タブレット端末を活用したプログラミング教育（４）
―ビジュアルプログラミングからOOPへ―

Programming Education Using the Tablet Device (IV)
- The Proper Perspective from Visual Programming to OOP -

田近一郎，本多一彦，杉江晶子[1)]，森　　博
Ichiro TAJIKA, Kazuhiko HONDA, Akiko SUGIE, Hiroshi MORI

本タイトル第４報ではタブレット端末を利用したビジュアルプログラミング教育の急速な発展を紹介するとともにその特徴について議論する．以前の報告でタブレット端末を利用した先進の教育を紹介してきが，この分野の発展は著しくプログラミング教育における最新の動向とそれの基づいた方向性を打ち出す必要性があると考えたからである．本報は１：様々なビジュアルプログラミングツール，２：オブジェクト指向プログラミング理解のためのビジュアルプログラミングの活用，３：教養教育としてのプログラミングの３部構成になっている．最後にここ数年，大学におけるプログラミング教育のあるべき方向性について考察する．

In the fourth report for the current title we show and discuss the explosive progress of visual programming education on tablet devices. In the previous reports we tried to show the advanced experiences for programming education on the devices. The progress for this field, however, diverges beyond our expectancy. Therefore, the understanding of the recent progress and the proper perspective for programming education are required. This report consists of three parts: 1 the variety of visual programming tools, 2 the advanced understanding for object oriented programming（OOP）using visual programming, 3 visual programming for liberal arts education. The prospect of programming education in university is finally surveyed for some years ahead.

キーワード：プログラミング教育，タブレット端末，iPad，プログラミング言語，ビジュアルプログラミング，iPadアプリ，オブジェクト指向プログラミング，デザインパターン
programming education, tablet device, iPad, programming language, visual programming, iPad app, object oriented programming, design pattern

１．はじめに

本学では平成23年度以来，iPadを教育に活用するさまざまな試みに取り組んできた[1)-9)]．特に著者らは平成24年度からタブレット端末特有のインターフェースを利用したプログラミング教育の可能性について実践と方法論の両面で研究を進めてきた[10)-14)]．平成28年度は，iPad上で機能する新たなビジュアルプログラミングアプリケーションのサーベイをおこなうとともに，ビジュアルオーサリングツールをプログラミングの事前教育としての電子書籍作成に適用した教育事例の紹介，ビジュアルプログラミングをオブジェクト指向プログラミング教育に利用するための具体策の提案をおこなう．本論文の構成は以下の通りである．2章では新しく登場したビジュアルプログラミングアプリケーションについて，コーディング中心のプログラミング教育への接続の観点からその特徴を紹介する．3章では，オブジェクト指向プログラミング言語Javaの基本文法を習得した学生がオブジェクト指向の設計原理に基づいたプログラムを作成できる技能を習得するための，ビジュアルプログラミング言語ScratchJr[16)]を用いるプログラミング教育の具体策を提案する．4章ではビジュアルオーサリングツールをプログラミングの事前教育としての電子書籍作成に適用した教

1) 名古屋文理大学短期大学部

育事例を紹介する．

2．ビジュアルプログラミングの動向

2．1　さまざまなビジュアルプログラミングツール

前報[12]で，タブレット端末を使ったプログラミング教育にビジュアルプログラミングが効果的であることを述べた．ビジュアルプログラミング言語は，プログラムコードの記述がなくても視覚的な操作でプログラミングが可能なプログラミング言語である．Web アプリケーションとして，または単体のアプリケーションとして提供され，ドラッグ＆ドロップ操作だけでプログラミング学習が可能である．ツールの中には完成したプログラムを既存のプログラミング言語のソースコードに変換できるものもある．主なビジュアルプログラミングツールを表1にまとめた．

文部科学省では，タブレット端末などに収めた「デジタル教科書」や小学校での「プログラミング教育の必修化」を，2020年度から導入する方向で検討がなされている．ただし，日本におけるプログラミング教育の公教育への導入は世界から遅れているのが現状である．アメリカでは既に2011年から政府が主導し，民間の NPO 団体らと協力しながらプログラミング教育の普及を目指し成果を上げている．

たとえば2013年に設立された「Code.org」[25]は，インターネット上で誰でもいつでもプログラミングが学べる環境「Code Studio」[25]を提供しており，ビル・ゲイツ氏やマーク・ザッカーバーグ氏などの著名人が出演した動画をネット上に公開するなどして，世界規模での利用者の拡大を図っている．

図１　Code.Org トップページ

2．2　Blockly と Bockly Games

Blockly[23]（表1）は，開発者向けに Google が提供する Web ベースのビジュアルプログラミング・エディタ環境を作るためのライブラリである．開発者はライブラリを利用して，既存の任意のプログラミング言語の文法に対応したブロック群を設計・作成し，その言語のためのビジュアルプログラミング･エディタを実装する．エディタの利用者（プログラミングの初心者）は，マウス操作だけでブロックを組み合わせてプログラムを作成し，そのプログラムを既存のプログラミング言語のコードに変換する．Google の公式サイトでは，Javascript，Python 等のコードに変換できるサンプルエディタが提供されている．一部日本語には対応していない部分もある．

表１　主なビジュアルプログラミングツール

プログラミングツール		Web／アプリ	コード変換機能	開発・提供	日本語対
Scratch 2.0[15]	スクラッチ2.0	Web	×	MIT メディアラボ	○
ScratchJr[16]	スクラッチジュニア	iOS・Android アプリ	×	MIT メディアラボ	○
Pyonkee[17]	ピョンキー	iOS アプリ	×	ソフトウメヤ	○
MOONBlock[18]	ムーンブロック	Web	○ JavaScript	秋葉原リサーチセンター	○
プログラミン[19]		Web	×	文部科学省	○
VISCUIT[20]	ビスケット	Web	×	NTT 研究所	○
JointApps[21]	ジョイントアップス	iOS・Android アプリ	×	デジタルハリウッド(株)	○
Smarlruby[22]	スモウルビー	Web	○ Ruby	まつもとゆきひろ	○
Blockly[23]	ブロックリー	Web	○ JavaScript,Python	Google	△
Code Studio[25]	コードスタジオ	Web	○ JavaScript	Code.org	○
Swift Playgrounds[26]	スイフトプレイグラウンズ	iOS アプリ	○ Xcode	Apple	×
Cargo-Bot[28]	カーゴボット	iOS アプリ	×	Two Lives Left	×
Hopscotch[29]	ホップスコッチ	iOS アプリ	×	Hopscotch Thechnologies	×

