絵画的プログラミング

久保田 秀和　　西村 拓一
h.kubota@aist.go.jp　taku@ni.aist.go.jp
産業技術総合研究所

概要

プログラミングにおける装飾美とスケッチ感覚に着目した絵画的プログラミングの概念を提案する．また，Web アプリケーションとして実装した Crowkee システムについて述べる．

Picturesque Programming

Hidekazu Kubota　　Takuichi Nishimura
National Institute of Advanced Industorial Science and Technology (AIST)

Abstract

This paper describes about Picturesque Programming that pursues decorative beauty and sketchy approarch for programming. Crowkee system that realizes Picturesque Programming has been implemented as Web application.

1．はじめに

人にとってプログラミングとは計算機の動作を記述することであり，人が自分の意図や気持ちに沿って何かを表現するという点においては文学や絵画，歌や踊りに類する活動であるとも言える．Donald Knuth の文芸的プログラミング（Literate Programming）[Knuth84] はプログラミングを表現の観点から論じたものとして広く知られており，ここでは人にとってのプログラミングが計算機に対して指示する活動であるよりむしろ，計算機を用いて何をしたいのかについて人に対して伝える表現活動であると主張されている．このため文芸的プログラミングにおいて求められる態度とは，そこで実現される計算機の動作について説明することやその文体の美しさについて，まるで随筆家が筆を執るときのように注意を払うことであり，その結果として随筆とソースコードの混交した文書がプログラムファイルとして制作される．

文芸的プログラミングは Knuth のような文章を書くことを好む人物にとって馴染みやすい手法であると考えられる．文芸的プログラミングの概念を敷衍するならば，プログラミングは絵画に慣れ親しんだ者にとっても取り組みやすい表現活動となることが期待できる．

本研究では，個人がその表現としてのプログラミングを画家の態度をもって実現することのできる絵画的プログラミング（Picturesque Programming）の概念を提案し，その実装システムである Crowkee を示す．画家の態度とはプログラムの見た目の装飾的な美しさやアイデアを大雑把に描き出すスケッチに対して注意を払うことであり，その結果として絵画コンテンツとソースコードの混交した表現がプログラムとして制作される．絵画的プログラミングによって記述されたプログラムは観る者にとって必ずしも説明的な判りやすさがあるわけではない一方で，絵を鑑賞するときのような楽しさ，美しさ，あるいはその他の言語化の難しい感情をもたらす可能性がある．

2．絵画とプログラミング

本節では文芸的プログラミングに代表されるような，プログラミングに対する芸術的なアプローチについて解説し，その中での絵画的プログラミングの位置づけについて述べる．

2.1 文芸的プログラミング

文芸的プログラミングの概念はプログラミングに対する芸術的なアプローチのなかでも歴史的に古く，また複数の派生的な実装がある[Pieterse04]［林 87］という点で代表的であると言える．文芸的プログラミングの最初の実装はWEB［Knuth84］と呼ばれるシステムであり，組版用言語である TEX を用いた記述とプログラミング言語である PASCAL を用いた記述が混交したWEB プログラム（図 1 ）を執筆することによりプログラミングを行う．WEB プログラムの特徴は，プログラムの動作を解説する文章中の任意の位置に，プログラミング言語で書かれたソースコード断片を処理の順序とは独立した自由な順序で挿入できる仕組みを独自のマークアップ言語によって提供する点である．図 1 では解説文とその解説に関連するソースコードとが交互に現れている様子を見ることができる．記述の際には混交している解説文とソースコードは，後処理を実行することによって TEX 文書と正しい順序に変換された PASCAL のソースコードに分離され，それぞれの処理系によって実行可能となる．

ここで Knuth にとっての文芸的プログラミングは自身が開発した TEX の副産物（spin-off）として捉えられている点［Knuth84］［Shustek08］が興味深い．Knuth にとっては美しい組版済み文書を用いた表現活動とプログラミングとは切り離せない関係にあると想像される．一方，3 節で述べるように絵画的プログラミングは筆者が開発したレイアウト自由型の Web 組版システムである SaasBoard[久保田 08]の副産物であり，絵画的な表現活動と切り離せない関係にある．

