情報処理学会研究報告
IPSJ SIG Technical Report

PenFlowchart の開発

中　西　　渉[†1]

本校の情報 B の授業では，初学者向けプログラミング学習環境である PEN を利用してきた．入力支援ボタンで構文が簡単に入力できるのだが，慣れない生徒はその構造を壊してしまってエラーを起こしてしまう．そこでフローチャートをマウスで作ることで PEN のプログラムを生成するプログラム PenFlowchart を開発し，2011 年度の途中から PEN と併用した．開発の経緯と，生徒に使用させた上での感想について報告する．

Development of PenFlowchart

Wataru Nakanishi[†1]

We have used ‘PEN (Programming Environment for Novice)’ in our classes. Program code can be entered easily with input-assist-buttons, but some students erase the lines by mistake, and they break its block-structures and invite errors. So I developed a program ‘PenFlowchart’, which can produce programs for PEN by making flowcharts using mouses. I will report circumstances of development and impressions when using it in our classes.

1. はじめに

PEN[1] は大阪学院大学情報学部西田研究室と大阪市立大学大学院創造都市研究科松浦研究室の共同プロジェクトとして開発されている，初学者向けのプログラミング学習環境である．用いているプログラミング言語は，大学入試センターの入試科目「情報関係基礎」で用いられている手順記述言語 DNCL を拡張した xDNCL[2] である．xDNCL は日本語をベースにしているため初学者にも読みやすいが，手入力するには変換が面倒であるし，ささいな表記の違いによって文法エラーが起こりやすい．そこで PEN には入力支援ボタンが用意されていて，ボタンを押すだけで制御構造の雛形がエディタ画面に挿入できる（図 1）．これによって，学習者は少ない手数でプログラムを入力することができる[3]．

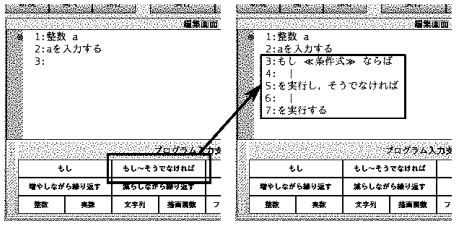
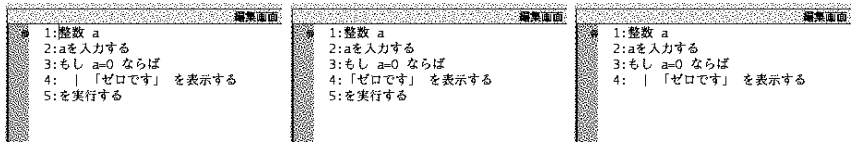


図 1　入力支援ボタン

しかし，生徒は入力支援ボタンで入力された構文をブロック構造として理解できず，図 2 のようにインデントやブロックを壊してしまい，エラーで行き詰まってしまう（インデントはなくてもエラーにはならないが，見通しが悪くなる）．さらに何人かは見本のプログラムを，（入力支援ボタンを使わずに）ワープロを打つように手入力してしまうため，プログラムを行単位で見てしまい，ブロック構造を理解しない．

筆者は PEN における入力支援ボタンは入力を簡単にすることよりもむしろ，プログラムのブロック構造を意識するためにあると考えているので，生徒がそれを活かせずにこのような状況に陥ってしまうのは非常に残念である．そこで，フローチャートを作ることで PEN のプログラムを生成することができれば，この問題がいくらか解決できるのではないかと考え，PenFlowchart を開発した．



(a) 正しい表現　　(b) インデントの破壊　　(c) ブロックの破壊

図 2　生徒がやってしまう間違いの例

†1 名古屋高等学校
Nagoya High School

2. PenFlowchart の概要

2.1 実 行 画 面

実行画面は図 3 の通りである．左上の部品は左下のフローチャートにドラッグ&ドロップすることで追加できる．そこで使われる値や式などの編集は，右クリックでメニューを呼び出すことで行なえる（ダブルクリックでも編集できる）．変数宣言は右上の窓に変数名を列挙することで行なう．部品の削除やカット&ペーストは右クリックで出るメニューで行なう（ただし分岐やループはその終わりまでをブロックと考えてまとめて処理される）．

PEN のプログラムは右下に表示される．これを実行する方法は

- 右下のプログラムをコピーして PEN の編集画面にペーストする．
- 「PEN 形式で出力」したプログラムを PEN でロードする．
- メニューから「実行」を選択する（編集不可能な状態で PEN が起動する）．