同じく Google がプログラミング初心者向けに提供しているプログラミング学習ゲーム「Blockly Games」[24] では，7種類のゲームが用意されている（図2）．これらのうちコード変換機能があるゲームについては，スムーズにキーボードを使った実践的なプログラミング言語の習得に誘導することができる．なお，「Blockly」上でブロックを組み立てる作業は，ロジックを PAD 図[30)-32)] で表現することと類似している．

図2　Blockly Games トップページ

従来のコーディング中心のプログラミング教育は，プログラミング言語の文法の説明や実行時の文法エラーに時間や気をとられがちであった．しかし「Blockly」では，ロジックを試行錯誤しながら組み立てることに集中でき，コード中の文法エラーの確認も簡単にできることから，大学でアルゴリズムを考えさせるプログラミング教育を行うツールとして有益であると考えられる．

2．3　Swift Playgrounds

「Swift Playgrounds」[26] は，2016年9月に iPad 向けの

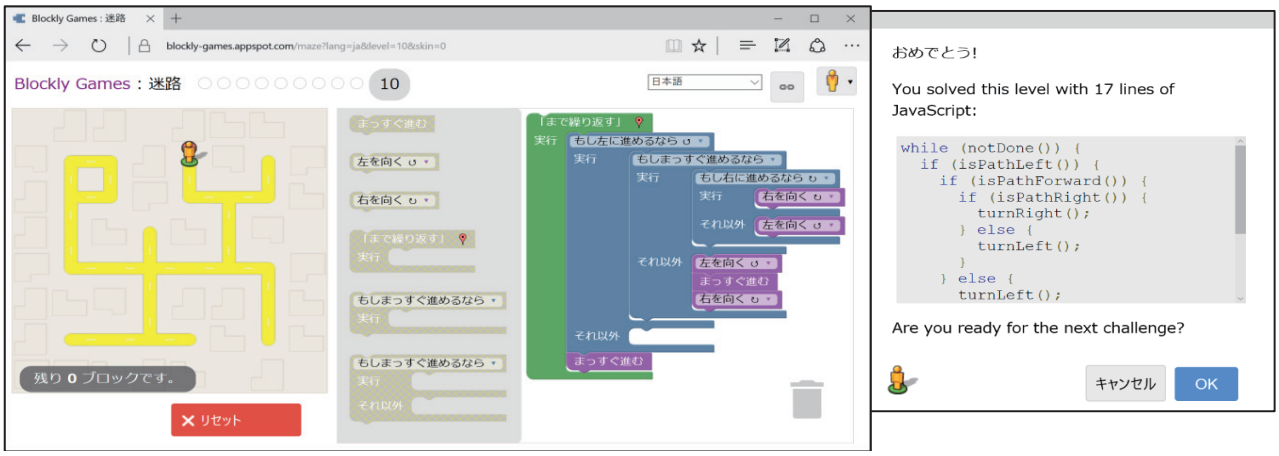


図3　Blockly Games 迷路10 実行例：どんな迷路にも応用できる汎用的なロジック

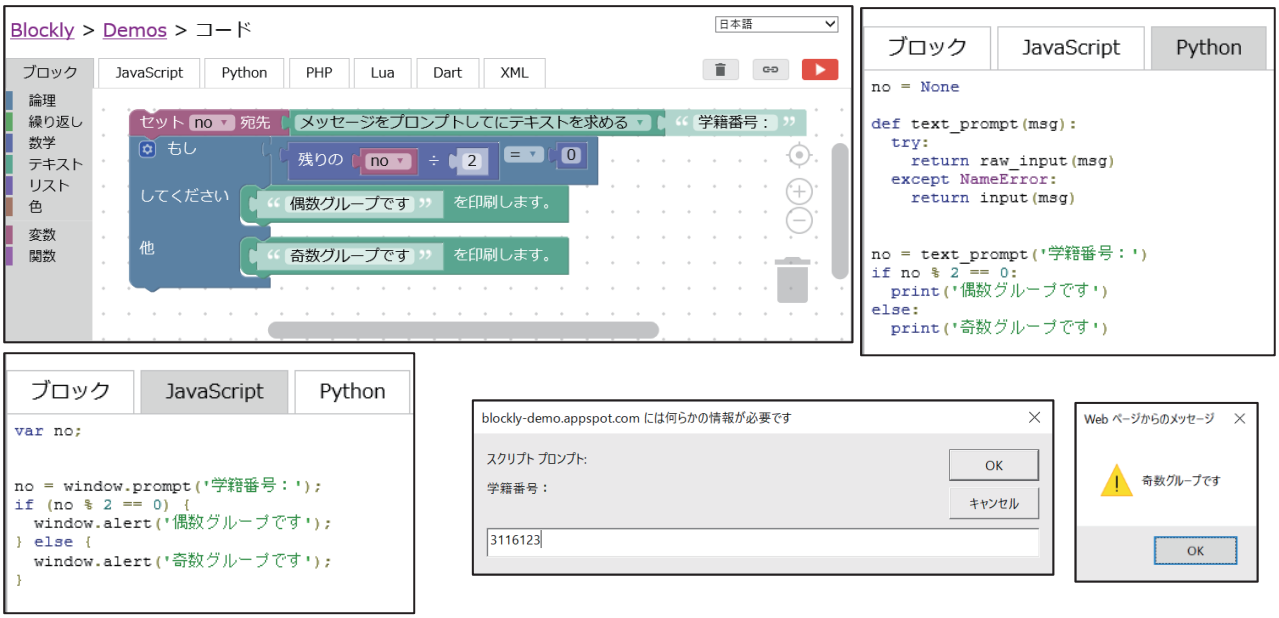


図4　Blockly で「学籍番号を入力して偶数／奇数グループに分けるロジック」作成・実行例

Apple 社製純正アプリとして提供が開始された．このアプリはプログラミング言語 Swift の学習用ツールで，パズルを解くための Swift プログラムを作成するチュートリアルモードと，自作の短い Swift プログラムを実行できるテストモードから構成される．アプリの利用者は，チュートリアルモードでは，迷路のようなパズルを順に与えられ，パズルごとに迷路上のキャラクタを途中の障害物を避けたり宝物を獲得したりしながらゴールに移動させるための Swift プログラムを作成する．図5に示されているように，チュートリアルモードの画面は，迷路やキャラクタのアニメーションを3D グラフィックで表示する画面右半分のアニメーション領域とキャラクタの動作用プログラムを記述するための画面左半分の編集用領域から構成される．迷路の盤面は「Code Studio」や「Blockly Games」では固定された平面だったが，「Swift Playgrounds」では3D グラフィックになっている上，拡大縮小・回転も可能で迷路上のキャラクタオブジェクトになりきって解答（のプログラム）を考えることができるようになっている．また，プログラムコードは画面左半分の最下部に用意されているコードが記入された各種ブロックをタップして編集用領域に挿入して作成してい

