コメントイン：コメントを先に書くことによる新形態の API リファレンス

薄羽 大樹* 宮下 芳明†

概要．本稿では，「使うべき API の仕様が分からないユーザ」を対象とし，コメントを先に書くことによる新形態の API リファレンス「コメントイン」を提案する．ユーザがコメントを入力すると，システムはそのコメントに関連して「そのプログラミング言語でできること」を推薦する．候補一覧から「やりたかったこと」をプレビュー・選択することを基本操作とし，ユーザはプログラムを記述していく．これにより，ユーザは仕様が分からない命令をも簡単に自身のプログラムに取り入れることが可能になる．また，評価実験により，提案システムを用いることで，仕様が分からない命令の追加，プログラムの修正が可能であることも示した．

1 はじめに

本稿では「使うべき API の仕様が分からないユーザ」を対象としたプログラミング支援を提案する．例えば，if 文や，四則演算といった基本文法を身につけてはいるが，自分のやりたいことに対応する関数の仕様が分からないユーザである．Processing には慣れているが Java に挑戦したい，Processing に慣れているが Arduino でフィジカルコンピューティングを行いたい， Ruby の経験はあるが音楽プログラミング環境 Sonic Pi はまだである場合などである．こうした場合，ユーザは通常，Web 上を検索することで API リファレンスからその API の仕様やサンプルコードを手に入れることができる．そして内容の理解に努め，サンプルコードをもとに目的に沿った改変と流用を行う．しかし，Web 上を検索するためには自身の思考を適切な検索クエリにしなければ情報は得られず，たとえサンプルコードが見つかったとしてもそれが自分の望むパラメタを用いた例である保証はなく，流用できるほどに消化できないこともある．本稿で提案する新形態の API リファレンス「コメントイン」は，こうした問題の解決を目指すものである．

提案システムでは，まずユーザが「後にコメントとなる自然言語」(従来のコメントと区別するため以下キーコメントと呼ぶ）を入力すると，システムはキーコメントに関連して「そのプログラミング言語でできること」の一覧を推薦する．なお，書かれているプログラムおよびコメントを鑑みてその推薦順序を決定している．ユーザがそれらをマウスオーバーでプレビューし，「やりたかったこと」を選択（場合によってはこの時点でパラメタを入力）すると，該当するプログラミング言語の命令が記述され，キーコメントはコメントとなる．また，記述したプログラムをあとから修正する際，コメント部分を変更することで候補一覧を再び呼び出し，プログラムを変更することもできる．

「命令を推薦する」システムでユーザが正しい命令を選択するためには，その命令が既知であるか，命令名などから容易に実行結果を推測できる必要があるが，「そのプログラミング言語でできること」の一覧を推薦する本手法では，仕様が分からない命令をも簡単に利用できると考えられる．

提案システムの設計思想は，安村の提唱する Programming 2.0[1]に符合するところがある．安村は，ユーザ指向のプログラミングのための 8 つの条件を挙げており，提案システムは，それらの条件の中で，自然言語によるキーコメントの記述とそれに関連した候補一覧の表示によって，「インタラクティブに動く」「曖昧さを許す」「検索機能をもつ」「常識を持ち，ある程度の推論を行う」「分からないときは，ユーザ（プログラマ）に聞く」という点を達成したものであると考えている．

本手法は，コメントを先に書かせることを強制させるものではなく，「コメントを先に書いた場合」にその行動を支援する仕組みを通常のプログラミング環境に適用したものである．自然言語で記述できるよう言語自体を変えたわけでもなく，好意的解釈のような機構をコンパイラに搭載したわけでもない．よって，ユーザのプログラミングスタイルの自由度を制限することはない．

2 提案システム: コメントイン

提案システムは，Web アプリケーションとして実装し，p5.js でプログラミングを行うインタフェースに加え，「候補一覧」と「実行・プレビュー画面」で構成される（それぞれ 図 1 赤枠）．なお，本章で紹