2.2 随筆家と画家の態度

文芸的プログラミングにおいて求められる随筆家の態度に似た観点として，画家としての態度がある．Paul Graham は「ハッカーと画家」[Graham03]において，特にハッカーにとってのプログラミングは，画家が絵を描く行為に類似していると述べており，これは文芸的プログラミング

```
\font\ninerm=cmr9
\let\mc=\ninerm % medium caps
\def\WEB{{\tt WEB}}
\def\PASCAL{{\mc PASCAL}}
\def\[{\ifhmode\ \fi$[\mkern-2mu[$}
\def\]{$]\mkern-2mu]$\ }
    ⋮
\hyphenation{Dijk-stra}

@* Printing primes: An example of \WEB.
The following program is essentially the same
as Edsger Dijkstra's @^Dijkstra, Edsger@>
``first example of step-wise program
composition,'' found on pages 26--39
of his {\sl Notes on Structured
Programming},$^\Dijk$  but it has been
translated into the \WEB\ language. @.WEB@>

\[Double brackets will be used in what
follows to enclose comments relating to \WEB\
    ⋮
an informal top-level description.\]
```

```
@p @<Program to print the first thousand
prime numbers@>
```

```
@ This program has no input, because we want
to keep it rather simple.  The result of the
program will be to produce a list of the
first thousand prime numbers, and this list
will appear on the |output| file.

Since there is no input, we declare the value
|m=1000| as a compile-time constant. The
program itself is capable of generating the
first |m| prime numbers for any positive |m|,
as long as the computer's finite limitations
are not exceeded.
```

（中略）

```
@<Program to print...@>=
program print_primes(output);
const @!m=1000;
@<Other constants of the program@>@;
var @<Variables of the program@>@;
begin @<Print the first |m| prime numbers@>;
end.
```

図 1：WEB プログラムの例[Knuth84]
（実線枠内が組版用言語，点線枠内がプログラミング言語を用いて記述された箇所）

と以下の部分において共通している．

(a) 読み手となる人間を意識すること

Knuth はプログラムの動作を人間に対してうまく伝えるための説明的な記述と構成を重視している．Graham は画家が自分の作品を観る人の視点を意識しながら絵を描くという一面を採り上げて，ソースコードについてもそれを読む者の立場から判りやすくなるように書くべきであると述べている．

(b) 個人にとって自然な手順でプログラミングすること

表現活動において個人の力点の置き方やペース配分，試行錯誤の方法やひらめきを得るための習慣はそれぞれ異なるものである．Knuth は，人間が特定の順序（例えばコンパイラにとって正しい順序）を強制されることなく，心理的に正しい順序でプログラムを書ける点が WEB の長所であるとしている．もちろん個人差もある．例えば Knuth の流儀では，各章ごとに解説から始まり関

連するソースコード断片で終わる書き方が自然であるが,他の書き手は他の流儀を取ることが出来る.

Graham は絵を描き進めるうちに当初の計画が幾度となく変更されることに触れ,プログラミングも当初の仕様に完璧さを期待するよりも,作りながらの仕様変更を受け入れられるような書き方をするべきであると述べている. Graham にとってのそれは,動的型付けの言語を用いることであるという.

(c) 熱狂すること

Knuth の文芸的プログラミングに対する思い入れは“Enthusiastic reports”[Knuth84]あるいは“I love programming, but I really love literate programming.” [Shustek08] という言葉で自覚的に採り上げられている. Graham もプログラムの見た目の美しさに対する熱狂的な没頭について自覚的に述べている.ここではプログラミングにおける表現手法そのものへの没頭が両者のプログラミング人生を支えていることが伺える.

絵画的プログラミングも(a)のような読み手を意識した表現を生み出すが,絵画コンテンツとソースコードの混交表現は必ずしも説明的な判りやすさを備える必要はない. Graham の述べた画家の態度は比喩的であるが,絵画的プログラミングではより字義に近い意味において絵を描くことを考える. また, ここで個人にとって自然な手順とは,絵を中心として描き進めるアイデアノートの作成手順に似ている.まずはプログラムの見た目を絵としてスケッチし,そこにアイデアや説明, プログラムを文字として添え書きする. スケッチはプログラムによって動作することが確認され,必要に応じて修正を加えながら清書が進められてゆく.ここでもやはり熱狂的に取り組むことが欠かせないが,そこにあるのは随筆あるいはプログラミングそのものに対する熱狂であるよりむしろ, ざっとスケッチすることの楽しさ, 細密な絵を仕上げてゆくことへの快感であるだろう.