図５　Swift Playgrounds のチュートリアル画面

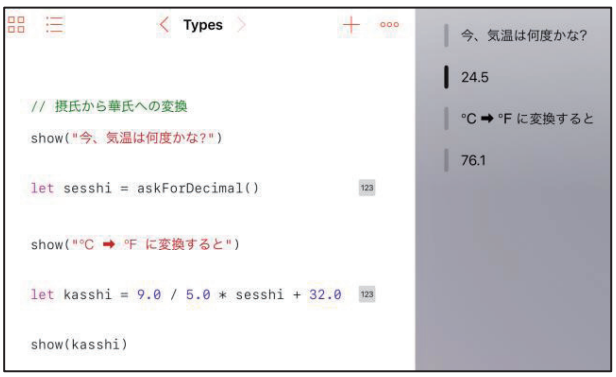


図６　テストモードでコードテスト例（摂氏華氏変換）

くしくみになっており，ブロックプログラミングではないものの，文字によるコード入力は必要最低限にとどまっていてタブレットでも学習しやすい環境になっている．

一方テストモードでは，比較的自由にコードを試すことができる．図6は，気温を摂氏で入力すると華氏に変換して出力するプログラムを「Swift Playgrounds」で作成し，実行したものである．かなりの制約はあるが，パズルを解くだけではなく，このように Swift のコードの働きを試してみることもできる．

なお，「Swift Playgrounds」と類似の環境として従来の iPad，iPhone 用アプリケーションの統合開発環境 Xcode にも「playgrounds」[27] というプログラミング練習用の環境があり，Swift のコードを実行できる．

３．ScratchJr から Java へ

３．１　オブジェクト指向技能習得の困難さと ScratchJr

前章では，ここ数年で急激に数を増やした iPad 上で利用可能なビジュアルプログラミングアプリが，プログラミング初心者にとって使いやすい洗練されたユーザインターフェースを持つようになり，導入・初級プログラミング技能を習得するための教育に適していることを見てきた．しかし，ビジュアルプログラミングはもともとオブジェクト指向的な考え方と整合性があり，その観点からビジュアルプログラミング教育の可能性や適用範囲は導入・初級プログラミング技能の習得にはとどまらないと考えられる．また，著者らは，iPad 上でのビジュアルプログラミングアプリによるプログラミング教育を，より高度なコーディング中心のプログラミング技能の習得に接続するための教育の方法論について考察してきた[10)-14)]．

一方，本学では国家資格である IT パスポート試験，基本情報情報技術者試験などの資格を在学中に取得することを学生に促すため，平成28年度入学生から IT 分野を強化した新カリキュラムを実施している．プログラム演習については，1年次後期開講科目「プログラミング入門」でブロックプログラミング言語 Scratch[15)]，軽量言語の Javascript と Python，2年次開講科目「プログラム演習 I・II」で手続き型言語である C 言語を通年，3年次前期開講科目「プログラム演習 III」でオブジェクト指向言語 Java を半年学ぶといったようにプログラミングを継続しておこなうカリキュラムとなっている．

ただしどのプログラミング言語にも言えるが，文法の習得や小規模なプログラムの作成技能は中～大規模なプ

ログラムのコードの理解や作成の技能の習得に必要だが十分ではなく，また習得すべきプログラミング言語のパラダイムが手続き型からオブジェクト指向に大きく変化する場合には，なおのことプログラミング技能の習得には大きな困難が伴う．そして，C++，Java，C#，Objective-C，Swift などのオブジェクト指向言語では，基本的な文法を用いてデータ処理をおこなうアルゴリズムのコードの実装だけでなくオブジェクト指向に関連する煩雑なコードの実装を強いられることもオブジェクト指向プログラミングを理解することを妨げている．

そこで，オブジェクト指向に焦点をあてたプログラミング教育の具体策として ScratchJr[16] を利用する教育を提案する．ScratchJr はキャラクタという限定されたものではあるが，最初から自然な形でオブジェクト（キャラクタ）が組み込まれており，オブジェクトに属する個々のメソッドも少数の「ブロック」の結合体としてプログラミングできる．したがって，ScratchJr によるプログラミングを適切に利用することで，Java のオブジェクト指向の設計原理を習得することが容易になるのではないかと考えられる．類似研究としてオブジェクト指向の考え方を習得するためのプログラミング教育ではないが，ビジュアルプログラミングからより高度なコーディング中心のプログラミングへと接続するためのビジュアルプログラミング教育の中等教育における事例を2つ挙げる．最初の事例では ScratchJr よりコーディング中心の言語に近く，記述能力も高いビジュアルプログラミング言語 Scratch により「変数」「条件付き実行（if 文）」「回数指定の繰り返し実行（for 文）」「条件付き繰り返し実行（while 文）」の概念を学習した後，Java（または C#）によるプログラミング学習をおこなう際，学習期間の短縮や理解度の深まりの効果が確認されている[33]．また，Scratch によるビジュアルプログラミング演習とコーディング中心のプログラミング演習を学期の途中で繰り返し行き来することで最終的にビジュアルプログラミングからコーディング中心のプログラミングへの概念的に飛躍のない移行に効果を上げている事例もある[34]．

3．2　オブジェクト指向の考え方と演習の目的

Web アプリケーションシステムをはじめとする中～大規模な情報システムでは，クラスを最小単位として，複数のクラスが集まってその機能を階層的に構成する．そこで便宜的に情報システムを次の3段階の粒度からなる構造物とみなす．

最も細かい粒度では，一つのクラスが一つの単位をなし，クラスはデータを処理するアルゴリズムを格納する容器とみなす．この粒度では情報システム全体の構造は捉えられない．一方，最も粗い粒度は情報システムの設計が準拠するソフトウェア・アーキテクチャ（フレームワーク，MVC モデルなどのモデルを想定）に基づくもので，アーキテクチャを構成する複数の階層がそれぞれ一つの単位をなす．本報ではこの粒度は扱わない．