* 明治大学総合数理学部先端メディアサイエンス学科
† 明治大学

介しているのは，後述の評価実験の参加者によるものではなく，著者らによる使用例である．

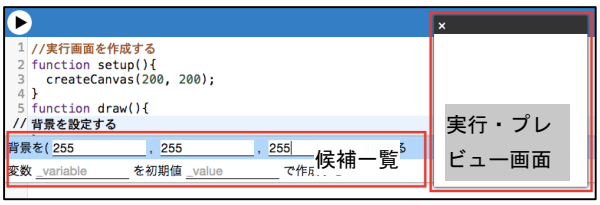

図 1. システム外観

2.1 基本仕様

現在行に入力されている文字列はキーコメントとして扱われ，エディタ中の行番号は，コメントを示すダブルスラッシュに変更される（図 2）．

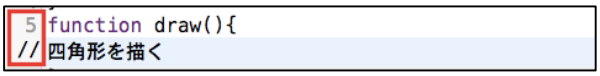

図 2. キーコメント入力時の画面

キーコメントは，API における関数名でもよいが，したいことに関連する自然言語で良い．システムは，キーコメントに関連して「そのプログラミング言語ができること」の一覧を推薦する．一覧の順序はキーコメントの関連度により決定している．

候補中のアンダーラインの箇所はテキスト入力欄になっており，任意にパラメタを設定できる．また，プレースホルダはパラメタの名前を示している（図 1 候補一覧）．候補にマウスオーバーすると，その候補の命令が書き加えられた実行結果をプレビューする．これにより，ユーザは実行結果を確認しながらパラメタの変更，候補の選択が可能である（図 3 赤枠）．候補を選択すると，パラメタの全角英数字が半角に置換され，現在行に命令が記述される．命令の記述後，入力されていたキーコメントはコメントとして現在行に追記され，キャレットは定められた行数分，自動的に下に移動する（図 4 赤枠）．また，命令が複数行にわたる場合，現在行ではなく，命令より 1 行上にコメントが追記される．

図 3. プレビュー機能（右ウィンドウ）

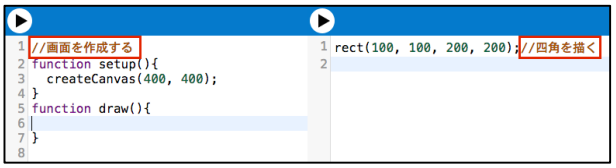

図 4. 命令によるコメントの位置

2.2 パラメタの推薦

提案システムでは，パラメタがあらかじめ設定された状態で，候補一覧に表示される場合があるのでそれについて説明する．

2.2.1 関数に基づくパラメタ推薦

プログラム中，現在行より上の記述に基づきパラメタの推薦が行われる場合がある．候補一覧の関数と現在行より上の関数のパラメタが同名であれば，それも鑑みた推薦がなされる（図 5 赤枠）．

図 5. 関数に基づくパラメタ推薦

ユーザが，同じ名前のパラメタを複数回使用している場合，同様の名前が付くパラメタは全て候補（図 6 赤枠）となり，もっとも現在行に近く，出現頻度が高いものが推薦される（図 6）．

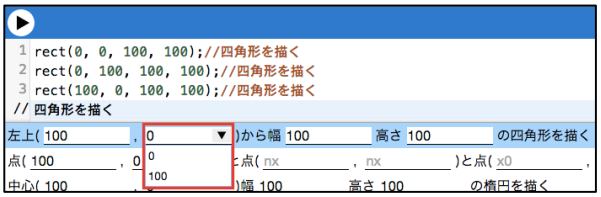

図 6. パラメタ候補の表示

2.2.2 コメントに基づくパラメタ推薦

変数宣言のコメントはその用途を示しており，ユーザはそのコメントに従ってパラメタの設定を行うはずである．そのため，例えば四角形を描く命令には，x 座標，y 座標，幅，高さが必要であり，各パラメタはコメントに従って推薦される（図 7）．

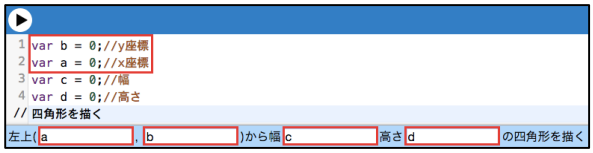

図 7. コメントに基づくパラメタ推薦

2.2.3 キーコメントによる推薦

例えば，キーコメントが『「何か」を赤くする』の場合，「何か」を設定する命令の RGB 値が 255，0，0 に設定されてほしい．また，時間や「マウスの座標に」といった場合にも，同様のことが考えられる．背景を設定する命令には，RGB 値の設定が必要であるため，提案システムではキーコメントに対応して，パラメタが推薦される（図 8）．

図 8. キーコメントに対応したパラメタ推薦例

2.3 プログラム内コメントの変更による修正

ユーザが，キーコメントを「背景を赤くする」とした場合，記述される命令は以下のとおりである．

```
background(255, 0, 0);//背景を赤くする
```

