生徒作品の中で使われた構文やオブジェクトを示す．ボタンやタイマーを含むオブジェクトを，85%以上の生徒が活用していることを確認できた．

6. 関連研究

オブジェクト指向言語を小中学生に使わせる試みとして，Smalltalk-80[4] を用いた報告がある．Goldberg ら[5] は，子供が Smalltalk を学ぶためのサンプルプログラムや教材例を紹介している．Kay ら[9] は，子供たちに Smalltalk を教えた経験と，代表的な生徒作品を紹介している．これらは研究室において個別指導に近い形で観察を行ったものであり，学校教育において Smalltalk を扱ったときに，必ずしもすべての生徒が扱えるものではないと考える．

Smalltalk の実装である Squeak[7] の上に構築された子供用のプログラミング環境として，SqueakToys[13] がある．画面上のパレットと呼ばれる領域に，値や制御構造を表すタイルを置く形でプログラムを記述する．SqueakToys では，タイルを用いることでテキストの入力を不要としているが，逆にテキストによる記述を行いたいときには Smalltalk の構文を用いた Squeak のプログラミングが必要になる．

子供用に開発されたビジュアルプログラミング言語として，ToonTalk[8] がある．論理型言語の節を表すロボット，プロセスの実行を表すトラック，プロセス間のメッセージを表す鳥などの画面上のキャラクタを操作することによりプログラムを定義する．ToonTalk は並行論理を基礎とするプログラミング環境であり，オブジェクト指向に基づいた命令型を基礎とするものではない．

画面上でシミュレーションを容易に行えるプログラミング環境として，Cocoa[1] がある．Cocoa は例示プログラミング（Programming by Demonstration）を採用しており，行いたい動作の例を示すことでプロ

グラミングを行う．隣接する領域のキャラクタの有無など，ある状態にある画面上のキャラクタを選択し，その位置や外観を変えることにより，キャラクタがその状態になったときの動作ルールを定義する．実行すると，画面上の状態にルールを適用し，次々に画面上の状態を修正する．状態の修正の様子を画面上で見ることにより，シミュレーションを観察することができる．Cocoa は画面上のシミュレーションを例示により簡潔に記述することを目的としており，汎用的なプログラミングを行うものではない．

小中学生が扱うことを意識したプログラミング環境には図形によるプログラミングを行うものが多い．本稿では，2 章で述べた理由から情報教育にはテキストによるプグラミング記述が望ましいと考え，実験授業を行った．

7. 考　　察

本研究では，プログラミング経験のない生徒を対象に，中学校の授業の中でドリトルを用いた情報教育を実施し，定期試験やアンケートに基づく調査研究を行った．

調査結果の分析を通して得られた知見は次のものがある．

- 中学校において，オブジェクト指向言語を用いた情報教育が可能であることを確認した．定期試験の分析（5.2 節）と生徒作品の分析（5.4 節）から，11 時間という短い時間の中で，繰返しやメソッド呼び出しなどのプログラミングの基本概念のほか，ボタンへの動作メソッドの定義やタイマーを用いたスレッドなどを扱えることを確認した．
- 学年全体（4 クラス 132 人）を対象とした集合教育の中で，プログラミングの授業が成立した．定期試験の分析（5.2 節）から，約 9 割の生徒がプログラミングの基本概念を理解していることを確認した．毎時間のアンケートの分析（5.1 節）から，授業が進むにつれて楽しさが増し，達成感につながっていることを確認した．
- プログラミングを学ぶうえで，オブジェクト指向の概念が特に問題になることはなかった．概念理解度のアンケート分析（5.3 節）では，オブジェクトの存在（設問 3）とメッセージ送信（設問 4）について，65%以上の生徒が肯定的な回答（「1」または「2」）をしていることから，図形をオブジェクトとして位置付け，色を塗ったり位置を移動したりできるという形でのオブジェクト指向の利用は，生徒にとって抵抗なく受け入れられた．また，毎時間のアンケートの分析（5.1 節，図 7）では，オブジェクトの生成を扱った授業（たとえば複製によりタートルオブジェクトを生成した 1 時間目，描いた軌跡から図形オブジェクトを生成した 3 時間目など）に難易度が増す傾向は見られないことから，クラス定義を用いずにオブジェクトを複製によって生成するプロトタイプ方式の採用は，オブジェクト指向言語に対する初期の敷居を下げることができたと考える．