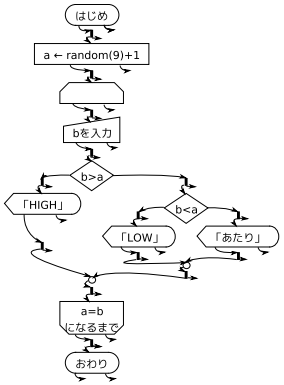
のいずれかである．

作成したフローチャートは `penf` という拡張子で保存される（中身は XML）．前述したように PEN 形式のプログラムは出力できるが，PEN 形式のプログラムを読み込むことはできない．フローチャートは EPS 形式で画像として出力できる．

2.2 データ構造

フローチャートは各ノードが左右に次ノードを持つような単方向リストとして管理されている．各記号や縦棒が一つのノードである．たとえば図 3 のフローチャートであれば右図のようなイメージである。何も指していない矢印は `null` を指している．

`endif` を表す○や縦棒をノードにしたのは，その方が管理が簡単になるからである．いずれ 6.1 にあげた「繰り返しを抜ける」を実装するとしたら，このことが役に立つであろう．



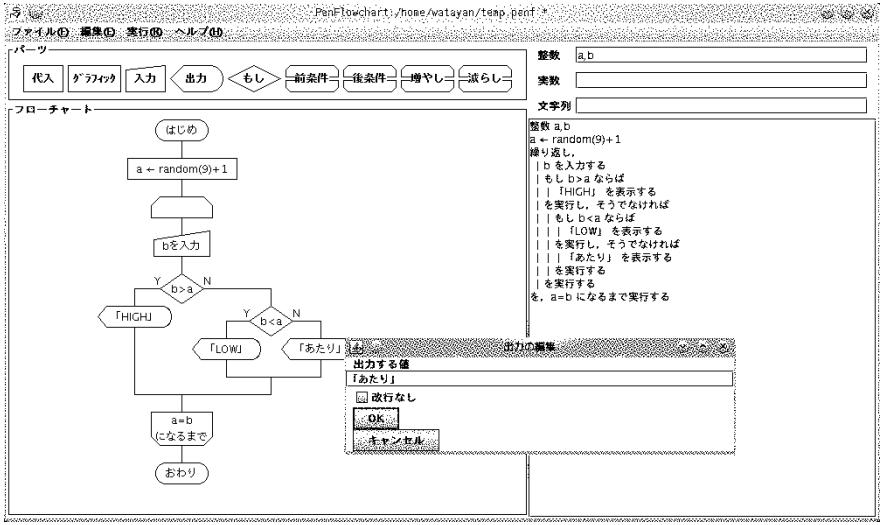


図 3 PenFlowchart の実行画面

3. 開　　発

3.1 開 発 環 境

PEN が Java で作られていること，本校の情報教室では端末に Linux を使用していることなどから，開発言語は Java を用いた．プログラムを作成するにあたって 4) などのサイトを参考にし，また Java のメモリ管理については筆者自身が Anchor Garden[5] を用いて勉強した．

3.2 バージョンアップ

2011 年 8 月下旬に開発を開始し，8/29 にバージョン 0.1 を自サイト[*1]で公開したが，これはマウスで作ったフローチャートを元に PEN のプログラムを出力できるだけで，セーブ・ロードもできないものであった．その後，使った方からの助言を参考に開発を進め，9/21 にリリースしたバージョン 0.5 から授業で使い始めた．この時点では，作ったプログラムを PEN 形式のプログラムを出力させて PEN で読み込むか，PEN の編集画面にコピーし

⋆1 http://watayan.net/prog/

ないと実行できなかったが，その手間は筆者自身も面倒と感じていた．そこで PEN のソースコードをまるごと取り込んで，直接実行できるようにしたバージョン 0.6 を 10/21 にリリースし，授業ではこれを最後まで使った．

このバージョン 0.6 では，ブロックの中に新しいブロックを作ることはできるが，既にあるブロックを新しいブロックに入れることはできない．たとえばジャンケンのプログラムを作ったあとでこれを 3 回勝負にしようと思ったときには，新たに作ったループの中に既に作ったジャンケンのプログラムをもう一度ゼロから書き直さなくてはいけないということだ．PEN の入力支援ボタンがそういう構造になっているのだから，筆者は当初それが正しいと考えていたのだが，PEN を使わずに PenFlowchart を使った人からはこの点が不満だとの意見をいただいた．そこでグラフィック命令をサポートしたバージョン 1.0（10/19）の直後，ブロック単位でカット&ペーストができるようにしたバージョン 1.1 を 10/26 にリリースした（本稿を執筆中に見つけたバグを修正したバージョン 1.2 を近日中にリリースする予定）．