背景を赤ではなく，青に修正したい場合，キーコメント以外を削除することで，候補一覧を再び表示できる．表示後，キーコメント中の「赤」を「青」に変更し，再び候補一覧から選択することで，プログラムも修正される．これにより，ユーザは，青の RGB 値や，背景を設定する命令の仕様を知らずともプログラムの修正が可能である（図 9）．

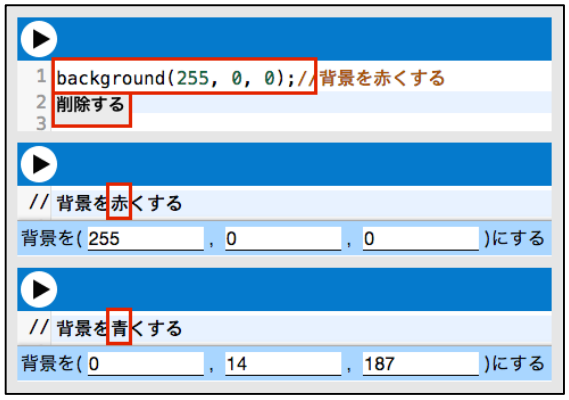

図 9. プログラム内コメントの変更によるプログラムの修正

2.4 実装

2.4.1 キーコメントの処理

```
"四(角 | 辺)(形)?", "(図)?形", "正方形", "rect", “長方形”
```

例えば，rect 関数には上記の正規表現（それぞれを以下トリガと呼ぶ）が用意されている．トリガの選び方に関しては，6 章で後述する．入力されたキーコメントはトリガとマッチをとり，1 回以上マッチした場合，候補一覧にその命令の内容が表示される．キーコメントと命令名が一致した場合にも問題なく動作できる．また，2.1 節で述べた関連度は，よりマッチしたものを推薦したいため，マッチした回数で定義している．

2.4.2 プレビュー機能

提案システムは，Web アプリケーションとなっており，プレビュー機能はインラインフレームを用いて実現している．そのため，エラーを含むプログラムや，無限ループを含むプログラムを書いた場合，プレビューを行うことができない．

3 想定されるプログラム例

本章では，提案システムを用いながら作成できるプログラムを紹介する．著者らの作成した想定例であり，後述の評価実験によるものではない．

3.1 アラームアプリの作成

指定した時間に YouTube の動画ページを開き，その動画の音をアラーム音にするアラームアプリを提案システムで作成する場合，以下の手順になる．

まず，実行画面を作成する．「実行画面を作成する」と入力し，幅と高さを 100 に設定する．キャレットは 6 行目に自動的に移動する．次に，指定時刻に動画ページを開きたいことから，時間による条件分岐を行う．今回，時間は 7 時 19 分とする．「7 時 19 分になったら」と入力すると，パラメタが推薦され，候補一覧が表示される．1 番目を選択し，選択後，キャレットは 8 行目に自動的に移動する（図 10）．