授業実践を通して得られた知見は次のものがある．

- GUI 部品（特にボタン）を学び，「ボタンを押すと何らかの動作が起きる」形のプログラムを作成することに多くの生徒が強い興味を示した．授業を担当した井戸坂教諭は，「GUI 部品を利用したペイントソフト等の作品により，生徒たちは実社会で利用されているソフトウェアと，自分たちが学んだプログラミング学習を結び付けることができた」[17] と観察している．
- タイマーを用いたアニメーションを，授業の中で学習できた．ただし，毎時間のアンケートの分析（5.1 節）から，タイマーは 6〜8 時間目に扱う内容としては難易度が高いことが分かった．9 時間目以降に実施したボタンオブジェクトをタイマーよりやさしいと感じる生徒が多いことから，今後はボタンオブジェクトで対話的な操作を学んだ後でタイマーによる自動的な繰返しを学んだり，タイマーの学習に費やす時間を増やすなど，カリキュラムの改善を行いたい．

今後の課題には次のものがある．

- 今回の授業では，制御構造として繰返しを扱った．今後は条件分岐など，今回のカリキュラムで扱わなかった制御構造についても検証したい．
- 今回の授業では，ボタンの動作という特定の用途においてメソッドの定義を扱った．今後はオブジェクトに汎用的なメソッドを定義して拡張する形でのプログラミングを検証したい．
- 今後小学校でも利用される可能性を考えると，エラーメッセージの提示方法などユーザインタフェースの改良も行っていきたい．

8. ま と め

教育用に設計したオブジェクト指向言語「ドリトル」を，中学校の授業で活用し，評価を行った．その結果，ドリトル言語を用いた情報教育では，従来の教育用言語（BASIC，LOGO）を用いた場合よりもスムーズに生徒の学習を進めることができ，またさほど多くな