中間の粒度は，互いにメソッド呼び出し等のメッセージパッシングを密におこなうクラスの集まりを一つの単位とするものである．同じ単位に属するクラス同士をオブジェクト指向の設計原理に基づき関連付けて設計するとき，現れやすい設計上の問題を解決するためのクラスの構成方法の定石集が「デザインパターン」[35),36)] である．この他にも中間の粒度のクラスの集まりについて，クラスの変更や追加に対して柔軟に対応できるようなクラス同士の関係を保持するための設計指針を集めたオブジェクト指向設計原理「SOLID 原則」[37)] 等もある．このようなオブジェクト指向設計・プログラミングの技能の習得が中～大規模な情報システムの構築に向けて必要とされる．

提案するオブジェクト指向に焦点をあてたプログラミング教育の適用先として3・4年のゼミナールにおける Java プログラミング演習を考えている．この演習では学生に関心を持ってもらいながらオブジェクト指向への理解を深めてもらうため，従来からキャラクタが複数出現するアクションゲームやシューティングゲーム等のゲームプログラムの設計・制作の演習を実施しているが，これをオブジェクト指向の設計原理を明示的に取り入れたもの[38),39)] へと改訂したい．そこで，本報ではゲーム内で複数のキャラクタを同時に扱うために適当なデザインパターンとして「ストラテジーパターン」[35)] を取りあげ，学生にまず「ストラテジーパターン」の ScratchJr プログラムの実装と Java プログラムの実装をおこなってもらい，次に両者の比較と対応するパーツの確認をおこなってもらい，最後に「ストラテジーパターン」の理解に至るというプログラミング演習の具体策を提案する．なお，デザインパターンの理解のためにビジュアルプログラミング言語を用いるプログラミング教育の具体策は，特定用途に開発されたツール[40)] を除いて調査の範囲内では見当たらなかった．

もしゼミナールで取り上げるゲームプログラムを「ストラテジーパターン」等のデザインパターンを利用せずに手続型的なプログラミングスタイルで実装するならば，ゲームに登場する複数のキャラクタに相当する各種

パラメータ群は，登場キャラクタ数×各キャラクタのパラメータ数のサイズを持つ巨大な2次元配列として実装・保持することになる．その結果，キャラクタの変更・追加にともなうプログラムの修正では，キャラクタが持つ振る舞いの種類の個数分の if 文の変更や if 文自体の追加などをおこなう必要が生じ，その作業はかなり大きなものとなる．そして，せっかくオブジェクト指向言語で実装しても，その意義はまったく失われてしまう．

3．3　ストラテジーパターンと演習の具体策

デザインパターンの一つ「ストラテジーパターン」は以下の通りである．機能は同じだが処理の方法が異なる複数のアルゴリズム（これをストラテジーと呼んでいる）をそれぞれストラテジークラスとしてカプセル化し，アルゴリズムの詳細をそのクラスの中に隠蔽する．クライアントクラスはこれらのストラテジークラスを利用する際，どのストラテジークラスもインターフェースを通して共通のメソッド名で呼び出すことができる．そして呼び出された各ストラテジークラスはそのクラス固有の処理をおこなう．このしくみをポリモーフィズム（多態性）と呼んでいる．

「ストラテジーパターン」を，実装を想定しているゲームプログラムに適用する際，各種ストラテジークラスを利用する側のクライアントクラスの機能を，ゲームを進行させるクラス（メインクラス）に割り当て，各ストラテジークラスの機能をゲームに登場する各キャラクタクラスに割り当てることにする．また，「ストラテジーパターン」にしたがい，ゲームに登場するキャラクタの保持／切り替えの処理とキャラクタの動作の起動を実質的におこなうコンテキストクラスを用意する．各キャラクタクラスが共通して持つメソッドを定義しているインターフェースクラスも用意する．

以上のしくみにより，メインクラスは，キャラクタ A（concreteCharA.java）（図9）とキャラクタ B（concreteCharB.java）（図10）の差異を意識することなくコンテキストクラスにキャラクタの切り替えやキャラクタの動作の起動の指示を与えるというシンプルな処理が可能となる．「ストラテジーパターン」をクラス図として表す（図7）．

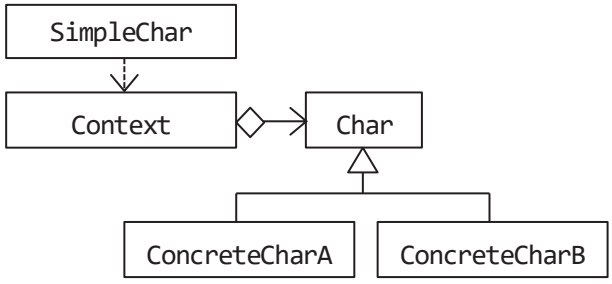


図７　実装するストラテジーパターンのクラス図

上記で説明した各クラスの Java プログラムは以下の通りである．この Java プログラム（図8～12）は図中に示す番号①～⑱の順に実行され，コンソールに「キャラクタ A が動作」「キャラクタ B が動作」の順に実行結果が表示される．

```
public interface Char {
    public abstract void move();
}
```

図８　Char.java（インターフェース）

```
public class concreteCharA implements Char{
    public void move(){                          // ⑨
        System.out.println("キャラクタAが動作"); // ⑩
    }
}
```

図９　ConcreteCharA.java（キャラクタ A）

```
public class concreteCharB implements Char{
    public void move(){                          // ⑰
        System.out.println("キャラクタBが動作"); // ⑱
    }
}
```

図10　ConcreteCharB.java（キャラクタ B）

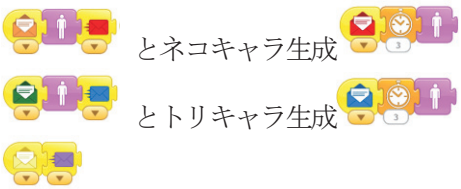
```
public class Context {
    private Char currentchar;
    public Context(Char chr) { // ②
        setChar(chr);          // ③
    }
    public void setChar(Char chr) { // ④, ⑫
        this.currentchar = chr;     // ⑤, ⑬
    }
    public void moveCurrentChar() { // ⑦, ⑮
        currentchar.move();         // ⑧, ⑯
    }
}
```

図11　Context.java（コンテキストクラス）

```
public class SimpleChar {
   public static void main(String[] args) {
      Context context = new Context(
                            new concreteCharA());// ①
      context.moveCurrentChar();                 // ⑥
      context.setChar(new concreteCharB());      // ⑪
      context.moveCurrentChar();                 // ⑭
   }
}
```