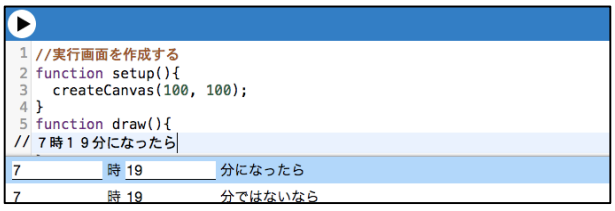

図 10. 時間による条件分岐パート

最後に，動画ページを開くため，「ページを開く」と入力すると，候補一覧が表示されるので，パラメタを動画の URL に設定する（図 11）．完成したア

ラームアプリは図 12 のとおりである．

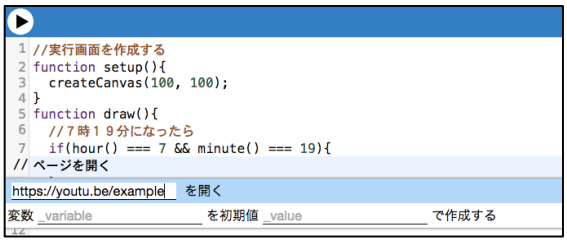

図 11.「ページを開く」の候補一覧とパラメタの設定パート

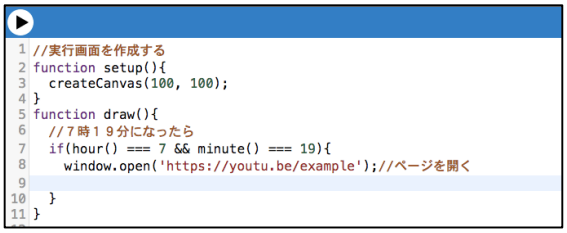

図 12. 完成したアラームアプリ

3.2 ペイントアプリの作成

マウスが押されている間，ランダムな色で塗りつぶされた円を描くペイントアプリを提案システムで作成する場合，以下の手順になる．

3.1 節と同様，実行画面を作成する．次に，「マウスが押されている間」とし，候補一覧が表示される（図 13）．1 番目を選択し，選択後，キャレットは 8 行目に自動的に移動する．

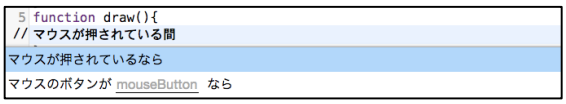

図 13. マウスによる条件分岐パート

さらに，「ランダムな色で塗りつぶす」と入力し，塗りつぶす命令が記述される．最後に，「マウスの座標に円を描く」と入力すると図 14 のようになる．2.2.3 項のとおり，マウスの座標を示す変数が，それぞれのパラメタに推薦される．候補一覧の 1 番目を選択し，プログラムは完成する．

図 14. ペイント部分の設定パート

4 評価実験

提案システムの有用性を確認するため，高校生 9 名による評価実験を行った．参加者のうち 3 名は授業などでプログラミングに触れたことがあり，それ以外はプログラミング未経験者であった．後者は本稿で設定する対象ユーザと異なるが，実験前にインストラクションと提案システムのデモンストレーションによって，プログラミングの基礎的な考え方を教えたうえで，本稿で支援対象としているユーザに近づけて実験を行った．

まず，提案システムを使わずにプログラムを記述してみせた．参加者にも同様にプログラムを書かせ，提案システムなしではプログラムを完成できないことを確認した．その後，提案システムのインストラクション，デモンストレーションを行った．そして，参加者にも提案システムを用いて，図 15 を左から順に 3 つのプログラムを書かせた．実験中，完成図は常に提示するが，命令や，追加されるコードについての説明は行わないこととした．

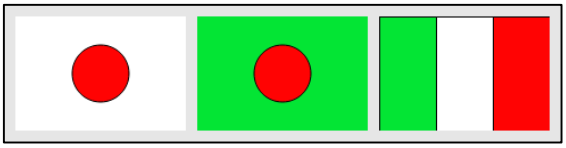

図 15. 実験で記述させる 3 つのプログラムの実行結果

4.1 目的

以下の仮定について検証を行う．提案システムを用いることで，1) 未経験の言語でも記述可能か，2) 命令の仕様が分からなくともプログラムの変更が可能か，3）命令の存在を知らなくても扱えるか．

1 番目のプログラムから，2 番目のプログラムを書くためには背景を白から緑に変える必要がある．提案システムを用いない場合，背景の色を設定する命令や，その命令が持つ引数の意味，RGB 値，もしくはカラーコードを学習済みでなければならない．提案システムでは，2.2 節で述べたとおり，上記の知識は必要なく，コメントを変更することでプログラムの修正が可能である．これにより，2）が達成できると考えられる．また，3 番目のプログラムは，手本で見せる円を描く命令ではなく，四角形を描く命令を知っている必要がある．しかし，参加者は「長方形を描く」や「四角形を描く」といったキーコメントを入力することで，四角形を描く命令を得られる．そのため，提案システムを用いた場合には 3）も達成できるはずと考えられる．

4.2 結果

提案システムなしでは誰もプログラミングを行えなかったにもかかわらず，参加者 9 人中 9 人がプログラムを 3 つとも完成させることができた．実験後の感想として，「楽しかった」や，「これならプログラミングをやってみたい」といった声を得ることが

でき，参加者のプログラミングに対するモチベーションを高めることもできたのではないかと考えられる．命令の追加後，キャレットが自動で移動するため，実験中，参加者が，次はどこから書けばいいか迷うことはなかった．そして，マウスオーバー時のプレビューにより，参加者が間違った候補を選択することもなかった．