い時間数にもかかわらず，生徒のコンピュータの原理に対する理解を高めるという目標が達成できたと考えている．

今後は，今回の実験授業で得られた成果を発展させ，全国の学校教育で活用可能なカリキュラムの作成と処理系の改良を進める予定である．

謝辞 本研究の実験授業にご協力いただいた，松阪市立鎌田中学校の井戸坂幸男先生と生徒の皆様に感謝いたします．

参 考 文 献

1) Cypher, A. and Smith, D.C.: KidSim: End user programming of simulations, *Proc. ACM Conference on Human factors in computing systems* (*CHI'95*), pp.27–34 (1995).
2) Association, E.C.M.: ECMAScript Language Specification. http://www.ecma.ch/ecma1/stand/ecma-262.htm
3) Gagnon, E.: SableCC, An Object-Oriented Compiler Framework, Master's thesis, McGill University (1998). http://www.sablecc.org/
4) Goldberg, A. and Robson, D.: *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley (1983).
5) Goldberg, A. and Ross, J.: Is the Smalltalk-80 System for Children?, *BYTE*, Vol.6, No.8, pp.348–368 (1981).
6) Gosling, J., Joy, B. and Steele, G.: *The Java Language Specification*, Addison-Wesley (1996).
7) Ingalls, D., Kaehler, T., Maloney, J., Wallace, S. and Kay, A.: Back to the future: the story of Squeak, a practical Smalltalk written in itself, *Proc. 1997 ACM SIGPLAN Conference on Object-oriented programming systems, languages and applications*, pp.318–326 (1997).
8) Kahn, K.: Drawings on napkins, video-game animation and other ways to program computers, *Comm. ACM*, Vol.39, No.8, pp.49–59 (1996).
9) Kay, A. and Goldberg, A.: Personal Dynamic Media, *IEEE Computer*, Vol.10, No.3, pp.31–41 (1977).
10) Kemeny, J.G. and Kurtz, T.E.: *Basic programming*, John Wiley (1967).
11) Meyer, B.: *Object-oriented Software Construction*, Prentice Hall (1994).
12) Papert, S.: *Mindstorms : children, computers, and powerful ideas*, Basic Books (1980).
13) Steinmetz, J.: Computers and Squeak as Environments for Learning, *Squeak: Open Personal Computing and Multimedia*, Prentice Hall (2001).
14) Ungar, D. and Smith, R.B.: Self: The Power of Simplicity, *OOPSLA'87*, pp.227–242 (1987).
15) 井戸坂幸男，紅林秀治：「ドリトル」ではじめる情報教育第3回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，NEW 教育とコンピュータ，Vol.18, No.3, pp.84–87 (2002).
16) 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，Vol.42, No.SIG11(PRO12), pp.78–90 (2001).
17) 兼宗 進，中谷多哉子，井戸坂幸男，御手洗理英，福井眞吾，久野 靖：教育用オブジェクト指向言語「ドリトル」による授業実施とその評価，情報処理学会情報教育シンポジウム（SSS2002），pp.229–236 (2002).
18) 中谷多哉子，兼宗 進，御手洗理英，福井眞吾，久野 靖：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，情報処理学会論文誌，Vol.43, No.6, pp.1610–1624 (2002).
19) 文部省（編）：高等学校学習指導要領 (1999).
20) 文部省（編）：小学校学習指導要領 (1999).
21) 文部省（編）：中学校学習指導要領 (1999).

付録 使用したテキスト例

図 11 に，授業で使用したテキストの例を示す．5時間目の「いろいろな図形を描く」で使用された資料である．内容としては一定回数の繰返し実行を扱っており，次の内容が説明されている．

- その日に学ぶ内容の説明
  四角形の描画を題材に，繰返しを使わない4行のプログラムと，繰返しを使った1行のプログラムを対比して説明している．
- 新しい構文の説明
  繰返しの構文を説明している．
- ワークシート
  自分でプログラミングを試すための例題がある．

(平成 14 年 12 月 24 日受付)
(平成 15 年 7 月 1 日採録)

４．使ってみよう(3)

年	組	席	名前

４．同じ命令を繰り返す命令（繰り返し命令）

```
かめた！１００ぽ…あるく…９０ど…みぎまわり。
かめた！１００ぽ…あるく…９０ど…みぎまわり。
かめた！１００ぽ…あるく…９０ど…みぎまわり。
かめた！１００ぽ…あるく…９０ど…みぎまわり。
```

↓ ※繰り返す命令を使った場合

```
「かめた！１００ぽ…あるく…９０ど…みぎまわり」！４かい…くりかえす。
```

命令	「○○」！△かい…くりかえす 「○○」！△回…繰り返す
解説	「　」の中の命令を△回くりかえして実行します。

練習３　繰り返す命令を使って、いろいろな図形をえがいてみよう。（１００ぽの大きさでつくろう。命令は、空欄にメモしておこう。）

①「正方形」

②「三角形」

③「星」

④「円」

⑤５角形、６角形、８角形、１０角形、１２角形、２４角形……

図 11　使用したテキスト例
Fig. 11　Text sample.

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ビジネス科学研究科企業科学専攻博士課程在学中．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．ACM，IEEE 各会員．

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．

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．ソフト開発に従事し，現在に至る．1985 年放送大学教養学部産業と技術専攻卒業，2001 年筑波大学経営・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，筑波大学大学院ビジネス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会，ACM，IEEE Computer Society 各会員．