図12　SimpleChar.java（メイン／クライアントクラス）

表２　ストラテジーパターンの各クラス，ScratchJr の各キャラクタ，Java の各クラス・メソッドの対応関係

	ScratchJr プログラム	Java プログラム
メインクラス／クライアントクラス	キャラクタ[ネコ]，[トリ]，[M]と四角形 	SimpleChar.java context.setChar(new concreteCharA()) context.setChar(new concreteCharB()) context.moveCurrentChar()
コンテキストクラス	キャラクタ[C]と四角形  とネコキャラ生成 とトリキャラ生成	Context.java new Context(new concreteCharA())と setChar(new concreteCharA()) setChar(new concreteCharB()) moveCurrentChar(){currentchar.move();}
インターフェース	対応するキャラクタはなし	interface Char.java abstract move();
ストラテジーA	ネコキャラ 	ConcreteCharA.java move(){System.out.println("左回転");}
ストラテジーB	トリキャラ 	ConcreteCharB.java move(){System.out.println("右回転");}

以上の Java プログラムとほぼ同等の機能を持ち，実行時にもほぼ同様の振る舞いをする ScratchJr プログラムを以下で与える．先に ScratchJr プログラム実行時の画面を示す（図13）．まず図13の左上の四角形の枠で表されたメインクラス内で[ネコ]をタップして「コンテキストキャラクタ」[C]と「ネコキャラクタ」（キャラクタ A に相当）を生成する．次に「ネコキャラクタ」に動作をおこなわせるため[M]をタップして「ネコキャラクタ」を回転させる（図13）．キャラクタを「トリキャラクタ」に切り替えるには[トリ]をタップする．「トリキャラクタ」でも[M]をタップすると回転動作がおこなわれる．こうして共通のメソッド呼び出しに相当する共通の動作用ボタン[M]で現在出現している任意のキャラクタの動作を実現する．

なお，ScratchJr でのプログラミングは図14に示した画面上でおこなう．画面上部・中央はプログラムの試行画面，画面下部はプログラム編集用画面である．

(1) メインクラスに相当するキャラクタ

キャラクタ[ネコ]，[トリ]はタップされたら，それぞれ「ネコキャラ」「トリキャラ」を生成するためオレンジ／緑色メッセージをコンテキストキャラクタ[C]に送信する．

また，[M]はタップされたら，キャラクタ動作のため黄色メッセージをキャラクタ[C]に送信する．3つのキャラクタを囲む四角形の枠はメインクラスの範囲を指し示すためのもので背景画像として作成した（図15）．

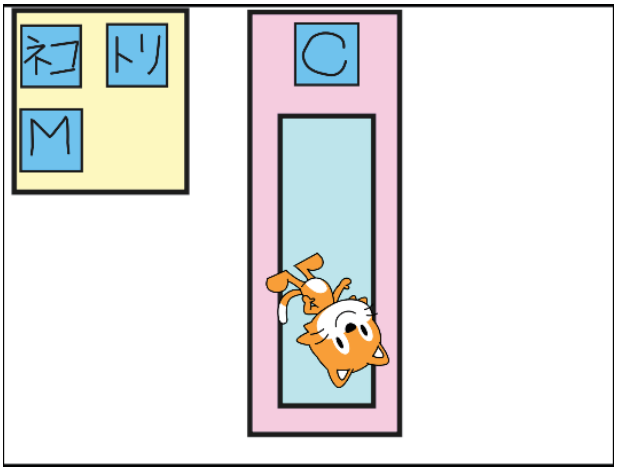


図13　ScratchJr 実行時の画面

図14　ScratchJr 編集用画面

図15　メインクラスに相当するキャラクタと四角形

(2) コンテキストキャラクタ

コンテキストキャラクタ C は ScratchJr 画面(図14)右上の「緑色の旗」をタップすることによりプログラム（図16の緑色の旗のブロックと透明化ブロックからなるモジュール）の実行が開始されると一度「透明化」されて存在しない状態になるが，キャラクタ ネコ または トリ のタップによるオレンジ／緑色メッセージを受信すると「実体化」(new に相当）し，さらにそのメッセージの種類に応じて「ネコキャラ」，「トリキャラ」のどちらかに「実体化」のため，それぞれ赤色／青色メッセージを送信する．また，キャラクタ M のタップによる黄色メッセージを受信すると，現在「実体化」しているキャラクタに動作のため紫色メッセージを送信する．なお C を囲む四角形の枠はコンテキストクラスの範囲を指し示すためのもので背景画像として作成した．

図16　コンテキストキャラクタと四角形

なお，Java プログラムでは new Context() で Context クラスを生成し，その後メソッド context.setChar(new concreteCharA()) で「ネコキャラクタ」を生成して Context クラス内の現在のキャラクタを保持する変数 currentchar に格納している（図11）．この機能に相当する ScratchJr のプログラムは ネコ の  ，コンテキストキャラクタの  とネコキャラクタの  というメッセージの連鎖として実装している．

ここで，ScratchJr は変数（内部状態）を保持する機能がないので，代わりとして ScratchJr 画面内のコンテキストキャラクタ C の直下に現在のキャラクタを生成・配置することで Java プログラムの currentchar に相当する機能を実現した．

(3) ストラテジークラスに相当するキャラクタ

「ネコキャラ」(oncreteCharA.java，キャラクタ A に相当）と「トリキャラ」(oncreteCharB.java，キャラクタ B に相当）は ScratchJr 画面(図14)右上の緑色の旗をタップすることによりプログラム（図17の緑色の旗のブロックと透明化ブロックからなるモジュール）の実行が開始されると一度「透明化」されて存在しない状態になるが，ネコ または トリ のタップによる赤色／青色メッセージが C 経由で受信されると「実体化」(new に相当)する．なお，「ネコキャラ」「トリキャラ」が同時に「実体化」している状況は好ましくないため，実装上の工夫として一方のキャラクタが「実体化」する直前にもう一方のキャラクタを「透明化」している（図17の上下段，左から3個目のモジュール)．いずれのキャラクタも紫色メッセージを受信すると回転動作をおこなう（図17の上下段，右端のモジュール)．

図17　ストラテジークラスに相当する2種類のキャラクタ

(4) ScratchJr プログラムと Java プログラムの対応

以上の方法で Java プログラムとほぼ等価な機能を ScratchJr プログラムで実現した．ScratchJr プログラムと Java プログラムの対応関係の詳細を表2にまとめる．