本実験においては， 参加者が「function」や「実行画面を作成する」といったプログラミング特有の概念に戸惑うことはなかったが，これは提案システムによる貢献ではなく，実験時のインストラクション，デモンストレーションに起因するものと考えられる．「実行画面を作成」するというフレーズは，著者らがシステムのインストラクション時に使用している．そのインストラクションによって参加者が自ら学習したと考えられる．

本稿の対象ユーザではないが，もし提案システムを完全なプログラミング未経験者にインストラクションなしで使用させた場合，おそらく実行画面の作成から戸惑うのではないかと考えられる．「完全なプログラム未経験者」まで支援対象のユーザ層を広げるのであれば，起動時から実行画面が作成可能なキーコメントを入力するなどの対応が必要である．

また，評価実験で示したのは，「こちらが設定したプログラムにおいて」提案システムが有効であるかのため，ユーザが自由にプログラミングを行った場合にも正しく動作するか検証する必要と考えられる．

5 関連研究

増井の提案する展開ヘルプ[2]では既存のヘルプシステムと違い，ユーザの入力したキーワードに対して，やり方や必要なコマンドではなくシステム実行可能な機能を提示し，ユーザはそれを選択することで実行可能である．プログラミング支援に関する研究でないものの，展開ヘルプは，ユーザが理解しやすい内容を提示する点で本稿の提案システムに非常に近いものであると考えられる．また，Microsoft on{x}[3]では，文を変更することで実行可能なスクリプトを作成できる．

キーワードからプログラムの記述を可能にしている研究は多くある．Little らは検索クエリから命令の記述[4]，青島らはコメントから Java プログラムを生成可能にした[5]. Brandt らのシステム[6]では，入力されたキーワードから Web 上のサンプルコードと説明文を検索し，その内容をユーザのプログラムに追加できる．また，入力キーワード，Web ページのリンク，検索した日時はコメントとして記録される．既存研究と提案システムでは，キーワードからプログラムを作成する点，コメントが追加される点で類似している．Wightman らのシステム[7]も，キーワードからプログラムの作成が可能であり，作成されるプログラムは変数を参照するが，入力されたキーワードはコメントとして残らない．これらの既存研究では，サンプルコードや，提示されるプログラムの理解が必要となっている．提案システムでは，キーコメントに対応した「そのプログラミング言語でできること」を提示・プレビュー表示するため仕様が分からない命令であっても記述可能である．

Omar らのシステム[8]では，既存のエディタに搭載されている自動補完と異なり，例えば，色を示す型を返す関数の場合には，カラーパレットを表示し，そこから選択することでプログラムの補完を可能としている．Omar らの手法は，提案システムにおいても有用であると考えられる．

Kato らが提案する Picode[9]では，プログラム内に画像を貼り付けることが可能である．人やロボットの姿勢といった，文字や記号では理解しづらい情報を分かりやすく示すことができる．提案システムにおいても，プレビュー表示や，グラフィカルなインタフェースによる支援を行っている．

また，提案システムでは，キーコメントとして自然言語を入力する．自然言語である日本語でプログラムの記述が可能な日本語プログラミング言語「なでしこ[10]」などがある．また，中橋らが提案する好意的解釈により，命令にスペルミスがあった場合にも，正しい命令に置換され，エラーなくプログラムの実行が可能になった[11]．好意的解釈を搭載した日本語プログラミング言語がもし将来生まれるなら，自然言語でプログラムを記述可能になるかもしれない．提案システムにおいては，自然言語でプログラムを記述するのではなく，書くのはコメントであり，当然，別物である．

1 章でも述べたとおり，本稿は，コメントを先に書かせることを強制させるものでは決してない．「コメントを先に書いた場合」にその行動を支援する仕組みを，通常のプログラミング環境に適用したものにすぎない．上で紹介した研究のように自然言語で記述できるよう言語自体を変えたわけでもなく，好意的解釈のような機構をコンパイラに搭載したわけでもない．つまり，通常のエディタにオンデマンドで支援する機能を付したにすぎず，ユーザのプログラミングスタイルの自由度を制限することはない．

6 議論

既存のエディタや，前章で述べた関連研究の多くは，API における関数名をトリガとし，自動補完を行うものであるといえる．自動補完機能の利点は，入力文字列に対して候補が表示されるため，うろ覚