本校の授業でバージョン 0.6 を使いつづけたのは，学期途中から使い勝手が変わってしまうのでは生徒が混乱するだろうと考えたからである（0.5 から 0.6 にしたのは，直接プログラムが実行できるという利点を重く見たため）．次年度の授業では初めから最新バージョンを使用する予定である．

4. フローチャート

4.1 他の表現方法

このプログラムを作るにあたり，フローチャート以外の表現方法についても考慮した．PEN で作ったプログラムは必然的に構造化されているので，それを表現するのであれば PAD などの構造化チャートの方が適しているのではないかと考えたからだ．しかし，高校の情報の教科書で扱われているのはフローチャートばかりであり，PAD などは教える側（筆者自身も含めて）にもあまり馴染みがないと考えたので，やはりフローチャートで表現することにした．しかし，プログラムの実装に関していえば，PAD の方がずっと楽だっただろうと思われる．

4.2 xDNCL，PEN との相違

xDNCL ではキーボードからの入力は「《変数》 ← `input()`」になっており，PEN では「《変数》を入力する」が許容されているものの，入力支援ボタンで入力されるのは前者である．

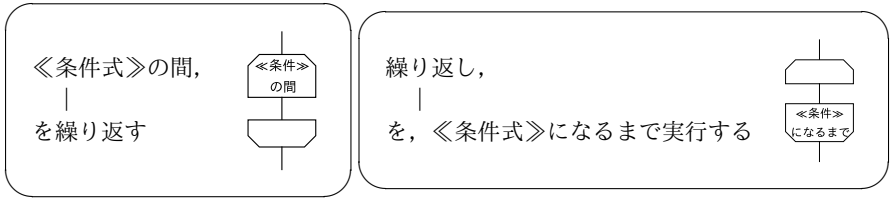
しかし代入と入力を取り違える生徒は多い．それが両方とも「←」で表現されているので，入力支援ボタンの「代入」で右辺を `input()` にすることで「入力」にしてしまうということが多く見受けられた．そこで本校の情報教室の端末にインストールした PEN は入力が「《変数》を入力する」になるように入力支援ボタンを書き換えてあるし，PenFlowchart でもその表現を使用することにした．フローチャートの記号を，手操作入力と処理で違うものにしておきたかったというのも理由の一つである．

4.3 JIS との相違

フローチャートの書き方は JIS[6] に定められたものがあるが，PEN のプログラムを生成することが主目的なので，いくつかの点でそれとは異なる表現を用いている．

フローチャートでは多岐の分岐も許されているが，授業で「条件は Yes/No で答えられるもの」と説明していることにあわせて 2 つの分岐しか用意していない．PEN にも `else-if` の構文があるのだが，フローチャートの多岐の表現がそれを意味しているかというと疑問である⋆1．

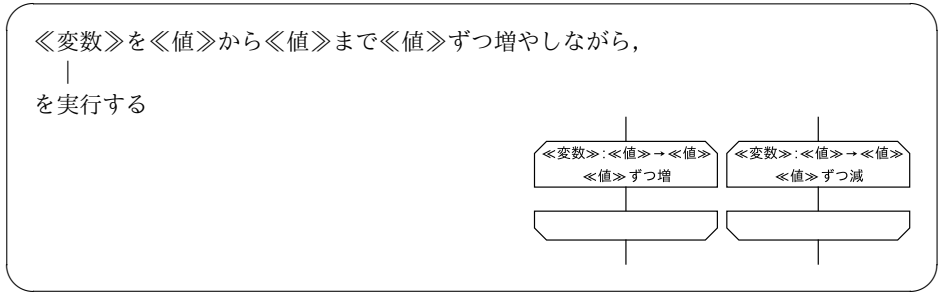
ループ端記号には「ループ名，初期化，増分，終了条件」を書くのが正式であるが，PEN にループ名はない⋆2．また xDNCL のループは条件判定が始端にあるものは継続条件「〜の間」，終端にあるものは終了条件「〜になるまで」の構文になっている．これを JIS にしたがって終了条件に統一してしまうと PEN での表現との不一致のために生徒が混乱することが予想されるので，「〜の間」「〜になるまで」のままの表現でループ端に入れることにした．



xDNCL では，`for`〜`next` 型のループで増分が負の値であるときには「〜ずつ減らしながら」という構文を使う．「〜まで」という終了条件を不等号で評価するため，「`-1` ずつ増やしながら」では正しく動かないからだ．そこで PenFlowchart でも増分を負で表すのでなく，「〜ずつ増」「〜ずつ減」と表示することにした．情報の教科書でも出版社によってさまざま