なお，ScratchJr では Java における他のオブジェクト内のメソッドを呼び出す機能がないので，代わりとして「メッセージパッシング」の機能を利用した．また，ScratchJr では Java におけるオブジェクト生成機能(new)や不要なオブジェクトをガベージコレクトする機能もない．一方，ScratchJr では，画面上に存在しているキャラクタは生成済みのオブジェクトとみなせ，その生成と消滅は「実体化」と「透明化」のブロックでおこなえる．そこで，プログラム起動時にキャラクタを一度「透明化」しておき，生成すべきときに「実体化」することでオブジェクトの生成（new）に相当する機能を実現した．

(5) プログラミング演習の手順

提案するプログラミング演習では，学生は ScratchJr プログラムを先に実装し，動作確認を何度もおこなって「ストラテジーパターン」のしくみを把握してもらう．次いで Java プログラムを実装する．その後，両者を比較し，ScratchJr プログラムで把握しておいた各パーツが Java プログラムのどのクラスやメソッドに対応する

のかを確認してもらう．このように学生は把握しやすい ScratchJr プログラムを何度も参照しながら，「ストラテジーパターン」に基づく Java プログラムのしくみを確認することで，コーディング中心のプログラミング言語におけるオブジェクト指向設計の考え方の一端を直観的に理解するに至る．

3．4　まとめと今後の課題

本報では，ScratchJr を用いてデザインパターンのうち「ストラテジーパターン」と同等の機能を果たすプログラムを作成し，Java プログラムと比較することでオブジェクト指向設計の一端を理解するためのプログラミング演習の具体策を提案した．

実際に，ゼミ・授業等でこのプログラミング演習を実施し，プログラミング技能習得に与える効果を探る予定である．また，ScratchJr の文法上の制約，特にメッセージ送信を6種類しか持たないことやキャラクタの内部状態を保持する変数の機構が備わっていないことなどの制約が実装上の問題となる可能性はあるものの「ストラテジーパターン」以外のデザインパターンについても簡素な ScratchJr プログラムの実装を試み，オブジェクト指向の考え方に関して，より広い範囲をカバーできるようなプログラミング演習の具体策を提案する予定である．

4．Publishing on Mobile

著者らは iPad を用いたプログラミング教育に関して，Programming on Mobile と称して講義やゼミナールでの経験を中心に報告してきた[12),13)]．本報でも2，3章において，タブレット端末の利用で効果を発揮するビジュアルプログラミングと従来のコーディング中心のプログラミングとの関連と連携について述べてきた．しかしプログラミングは仮想空間を支配する法則であるが故に何にでも応用できる反面，明確な方針を持っていないプログラミング初学者にとって，何を作ってよいか判断に困ることにもなる．そこで本章ではプログラミング以前の仮想空間を複数のカードの集まりに見立て，そのカード間の遷移で仮想空間を構築する演習について報告するとともに，その課題について考察する．なお，このカードは，ハイパーリンクが貼られた電子書籍の1ページと考えることもできる．

ハイパーリンク機能を備えた作品を作る際に，1990年代には Flash 以前のオーサリングツールである HyperCard[41)] がよく用いられていたが，現在 HyperCard を利用することはできない．HyperCard 互換ソフトウェアも存在するが，HyperCard が用いられていた当時と現在では利用するコンピュータの形態や処理能力が大きく異なるため，HyperCard を現在に復活させただけでは時代に即した効果を上げることが難しいと考えている．また HyperCard では，カードを制御する HyperTalk によって自然言語のようにカードを制御できることが特徴であるが，現在用いられている主要なプログラミング言語との差異が大きく，プログラミング言語の習得における連続性から考えて問題が残る．著者らは以前 iPad 上のプレゼンテーションソフトウェアである Keynote を用いたオーサリングツール的な利用法について紹介した[2)]．本章では，Programming on Mobile に倣い iPad を利用した電子的作品の制作を Publishing on Mobile と称して報告することにする．ただし，純粋な著作物を作成するという目的よりも，Programming on Mobile に連続的に繋がるように，プログラミングの要素を取り入れた作品の制作に焦点を当てることにする．

電子的作品制作の演習は２年次生対象の少人数のゼミナールで2年間にわたって行った．その概要を以下に紹介する．具体的には Book Creator[42)] という iPad 上のアプリケーションを利用した．Book Creator は名前の通り iPad や iPhone 上の iBooks 上で閲覧可能な作品を作るアプリケーションである．ただし，文章主体の電子書籍を作成するのではなく，文字，静止画，動画，音声等を等しく扱うオーサリングツールとしての性格が強い．iPad 上で作成した作品は，直ちに同じ機器上の iBooks で閲覧することができる．またメール等を利用して iPhone に転送すれば，iPhone 上の iBooks で閲覧することもできる．

Book Creator のプログラム的な制御機能は，オブジェクトがタップされた際のページ遷移機能のみという単純なものである．しかし作成した Book のページ間にハイパーリンクを貼ることで統一性のある作品を制作できる．プログラミングでページ遷移のようなジャンプ構造を理解することは比較的簡単である．しかしここで強調すべき重要ポイントは，ページ間にハイパーリンクを貼る機能そのものではなく，作成したページというデータをまず優先して考え，その後で各ページを効果的にコンテンツ全体の構造の中で関連づけるにはどのようにすればよいかを考えることである．ゼミナールでは作成する Book に合わせて iPad で絵を描く学生やネットでの検索によりコンテンツ作りに必要な情報を集める学生がいるなどさまざまなアプローチでの作品作りがおこなわれた． iPad のドローイングアプリを用いて作成された手

書きの絵本，アルバイト先の仕事で役立つ Book などがその例である．

提出された中で最も印象的な作品は，二択択一のクイズ Book であった．このクイズ Book では質問への解答の正否によってエンディングが異なるように工夫がなされていた．Book Creator の限られた機能でよく作成していると感心する一方で，Book Creator を簡易版オーサリングツールとして使うには限界があることも明らかになった．本来ならば解答の正否に基づいて難易度が適切なクイズが出題され，さらに正解総数に応じてエンディングでの総獲得ポイントと総評の表示ができることが望ましい（図18）が，ページ間の遷移機能のみで適切なエンディングを用意するのは難しい．このような機能の不足は，Book Creator だけの欠点ではなく，クイズ Book を簡単に生成できるアプリケーションが iPad 版，PC 版含めて見当たらないことが根本の原因である．もちろん PC でプログラミング言語を用いてクイズ Book を精巧に作ることは可能である．また電子ノベル作成ツールを使用しても作成可能かもしれない．しかし HyperCard でできたような，文字，画像，音声等を自由に配置し，それらに機能を付け加えていくことで目的のスタックを作成するスタイルに比べ難易度は高い．