2.3 プログラミングの美的性質

Knuth は折に触れてプログラムの美についても述べている[Bond05] [Shustek08]. 例えば,実用性があまりないアートのためのアートであることもプログラムの美的性質の一つであり, One-line プログラムや Polyglots (複数の言語としてパース可能なソースコード)あるいは Perl で書かれた詩 (Perl Poetry[Wall00]) のような技術的な熟練によってもたらされるものがこれに相当すると考えられる. これらの技術美 (technical beauty) とでも言うべき性質に対して, Knuth や Graham は主にプログラムの読みやすさや品質に関与する機能美(functional beauty)について述べてきたと考えられる.一方,絵画的プログラミングが目指すものは,絵を描いたり鑑賞したりするときの楽しさや,そのほか言葉では表しがたい美しさを伴う装飾美 (decorative beauty) である.

3. 絵画的プログラミングを可能とするWeb アプリケーション

一般にプログラムの制作において絵が描かれるのはアイデアスケッチの段階である.スケッチにはウィンドウや表示オブジェクトが描き込まれ, 主要な機能が矢印や説明文で示される. これに加え, ビジネスユーザ向けでないプログラム,例えば個人のデスクトップを飾るガジェットやファミリー向けのプログラム,ゲームプログラムなどの場合は,最終的な画面表示も絵画的に飾りつけられる. また, 動きのあるオブジェクトやキャラクターが含まれることもある.絵を描く場面が多いほど,絵画的プログラミングを適用できる機会がある.

絵を描くこととプログラミングを同時に取り組める環境として, 筆者はこれまでに SaasBoard と呼ばれる Web 標準に基づいた組版システムを開発してきた[久保田 06][久保田 09].SaasBoard の目的は,画用紙に絵や文字を描き込むような感覚で Web ページを自由にレイアウト可能にすることである. SaasBoard はブラウザ上で動作する Web アプリケーションであり, 電子地図のようにズーム／スクロールできるページの上に,直接文字や絵, 矢印を書き込んだり, 写真や動画を貼り付けることができる.ページ上のコンテンツはサーバ側に保存し,サーバとブラウザ間の通信には Ajax 技術を用いている.

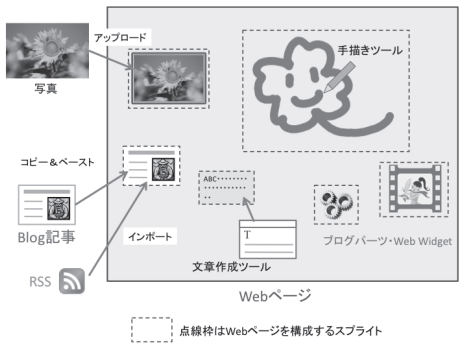

図２　スプライトを配置した Web ページ

SaasBoard はページやユーザの新規制作と管理のために SNS 機能を備える．SaasBoard で制作されたページは Web 標準（HTML4.01，CSS2.1，JavaScript）に沿って記述され，二次元空間上に複数配置されたスプライトから構成される（図 2）．ここでスプライトとは Web 標準に沿って記述された様々な情報をラッピングしたオブジェクトである．このため，SaasBoard のページ上では，絵や文字と同様に JavaScript を記述したスプライトを配置し，そのまま実行することも可能である．

この SaasBoard を絵画的プログラミングの概念に基づいて拡張することによって，Crowkee（クローキー）と呼ばれる Web アプリケーションを開発した[1]．Crowkee 上で記述されたプログラムの例を図 3 に示す．ここでは手描きの絵画コンテンツとソースコードが混交していることが判る．

３．１　スプライトモデル

はじめに Crowkee ページを構成するスプライトモデルについて説明する．スプライトモデルでは，sprite と呼ばれるノードを親とする入れ子構造のノードを用いてマイクロコンテンツを表現する（図 4(a)）．

・sprite ノード（必須）：ID と配置情報（絶対座標，幅，高さ，重ね合わせ順）を示す．プログラムはこの ID を用いることにより特定のスプライトに対する操作を行うことができる．