⋆1 6.1 にあげた `switch` 構文がこの形に該当するだろう．
⋆2 ループの始端と終端の対応がわかりやすいように表示してほしいという要望はあるので，何らかの方法で対応しようとは考えている．

な表現がされているので，JIS に従うことよりも直観的に意味が通じることが重要だと考えたからである．



4.4 ループ端記号

ループ端記号は 6) で 1986 年に追加された，比較的新しいものである．そのこともあってか，この記号を使わない人もいる．

たとえば「動くフローチャート」[7] はフローチャートそのものがプログラムとなって実行できるものであるが，これはループを分岐と goto で表現している．作者にその意図をうかがったところ「コンピュータの内部処理の通りに表現するのが良い」ということだった．その考えには納得させられる点があるが，筆者は PEN のプログラムの通りにフローチャートが書かれることを重視したのでループ端記号を用いることにした．

プログラムが繰り返し構文なのに，フローチャートはループ端記号でなく分岐と goto で表現されるものもときどき見かけるが[⋆1]，これをループの構文に読み替えることはそれほど簡単なことではないと考えている．

5. 生徒の反応

筆者が授業を担当しているクラス（高校 1 年生 7 クラス，男子 264 名）ではほぼ毎時間，課題のプログラムを PEN または PenFlowchart で作成し（どちらを使うかは生徒の選択），できあがったプログラムをメールで提出させていた．最後の授業で，PEN と PenFlowchart の使い分けについて記名のアンケートを行なった（回答数 248）．主にどちらを使ってきたか，使い分けるにあたってどういう点に着目したか（複数回答）という質問への回答は次の通りであった．

主に PEN	146
主に PenFlowchart	65
課題によって使い分ける	37

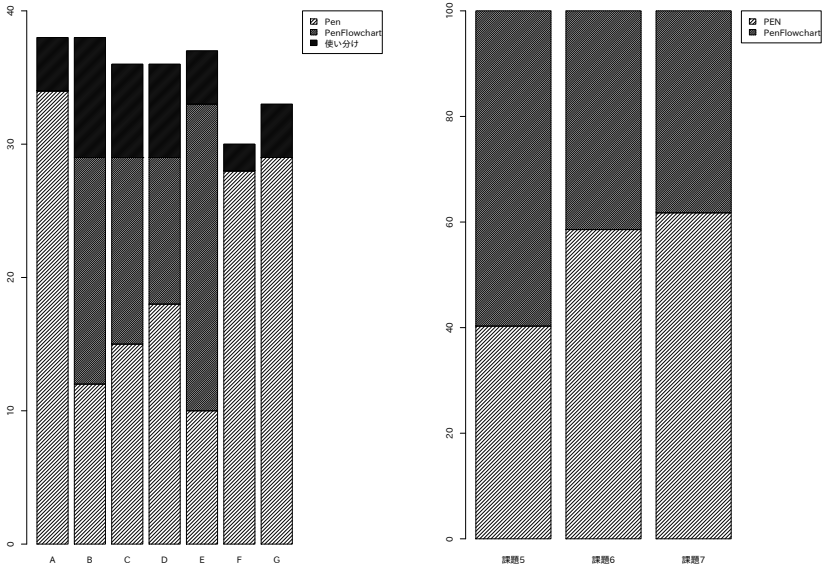
PEN	プログラムをキーボードで文字入力できる	66
	入力支援ボタンが便利	120
	最初から使っていたから	84
	ブロック構造を壊して再入力するのがいや	23
PenFlowchart	マウスでプログラムが作れる	97
	フローチャートのまま入力できる	73
	値の編集でダイアログが開くのがいや	35
	実行画面が別なのがいや	36

PEN 派と PenFlowchart 派を比べると圧倒的に PEN 派が多いのだが，実はクラスによって分布が大きく異なっている．クラス別に集計すると図 4(a) の通りで，A，F，G 組には PenFlowchart 派がいなくて，他のクラスはだいたい半々である．これは授業の行なわれる曜日の違いによるものだと考えている．PenFlowchart は導入当初こまごまとバージョンアップを繰り返しており，そのリリースがたいてい水曜日であった．A，G 組の授業は翌日の木曜日であったため，筆者の説明がこなれていなかったということではないだろうか．また，F 組は初めの方の授業で何度かマシントラブルがあったため，生徒が新しいものに対して慎重になったと考えられる．