えの関数であっても記述できることにある．提案システムでは， API における関数名に加え自然言語のキーワードをトリガにしている．つまり，提案システムは，API における関数名による自動補完を包含している関係にある．トリガとなる自然言語は，日本語化された API リファレンス[12]の説明文を参考にしている．図形を描く関数の場合，線の色と面の色を指定し，様々な色の図形を描くことが可能である．そのため，図形（関数）そのものに関連する単語，線に関連する単語と面に関連する単語（その関数に関連するパラメタ）をトリガしている．例えば, 説明文が「rect によって箱や枠のような図形（長方形）を表示する」の場合，「箱」，「枠」，「図形」，「長方形」をトリガとし,「表示」はトリガとしない．「background によってウィンドウの背景の色を決める」の場合,「背景」,「色」をトリガし,「決める」はトリガとしない．また，提案システムは API における関数名による自動補完を否定しているわけではなく，それを強化したものとして提案している．

複数の API において，トリガは重複する場合がある．複数のトリガにヒットした場合，ヒットした全ての API を提示する．提示された候補はプレビュー表示をしながら選ぶことができるため，ユーザは実行結果を確認でき，複数候補の場合でも適切な API を選択可能であると考えられる．トリガの選択は難しく，現状では人間の直感にたよっているという問題があり，今後の課題である．

7　まとめと展望

本稿では，コメントを先に書くことによる新形態の API リファレンス「コメントイン」の提案と実装，それを用いた評価実験について紹介した．提案システムでは，コメントを先に書き，そのコメントに関連する「そのプログラミング言語でできること」の一覧を推薦することで，仕様が分からない命令をも簡単に自身のプログラムに取り入れることが可能になった．また，評価実験により，提案システムを用いることで，仕様が分からない命令の追加，プログラムの修正が可能であることも示した．

提案システムでは，初歩的なプログラミング学習支援や，プログラミング体験に留まったものとなっている．そのため，ある程度の経験を積み，より高度なプログラミングをするためには既存の環境への移行が必要である．しかし，提案システムにおいて入力文字列に対する自然言語処理やデータベースを改良すれば，どのような入力文字列であってもプログラムに翻訳することが可能になると考えている．そうなった場合は，他の環境へ移行する必要はなくなると著者らは考えている．

提案システムを Web アプリケーションとして実装した意図は，誰もがこのプログラミング体験をできるようにするためである．本稿で議論している諸課題を克服したうえで，システムを公開できるよう改良していきたいと考えている．

謝辞

本研究は，JST，CREST の支援を受けた．

参考文献

[1] 安村通晃, Programming2.0：ユーザ指向のプログラミング, 情報処理学会夏のシンポジウム 2006, 2006.

[2] 増井俊之. 展開ヘルプ, インタラクション 2012 論文集, pp.89-96, 2012.

[3] Microsoft on{x}（2015/10/14 確認）
https://www.onx.ms/

[4] Greg Little, Robert C. Miller.
Keyword Programming in Java,
Automated software engineering, Vol.16, No. 1, pp.37-71, 2009.

[5] 青島大悟, 荒井大輔, 石川龍二, 采泰臣, 加藤友規, 野澤稔. 日本語コメント記述からのプログラム自動生成システム（2015/08/30 確認）
http://www.ipa.go.jp/files/000005756.pdf.

[6] Joel Brandt, Mira Dontcheva, Marcos Weskamp, Scott R. Klemmer,
Example-Centric Programming: Integrating Web Search into the Development Environment, CHI2010, pp.513-522, 2010.

[7] Doug Wightman, Zi Ye, Joel Brandt, Roel Vertegaal, SnipMatch: Using Source Code Context to Enhance Snippet Retrieval and Parameterization, UIST2012, pp.219-228, 2012.

[8] Cyrus Omar, YoungSeok Yoon, Thomas D. LaToza, Brad A. Myers. Active Code Completion, ICSE2012, pp.859-869, 2012.

[9] Jun Kato, Daisuke Sakamoto, Takeo Igarashi. Picode: Inline Photos Representing Posture Data in Source Code, CHI2013, pp.3097-3100, 2013.

[10] なでしこ（2015/08/30 確認）
http://nadesi.com/

[11] 中橋雅弘, 宮下芳明. HMMMML3 他人を意識したモチベーション向上を考えたプログラミング環境, インタラクション 2011, pp.511-514, 2011.

[12] ベン・フライ, ケイシー・リース,（中西泰人 監訳, 安藤幸央, 澤村正樹, 杉本達應 翻訳）, Processing:ビジュアルデザイナーとアーティストのためのプログラミング入門, p.27, ビー・エヌ・エヌ新社, 2015.