・region ノード（必須）：エンドユーザの操作に対する反応領域の形状を示す．

・plugin ノード（オプション）：RSS のインポートなど，スプライトの内容を動的に変化させるプラグインを示す．

・contents ノード（必須）：小さなひとまとまりのコンテンツを示す．

・info ノード（必須）：作者，更新時刻，URL，タグなど ID 以外のメタデータを示す．

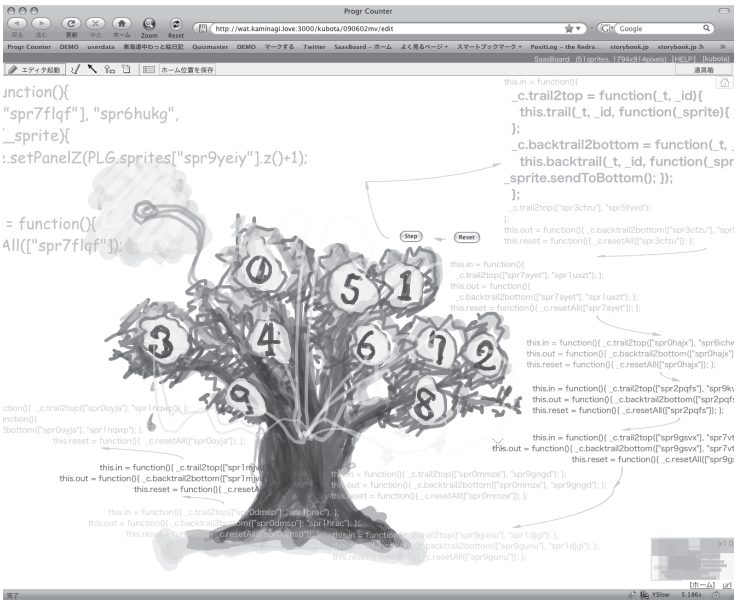

図３　Crowkee ページの例

(a) スプライトモデル　　(b) スプライトの記述例　　(c) スプライトの表示例

図 4：スプライトモデルとその実装

スプライトの各ノードは HTML4 における構造化のための要素 (div, span) を用いて実装する．スプライトの ID は要素の id 属性の値と対応し，ノードの名前は class 属性の値と対応する．図 4(b) に記述例を示す．配置は CSS2 に従って sprite ノードに絶対位置へ配置可能なスタイルを与え (position 属性の値が absolute)，top, left 等の属性値で絶対座標とサイズ，重ね合わせ順を指定する．contents の中身は HTML や CSS, JavaScript などブラウザが解釈可能な仕様に基づいて記述する．図 4(b) のように記述されたスプライトを Web ブラウザ上で表示した例を図 4(c) に示す．

３．２　Crowkee の概要

ユーザが Crowkee ページ上でコンテンツを作成するための典型的な手順を次の (1) から (5) に示す．

(1) ページ作成

SNS のメニューから，新たな Crowkee ページを作成する．

(2) スプライトの配置

ページ上にコンテンツを作成する基本的な手順は，スプライトを作成しそれを空間上の任意の位置に配置することの繰り返しである．各スプライトは，一般的な統合開発環境でよく見られるように，ツールボックスからドラッグ＆ドロップでページ上に配置するか，ページ上の任意の座標における右クリックメニューの選択により新たに制作・配置できる．簡単のため，頻繁に制作する文章スプライトは任意の座標をダブルクリックすることによって制作・配置できるものとする．

ここで新たに配置されたスプライトは，メディア毎に必要なエディタやツールを用いて編集できる．例えば画像スプライトを配置する場合は画像サイズを変更することが出来る．その他，文章スプライトを配置する場合はテキストエディタ，手書きのスプライトを配置する場合はペイントツールがページ上で起動する．また，配置したスプライトはドラッグ＆ドロップにより何度でも再配置したり，削除することができる．

(3) ソースコードの記述

ページ上でソースコードエディタを開き，JavaScript を用いてスプライトに対するプログラミングを行う．ソースコードも Code スプライトとしてページ上に制作・配置する．

ここでプログラムは絵的な表現を用いて記述，構成することもできる．スプライトとそれを操作するソースコードとは，矢印や手描きのアイコンを用いて対応づけることができる．ソースコード間の実行順序は矢印を用いて記述できる．また，絵的な表現から自動的にソースコードを生成することもできる (図 5)．

[1] http://crowkee.jp/

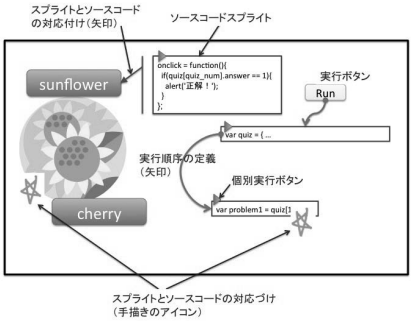

図 5　ソースコードの記述

(4) 実行とデバッグ

プログラムの実行ボタンを押して，コンテンツの動作を確認，デバッグを行う． プログラムは全体を実行することもできるし，ソースコードのスプライト毎に実行することもできる（図 5）．