アンケートとは別に，PenFlowchart を導入してからは，課題提出の際にどちらで作業したかを併記させてきた．図 4(b) のように，最初は PenFlowchart の方が多かったのだが，回を重ねると PEN の方が多くなっている．机間巡視で観察した印象でも，何人かは PenFlowchart から PEN に戻っていたし，大まかな流れだけを PenFlowchart で作ってから細かい作業を PEN で行なっている生徒もいた．そもそも PenFlowchart は PEN への橋渡しとして作ったものであり，制御のブロック構造を理解できるようになれば PEN のテキストエディタで作業した方が効率がいい．生徒たちがそれを理解して次第に PEN に移行することが作者としての望みであるが，その様子を把握するに足る観察はできなかった．特に今年度は数回目の授業で唐突に PenFlowchart を紹介し，口頭で操作説明をしただけなので，生徒の中には PenFlowchart にはまったく手をださず，最初から使っていてテキストに

⋆1 東京農工大学の平成 18 年度前期入試の問題 1 にもあったし，8) では同じページに両方の記述がある．

も説明のある PEN をそのまま使い続ける者も一定数いた．その点では生徒への調査を行なうのは，今年度の授業では無理があったといえる．



(a) クラスごとの分布（人）　(b) 課題ごとの推移（%）

図 4　PEN 派，PenFlowchart 派の分布

どちら派であるかということが期末テストの点数に影響するかということについても調べてはみたが，有意な差は得られなかった（全体，設問別ともに）.「課題によって使い分ける」とした集団は両方の利点を理解して使い分けているのだから成績が良いだろうと予想していたのだが，若干平均点は高いものの有意差があると判断するには至らなかった．もっとも，前述したような不揃いが起きてしまっているのだから，統計的な処理をおこなっても有益な結果は得られないと考えられる．

6. 今後の開発方針

6.1 PenFlowchart が実装していないもの

PEN には実装されているが，PenFlowchart には実装されていないものがいくつかある．たとえば

- `else-if` の構文「そうでなくもし」
- 繰り返し文の脱出「繰り返しを抜ける」
- `switch` の構文「〜の値に応じて〜のいずれかを実行する」
- ファイル I/O
- 手続き・関数

などがそうである[⋆1]．手続き・関数や構文に関するものはフローチャートでの表現方法が難しいと考えており，実装していない．ファイル I/O についてはグラフィック命令と同じように実装することはできるが，まだ実装していない．

6.2 今後の改良に関する考え

ループの始端と終端の対応を見えやすくすることが当面の課題である．ループを薄い色の枠で囲む方法を考えているが，もっと適切な方法があれば教示いただきたい．対応する始端と終端を同じ色で塗るという方法なら簡単に対応できるのだが，色が判別できない生徒がいることや，逆に色に意味があるような誤解をしてしまう可能性があることを考えると，この表現はできれば避けたい．

制御の構文のうち,「繰り返しを抜ける」は実装しておきたい．他の分岐に関する表現は PAD なら自然に表現できるが，フローチャートだと単純な Yes/No の分岐との使い分けが難しくなりそうなので，当面実装は考えない．手続き・関数はそれほど難しくないかもしれないが，前述したように PenFlowchart は制御構造を理解するまで過渡的に使うものと考えているので，プロシージャが必要なほど複雑なものまで作れるようにする必要があるかどうかは今後検討したい．

ファイル I/O については，グラフィック命令を実装した以上これを排除する理由はないので，近いうちに実装する考えでいる．

値の編集を，いちいちダイアログを開かずにフローチャートの部品上でできるようにならないかという要望もあるが，画面上の座標計算が面倒に思われるので躊躇している．特にグ

⋆1 「〜の値に応じて」や手続き・関数については 2) には記述がないが，サンプルプログラムでは使われている．

ラフィック命令は引数がたくさんあるので，ダイアログを出した方が各値の意味がわかりやすいと考えられる．

PEN のプログラムを読み込むこと，あるいは実行画面でプログラムを編集すること（生徒からも要望がある）を実装するためには，プログラムの構文を処理する必要があり，これは筆者の力不足のため簡単には実装できそうにない．