電子的作品制作の課題は具体的な作品を作りながら，その作品を豊かにする機能が存在すれば取り込んでいくという姿勢に則っている．これはプログラミング演習においても共通する方向性である．しかし，プログラミングでは言語の文法を理解して使いこなすまでにかなりの訓練を必要とすることから作品（プログラム）を俯瞰して考えることができるまでの道のりは遠い．したがって，

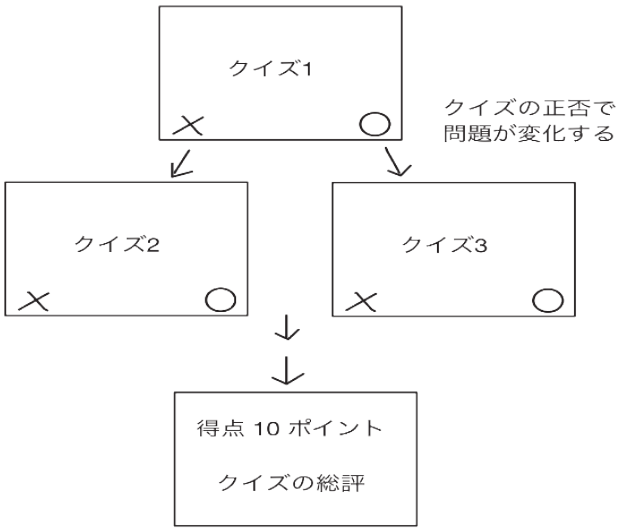


図18　クイズ Book の概要

その前段階の演習課題として，電子的作品制作の経験を通してデータの構造とデータ間の関連を結びつけるプログラム的な制御方法についてまず“当たり”をつけることの意義は大きいと考えている．

本章の最後に，どのようにすれば簡単にページ遷移や得点表示が可能なクイズ Book の作品が構築できるかについて考察してみたい．3章では iPad 上で稼働する，オブジェクトにビジュアルプログラミング言語を用いて機能を付加する ScratchJr を利用した教育研究について報告をおこなった．そのため ScratchJr を基にクイズ Book を作成するというアイデアが浮かぶ．しかし ScratchJr はページ（場面）ごとにオブジェクトが独立しており，また得点などの数値をページをまたいでメッセージとして送る方法がないため，クイズ Book の作成にはうまく活用できないことがわかった．次に思い浮かぶのは，パソコン用 Scratch の iPad ポート版である Pyonkee[17] を活用することである．機能が豊富であるため実現の可能性はあると考えているが，どのようにオブジェクトを構築するか工夫が必要で検証の途中である．クイズ Book の作成に成功すれば何らかの形で報告を行う予定である．かつて iPad で NovoCard というアプリケーションが存在した．これは HyperCard と同じコンセプトを持っているが，制御言語として Javascript に類似のものを使用する一部モダン化された HyperCard 類似アプリケーションであった．その後残念ながら NovoCard は配布が中止された．iPad のような機動性のある機器で稼働する新たな HyperCard 類似アプリケーションの登場を期待している．

5．プログラミング教育の展望

本報では，ビジュアルプログラミングの新しい流れ，特にコーディング中心のプログラミングに接続可能なビジュアルプログラミングアプリの展開について紹介した．また，オブジェクト指向に基づくプログラミング技能を習得するためのビジュアルプログラミングを用いるプログラミング教育の具体策を提案した．一方で，ビジュアルオーサリングツールを利用した教育の事例を紹介し，それがプログラミングの事前教育として適当であろうことを示唆した．

従来，ビジュアルプログラミングツールはプログラミング初心者のためのツールとして位置づけられることがほとんどであったが，本報ではプログラミングの事前教育からオブジェクト指向習得のための教育に至るプログラミング教育の各段階に対して，各種のビジュアルプロ

グラミングツール／ビジュアルオーサリングツールを適材適所で使い分けて利用していくというプログラミング教育の新しい方向性を示すことができた．

ただし，本報での試みはプログラミング教育の各段階におけるビジュアルプログラミングツールのいくつかの利用を提案するにとどまっている．ビジュアルプログラミングツールをプログラミング教育のより多くの場面に利用できる可能性は高いと考えられる．ツールを利用できる教育機会の探求とそこでの教育の具体策の提案および実践が望まれる．特に，本来オブジェクト指向を習得する際，学ばれるべき必須のデザインパターンの種類は10～20種類あるが，現状ではオブジェクト指向の基盤となる考え方である「ポリモーフィズム」と1種類のデザインパターン「ストラテジーパターン」しか具体策を提案できていない．その他のデザインパターンについてもビジュアルプログラミングツールを用いた直観的で理解しやすい教育の具体策を提案することが課題となる．

上で述べた「ポリモーフィズム」をビジュアルプログラミングで実装するアプローチは，プログラミング技能の高度化を図るものであった．一方4章で示した Publishing on Mobile は，プログラミングの素養を持った人材を育てる，いわゆる裾野を広げるアプローチである．ただし今後起こり得るであろうプログラミングを取り巻く環境の劇的な変化を考慮すると，単にプログラミングの重要性や楽しさを啓蒙するだけではなく，これらの変化に充分対応可能な内容にたえずプログラミング教育を更新して行くべきであると考えている．今日のプログラミングにおいては，基本的な制御構造はもちろんのこと，オブジェクト指向のみならず AI 技術をも含めた高度な概念の理解と実践が求められる．これらは順を追ってしっかりと学んでいくことが望ましいと考えている．

これらを学ぶにはかなり長期にわたる学習が必要となるように思われる．しかし，3章で報告した ScratchJr でのメッセージのやり取りを応用が効く程度に理解していれば，ScratchJr でスプライトと呼ばれるキャラクタたちを組み合わせてプログラムを作成することが，未来の AI オーサリングツール上で AI 機能が付加された AI エージェントたちを構築していく作業に置き換わったとしても，個々の AI エージェント内のモジュール性の理解や AI エージェントたち全体の構造およびそれらを統合する上位階層の制御プログラムの理解にはなんら問題はないであろう．むしろ AI エージェントの機能により現在の ScratchJr では実現できないプログラミングが可能になると期待される．

情報システムが AI 技術を搭載した高度で巨大なものに発展するのに合わせて，それらを扱うコンピュータサイエンスの領域も急拡大することになる．結果として学生は，この広大な領域の多岐にわたる概念群を大学教育という限られた期間内で習得せざるを得ない．そのための教育は，コアとなる概念群を含む教育コンテンツの選択と，それらの概念群を可視化して見通しを与え理解へと誘導する各種のビジュアルプログラミングツール／ビジュアルオーサリングツール／ビジュアライジングツールを効果的に活用していくことが鍵となると考えている．