Crowkee では Knuth の WEB のようにプログラムとして実行する際にソースコードを後処理によって分離する手法ではなく，自動的にソースコード部を解釈する手法をとる．

(5) 公開

デバッグを終えた後，公開する．Crowkee ではユーザ管理を行い，ページの閲覧権限を個別に設定することができる．

3.3　絵画的プログラミングの実装

Crowkee はソースコードの見た目や配置の自由度を高めるため，次のような特徴を持つ．

(a) インラインエディタ

ソースコードをカジュアルに編集できるようにするため，別のウィンドウやダイアログを開かずページ上に直接ソースコードを記述可能なインラインエディタを備えている．インラインエディタはユーザがソースコードを書き込みたい場所で開くことが可能であり，書き込み中と書き込み終了後のイメージがほぼ一致するように設計されている．

Crowkee ページにおけるプログラミングは，リッチテキストエディタで記述するだけでなく，絵的な表現を用いて記述，構成することもできる．以下では，Crowkee ページ上でのソースコードエディタと絵的な表現を用いたプログラミングモデルについて説明する．

(b) Code スプライト

JavaScript 言語と Code スプライトの関係を図 6 に示す．1 つの Code スプライトは JavaScript における 1 つの関数を表現しており，記述されたソースコードは，内部的には関数定義に埋め込まれる．Code スプライトは実行ボタンを押すことによって個別に実行可能であり，このとき，関数定義を元にインスタンスが作成され，インスタンスのメソッドを呼び出すことによって，元のソースコードが実行される．

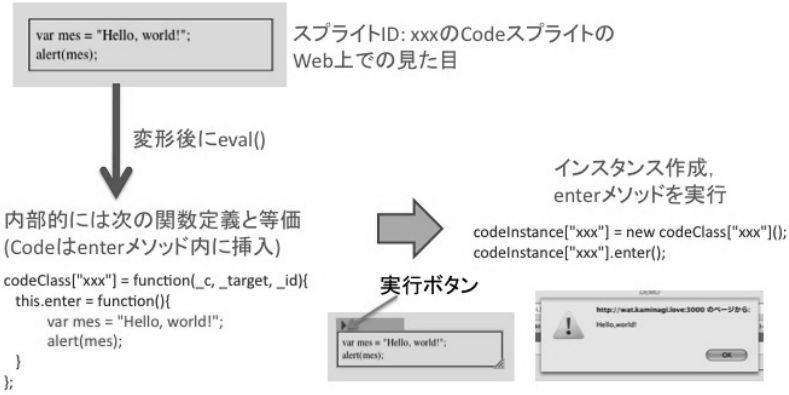

図 6　Code スプライトと JavaScript

・Run ボタン
押下すると，Code Chain のコードスプライトを矢印順に全て実行する．
コードスプライト内では enter()，exit()が順に実行される．

・Step ボタン
押下するごとに，Code Chain の Code スプライトを矢印順に１つずつ読み進めて実行する．
n ステップ目には，n-1 番目のコードスプライトの exit()が実行された後，
n 番目のコードスプライトの enter()が実行される．

・TickTack ボタン
押下すると，一定間隔を空けて Run ボタンの押下と同じ処理を自動的に繰り返す．

・TickTackStep ボタン
押下すると，一定間隔を空けて Step ボタンの押下と同じ処理を自動的に実行する．

・Reset ボタン
押下すると，Code Chain の Code スプライトを矢印順に全て実行する．
コードスプライト内では reset()が実行される．

図 7　Widget スプライト

(c) Widget スプライト

プログラムの実行制御はスプライト個別の実行ボタンのほか，Widget スプライト内のボタンを用いて行うこともできる（図 7）．

(d) Code Chain

プログラムの構造を視覚的に記述する Code Chain の設計と実装を行った．Code Chain とは，複数のコードスプライトを片矢印で接続することにより，プログラムの実行順序を表現するものである．矢印は多対多の接続を可能とした．