本プログラムは夏休みの終わりに思いついて二学期の授業に間に合わせるために大急ぎで作成した．大慌てで設計・実装をしたため，後になって場当たり的に修正を加えた部分があり，既にプログラムの見通しが悪くなっているところがある．これを全体的に見直すことが，本当は喫緊の課題ではある．

6.3 PEN への要望

筆者は PEN の開発には関わっていないが，生徒からの要望で本報告に関わりがあると考えられるものがあったので追記する．

その要望というのは「構文エラーを引き起こすような削除ができないように PEN のエディタを改造できないか」というものであった．しかしこれを実現するのはきわめて難しいであろうし，テキストエディタの持つ自由度を制限することにもなってしまう．たとえば Squeak や Scratch，Alice などのようにタイルを組み合わせる方法でプログラムを作れば構文エラーは不可能になる．しかし生徒アンケートの自由記述にも，PEN の良さとして「自由度が高い」「テキストで入力するのでカッコイイ」「マウスで作るのは面倒」といった意見をあげているものがあり，テキストエディタの形式をあまり大きく変えてしまうとこのような良さが失われてしまう．

エラーを出せないのがいいのか，エラーが出たら直せばいいとするのか（気づいて直すことも経験だと考える）は目的によって異なる．授業中エラーで手が止まっている生徒の多くは「エラーが何行目って書いてある？ その行に何か間違いがあるはずだからよく見てごらん[⋆1]」と助言するまでエラーメッセージを読もうとしない．これはプログラミングに限ったことではなく，周囲の同僚たちの中にもコンピュータで何らかのトラブルが起きたときにエラーメッセージを読もうとしない人は多いと感じている．だとしたらこのようにデバグをすることによって，コンピュータが出すメッセージは一見難しそうに見えるが読んでみれば有益な情報が含まれていること，その情報を元に修正すれば適切な結果が得られることを実感するのは有効な体験であると考える．

⋆1 いわゆる全角・半角の違いによるエラーも多いのだが，生徒の中にはそれが違っていても目で見てわからない（上の行が半角で次の行は全角だったとして，それを教師が指さして指摘しても違いがわからないと言う）者がいる．

7. おわりに

授業で用いるソフトウェアは，改変が可能であることが望ましいと考えている．たとえば PEN は GPL で配布されており，だから PenFlowchart にまるごと取り込んで（プログラムを編集不可能にしたり，メニューや入力支援ボタンを消したりするといった改変も行ないつつ），直接実行できるものを作ることができた．PenFlowchart もその意思を受けて GPL で配布しており，そのためか，これを Android に移植しようという動きがあると聞いている．

授業で扱うソフトウェアをカスタマイズしたいという欲求は多くの教員が持っているのではないか．実際筆者も PEN の入力支援ボタンは本校で行なう授業用にカスタマイズして使っている．そういったことができるようなスキルをすべての教員が持たなくてはいけないとは考えていないが，望めば自分の授業に合わせた改変ができるような状況であってほしい．

参考文献

1) 中村亮太，山本武生，西田知博，松浦敏雄：初学者向けプログラミング教育環境 PEN，（オンライン），入手先⟨http://www.media.osaka-cu.ac.jp/PEN/⟩（参照 2011-12-26）．
2) : xDNCL 言語マニュアル (2008).（PEN 1.19.6 付属 `xDNCL-Language-Manual.pdf`）．
3) 西田知博：PEN: 大学入試センター用言語を用いたプログラミング教育，情報処理学会教育用プログラミング言語に関するワークショップ 2006 報告集，pp.62–67 (2006).
4) 大島友明：Java Swing によるドラッグアンドドロップの実装，MY–NOTEBOOK（オンライン），入手先⟨http://osima.jp/java/java-swing-dnd/⟩（参照 2012-01-04）．
5) 三浦元喜，杉原太郎，　國藤進：Anchor Garden:オブジェクト指向言語におけるデータ構造の基本を理解するためのワークベンチ，情報教育シンポジウム論文集，Vol.2008, No.6, pp.55–62 (2008).
6) 日本規格協会：JIS X 0121:1986 情報処理用流れ図・プログラム網図・システム資源図記号 (1986).
7) 　山本恒：フローチャートによるアルゴリズムの視覚化と検証システムの開発，日本情報科教育学会第 4 回全国大会講演論文集，pp.102–103 (2011).
8) 水越敏行，　村井純（編）：新・情報 B，日本文教出版 (2011).