参考文献

1）本多一彦：「モバイル機器の変遷から情報教育機器としての iPad を考察する」，名古屋文理大学紀要，11，97-104（2011）

2）森博，田近一郎，杉江晶子：「タブレット PC を活用したマルチメディア教育の試み」，名古屋文理大学紀要，12，97-104（2012）

3）佐原理，大橋平和，長谷川旭，長谷川聡，KAISER Meagan：「タブレット端末による学校教育現場向け多言語情報配信システム」，名古屋文理大学紀要，12，105-112（2012）

4）長谷川旭，佐原理，尾崎志津子，本多一彦，山住冨也，長谷川聡：「名古屋文理大学における iPad 導入とアクティブラーニング」，モバイル学会研究報告集，7(2)，45-48（2011）

5）長谷川旭，長谷川聡，本多一彦，山住富也，佐原理：「大学教育でのタブレット端末の利用とその効果―iPad を無償配布した名古屋文理大学における学生意識」，コンピュータ＆エデュケーション，31，70-73，（2011）

6）尾崎志津子：「iPad を活用したオンライン英語多読の導入―名古屋文理大学情報メディア学科における事例―」，コンピュータ＆エデュケーション．；32，49-52（2012）

7）長谷川聡：「ソーシャルリーディングとソーシャルラーニング」，現代の図書館，50(2)，114-120（2012）

8）長谷川旭，小橋一秀，山住富也，長谷川聡：「タブレット端末の教育利用と情報インフラ：名古屋文理大の iPad 無償配布と大学図書館」，医学図書館，59(3)，186-191，（2012）

9）斉藤徹，河原潤，高下義弘：「教える！名古屋文理大

学」,『iPad で現場を変える !』, 日本経済新聞出版, 132-147 (2011)
10) 本多一彦, 田近一郎, 杉江晶子, 森博 :「タブレット端末を活用したプログラミング教育」, 名古屋文理大学紀要, 13, 85-92 (2013)
11) 田近一郎, 本多一彦, 杉江晶子, 森博 :「タブレット端末を活用したプログラミング教育 (2)」, 名古屋文理大学紀要, 14, 75-86 (2014)
12) 田近一郎, 本多一彦, 杉江晶子, 森博 :「タブレット端末を活用したプログラミング教育 (3)」, 名古屋文理大学紀要, 15, 17-27 (2015)
13) 田近一郎, 本多一彦, 杉江晶子, 森博 :「IT 教育のためのプログラミングオンモバイル」, モバイル学会研究報告集, 11, 143-146 (2015)
14) 本多一彦, 田近一郎, 杉江晶子, 森博 :「プログラミング・オン・モバイル」, メディア教育シンポジウム (2016/2/6)
15) Scratch 2.0,
https://scratch.mit.edu/scratch2download/ より2016年11月3日検索
16) ScratchJr,
https://itunes.apple.com/jp/app/scratchjr/id895485086 より2016年11月3日検索
17) Pyonkee,
https://itunes.apple.com/jp/app/pyonki/id905012686 より2016年11月3日検索
18) Moon Block,
http://www.moonblock.jp/docs/ より2016年11月3日検索
19) プログラミン,
http://www.mext.go.jp/programin/ より2016年11月3日検索
20) VISCUIT,
http://www.viscuit.com/ より2016年11月3日検索
21) JointApps,
http://www.jointapps.net/ より2016年11月3日検索
22) Smarlruby,
http://smalruby.jp/ より2016年11月3日検索
23) Blockly,
https://developers.google.com/blockly/ より2016年11月3日検索
24) Blockly Games,
https://blockly-games.appspot.com/ より2016年11月3日検索
25) Code Studio,
https://code.org/ より2016年11月3日検索
26) Swift Playgrounds,
http://www.apple.com/swift/playgrounds/ より2016年11月3日検索
27) playgrounds,
http://help.apple.com/xcode/mac/ より2016年11月3日検索
28) Cargo-Bot,
https://itunes.apple.com/jp/app/cargo-bot/id519690804 より2016年11月3日検索
29) Hopscotch,
https://itunes.apple.com/jp/app/hopscotch-learn-to-code-creatively/id617098629 より2016年11月3日検索
30) PAD 図 : ProblemAnalysisDiagram は日立製作所で発明され ,1979年に公表された流れ図に代わるプログラム図式を中心とした構造化プログラム開発技術
http://www.hitachihyoron.com/jp/pdf/1986/05/1986_05_02.pdf より2016年11月3日検索
31) 二村良彦 :「プログラム技法ー PAD による構造化プログラミングー」, オーム社 (1984)
32) 川合敏雄 :「PAD プログラミング」, 岩波書店 (1985)
33) Armoni, M., Meerbaum-Salant, O.& Ben-Ari, M.
From Scratch to “Real” Programming. ACM Trans on Computing Education. 14, 4 (2015), 25.
34) Weintrop, D. & Wilensky, U.
Bringing Blocks-based Programming into High School Computer Science Classrooms. Annual Meeting of the American Educational Research Association (2016)
35) 結城浩 :「増補改訂版 Java 言語で学ぶデザインパターン入門」, SB クリエイティブ, (2004)
36) 上田勲 :「プリンシプル オブ プログラミング3年目までに身につけたい一生役立つ101の原理原則」, 秀和システム (2016)
37) ロバート・C・マーチン, 瀬谷啓介 :「アジャイルソフトウェア開発の奥義 第2版 オブジェクト指向開発の神髄と匠の技」, SB クリエイティブ, (2008)
38) Ryo Asai :「達人プログラマーを目指して Java のクラスとオブジェクトについて再度解説を試みる」, http://d.hatena.ne.jp/ryoasai/20111217/ より2016年10月5日検索
39) 増田亨 :「システム設計日記 オブジェクト指向プログラミングの教え方?」, http://masuda220.jugem.jp/?eid=457 より2016年10月5日検索

40）石川裕季子，松澤芳昭，酒井三四郎：" オブジェクト指向言語におけるポリモーフィズムの概念を理解するためのワークベンチの試作 "，教育システム情報学会誌，Vol.31，No.2，pp.208-213，2014.
41）Hypercard，
http://hypercard.org より2016年11月3日検索
42）Book Creator，
https://itunes.apple.com/jp/app/book-creator-for-ipad/id442378070 より2016年11月3日検索