Code Chain の開始点は必ず Widget スプライトである．Widget を開始点として矢印で辿ることのできる範囲が Widget の実行スコープであり，スコープ情報を保持するコンテキストオブジェクト（_c）が伝搬される（図 8）．ここで１つの Code スプライトはで示すように複数の実行スコープに所属することができて，かつ，各実行スコープはそれぞれ非同期的に動作する．

Code Chain においては，Code スプライト内の 3 つのメソッド enter()，exit()，reset() が実行される可能性がある．Code スプライトは 3 つのうち最低 1 つのメソッドを持つ必要がある．各メソッドの実行タイミングは Widget 毎に異なっている（図 7）．なお，スケッチ感覚の試行錯誤を実現するために，ソースコードやコードチェインは実行中でも動的に変更可能とした．

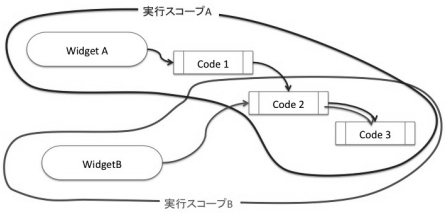

図 8　実行スコープ

(e) ターゲットアロー

ソースコード内では処理対象のスプライトを通常，ID を用いて指定するが，矢印（Arrow スプライト）を用いて視覚的に対応づけることも可能とした．矢印は Widget スプライトを開始点として，処理対象のスプライトを終了点とする．この矢印をターゲットアローと呼ぶ．Widget スプライトはコンテキストオブジェクトと同様に処理対象のスプライト情報を実行スコープ内へ伝搬する．

(f) バッジ

処理対象を指定する方法として，ターゲットアローよりも空間的な制約が少なく，かつ，絵的な表現を用いた手法として，あるスプライトを指す手描きのバッジの絵を一意に解釈可能な名前とする機能を追加した．

バッジを利用する手順は次の通りである．

(i) ユーザは操作対象としたいスプライトの上に重なるように，バッジのための手描きスプライトを描く．☆印，顔，手描きの文字など対象のスプライトに合った好みの絵を描くことが望ましい．写真シール機のような落描き感覚に近い．

(ii) ユーザは(i)の手描きスプライトをコンテキストメニューから複製する．システムは複製された手描きスプライトに対してオリジナル（複製元）のスプライト ID を記録する．

(iii) ユーザは複製された手描きスプライトをコードスプライトの上に重ねて置く.

(iv) ユーザがコードスプライトを実行すると,システムはコードを解釈する前に,上に重ねて置かれた手描きスプライトを検索する.オリジナルのスプライト ID をもつ手描きスプライトが見つかった場合,それはバッジとして解釈される.

(v) システムはバッジのオリジナルのスプライトの位置を検索し,その下に重ねて置かれたスプライトを操作対象(_target)として追加した後に,コードを解釈,実行する.

バッジを複製,移動することで,操作対象のスプライトをカジュアルかつ装飾的に指定することができる.

(g) 軌跡アニメーション

絵的な表現から自動的にソースコードを生成する手法として,軌跡アニメーションを開発した.軌跡アニメーションは描いた線に沿ってスプライトを移動させるソースコードを生成する機能である.ここで軌跡アニメーションを実現するために必要な機能は次の 4 つである.

(i) 移動させる対象を指定できること
(ii) 移動の軌跡を指定できること
(iii) 以上について複数の自由な線で描かれた絵として表現できること
(iV) 機能をコードスプライトとして利用できること

これらの機能を実現するため,ユーザがドローツールを用いて Drawing スプライトを作成する際の次の規約を導入した.

規約:はじめの 1 本目の描線は,軌跡として解釈される

ユーザはスプライトの軌跡を絵として表現する際に,線の色も太さも自由にしてどれだけ装飾的に描いても構わないが,そのうちのはじめの 1 本の線だけは実際の軌跡であることを意識して描く.

軌跡を描いた手描きスプライトに対して別途,コンテキストメニューから「コード生成」機能を呼ぶことによって,Code スプライトが生成される.このとき,手描きスプライトの 1 本目の線が抽出され,その開始点の直下にあるスプライトが移動させる対象として認識される.認識された内容はコードスプライトにソースコードとして埋め込まれる.たとえば,移動させる対象のスプライト ID が spr1abcd,手描きスプライトの ID が spr3dhea の場合は,spr1abcd を spr3dhea の指示する軌跡に沿って移動させる次のコードが生成される.

```
_c.trail(["spr1abcd"], "spr3dhea");
```

軌跡アニメーション関数としては,trail と backtrail を追加した.

_target.trail([target], spriteID)は,実行時に引数の spriteID で指定される手描きスプライトから 1 本目の線を抽出し,その点列に沿って[target]配列で指定されるスプライト(複数可)を移動アニメーションさせる.

_c.trail("view", spriteID)は,スプライトではなくページの表示位置(視点)を移動アニメーションさせる.

_c.backtrail([target], spriteID)は,この点列を逆順に解釈して[target]配列で指定されるスプライト(複数可)を移動アニメーションさせる._c.backtrail("view", spriteID)は,ページの表示位置について同様の処理を行う.

3.4 作成例

絵画的プログラミングを用いたコンテンツの作成例を示す.図 3 は数字をカウントアップするカウンタープログラムであり,Step ボタンを押す毎に 0 から 9 までの数字が順に中央へ移動して元の位置へ戻る動作を行う.ここで,Code スプライトのソースコードは絵に合わせた色や大きさによって飾り付けられている.画面上部の空に位置するソースコードは雲や空の色であり,中央の木の周辺に位置するソースコードは葉や幹の色としている.数字の移動は軌跡アニメーションによって,ページ上に描かれた自由な曲線に沿って行われる.また,図 9 は日本語の花名の英語での名前を当てるクイズゲームである.花の写真は Flickr から CC BY ライセンスの画像を検索して貼り付けている.写真の周りにはキャラクター画像が貼り付けられ,手描きのボタンの絵や吹き出

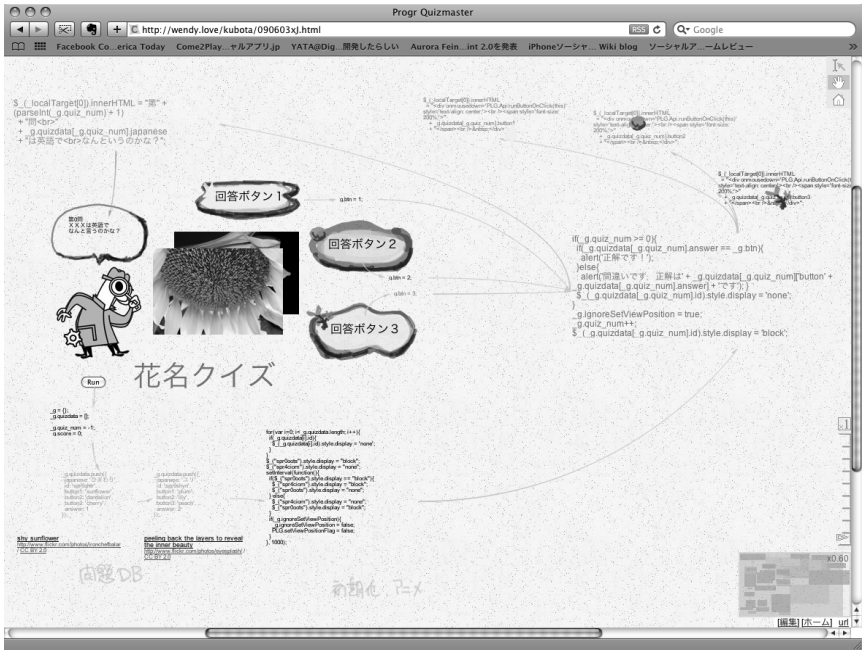

図 9　絵画的プログラム（花名クイズ）

しが描かれている．クイズの進行に伴って表示される画像や回答候補，吹き出しの中身が書き換えられる．ボタンのラベルはバッジによって Code スプライトと対応付けられており，現在の回答候補の英語名を表示するようプログラミングされている．

4．関連研究

Crowkee は独自のビジュアルプログラミング言語ではなく，全ての JavaScript の機能を利用可能な，絵画的プログラミングのための統合開発環境であるとも言える．

以下では，既存の Web サービスと Crowkee を比較することによって，Crowkee の特徴をより明らかなものとする．Web ベースの開発環境としては次のようなサービスがある．Uizard[2]は，Web 上で JavaScript を用いたグラフィカルユーザインタフェースを制作可能である． Bespin[3]は Web 上で JavaScript のソースコードとプロジェクトの編集が可能である．Atlas[4]は，Apple 社の開発環境である Xcode の Interface Builder を Web へ移植中である．Wonderfl[5]は，Web 上で Flash を開発し，ソースコード共有可能である．葵２[6]は，日本語プログラミング言語「なでしこ」の Web 版開発環境である．TIDE [7]は，さまざまな JavaScript コード断片の編集とテスト実行ができる．Crowkee はプログラムだけでなく，絵や文章などのコンテンツもその場で制作できる点において，これらの Web ベース開発環境と大きく異なる．Crowkee のユーザは，必ずしも始めにプログラミングから手を付ける必要はなく，日常的に Web コンテンツを作るなかで，気が向けばプログラミングをする，というスタイルを採ることもできる．また，Crowkee は，絵を描く者にとって自然な順序や様式でプログラミングできる絵画的プログラミングの可能な点が他と異なる．

その他，「Viscuit」[8]およびその Web サービス版である「うごうごブログ」[9]では，子供でも習得できる独自言語に基づくビジュアルプログラ

[2] http://uizard.org/
[3] http://labs.mozilla.com/projects/bespin/
[4] http://280atlas.com/
[5] http://wonderfl.kayac.com/
[6] http://aoikujira.com/wiki/aoi2/
[7] http://www.tide4javascript.com/
[8] http://www.viscuit.com/
[9] http://ugougoblog.com/

ミング環境を提供している．独自の言語を用いることによる利点もあるが，Crowkee では一般性をより重視し，充分に普及した汎用言語である JavaScript をユーザの用いる言語として採用する．このため Crowkee では完成されたコンテンツだけでなく，JavaScript ライブラリや実用的なツールも制作可能である．

また，Wonderfl や Viscuit で制作されるコンテンツが Flash ベースであるのに対して，Crowkee のコンテンツは HTML，CSS および JavaScript1.3 を用いた Web 標準ベースである．これらの Web 標準技術を用いたのは，既存の Web コンテンツの再利用や今後の持続性を重視するためである．Web 上にはこれまでに大量のコンテンツが蓄積されているが，Flash や Silverlight のような独自環境の上では，Web 標準に基づいて記述された既存のコンテンツをそのまま表示することが難しい．Crowkee では既存の HTML ソースをコピー＆ペーストで貼り付けて利用することも可能である．多くのブログパーツはこの方法で利用することができる．また，Greasemonkey[10]のようなこれまでの HTML 向けの技術をそのまま適用できる利点もある．

５．今後の課題と展望

今後はプログラミングをよりカジュアルに実現できるようにしたい．JavaScript の標準的なライブラリである jQuery（http://jquery.com/），prototype.js（http://www.prototypejs.org/）への対応や，初心者に人気のある Processing 言語への対応を予定している．

参考文献

[Bond05] Bond, Gregory. W.:Software as art. Communications of the ACM, Vol.48, No.8, pp.118-124, 2005.

[Graham03] Paul Graham: ハッカーと画家 コンピュータ時代の創造者たち，オーム社，2005.
（あるいは http://www.paulgraham.com/hp.html，邦訳 http://practical-scheme.net/trans/hp-j.html を参照）

[Knuth84] Knuth, Donald.E.: Literate Programming, The Computer Journal, Vol. 27, No.2, pp.97-111, 1984.

[Pieterse04] Pieterse, V., Kourie, D. G. and Boake, A.: A case for contemporary literate programming, In Proceedings of South African Institute for Computer Scientists and Information Technologists (SAICSIT) 2004, pp.2-9, 2004.

[Shustek08] Shustek, L en (editor): Interview Donald Knuth: A life's work interrupted, Communications of the ACM, Vol.51, No.8, pp.31-35, 2008.

[Wall00] Wall, L., Christiansen, T. and Orwant, J.: Programming Perl, Third Edition, O'REILLY, 2000.

[久保田 06] 久保田秀和: レイアウト・ベースド・コンピューティングへ向けて，人工知能学会主催第 7 回 AI 若手の集い（MYCOM2006）オンライン予稿集発表番号(5-3)，2006.

[久保田 09] 久保田秀和，前川博文，西村拓一：スプライトモデルを用いた絵地図型の Web コンテンツ構築システム，情報処理学会マルチメディア、分散、協調とモバイル DICOMO2009 シンポジウム，DS-13，2009.

[林 87] 林恒俊: WEB システムとその処理系，コンピュータソフトウェア，Vol.4，No.3，pp.249-256，1987.

[10] https://addons.mozilla.org/ja/firefox/addon/748