

プログラミングに対するモチベーションを向上させる新言語 HMMMML の開発

宮下 芳明

homei@homei.com

明治大学 理工学部 情報科学科

本稿では、プログラミングに対して後ろ向きになっている学生のモチベーションを高めることを目的として新言語をデザイン・実装した。提案言語 HMMMML がまず目指したのは `{ }` の完全廃止であり、ループや条件判定のためのタグ命令を用意した。提案システムではセミコロン抜けや宣言忘れなどのミスも許容して実行されるよう工夫している。また、音声合成による文字列の読み上げ、インターネットを利用した検索や翻訳、MIDI 音の出力、Google マップの表示といった処理が 1 命令で実行可能である。HMMMML エディタのインタフェースには無限ループボタンを用意し、これを利用することでカーソルキーによるキャラクタ操作やペイントプログラムも 1 行で記すことが出来る。さらに、C や Java や Javascript など他言語と混在・混同して書いても実行できるといった特徴を備えている。

HMMMML: A New Programming Language Designed to Motivate Students

HOMEI MIYASHITA

Department of Computer Science, Meiji University

In this paper I designed a new programming language “HMMMML (Homei Miyashita's Motivating Multilingual Markup Language)” to motivate students that not feel up to program. In HMMMML, we use `<for>` tags and `<if>` tags so that we do not need to use `{ }`. HMMMML allows lack of semicolons and declarations, and it is easy to make programs including text-to-speech reading, web search, MIDI output, and Google map display. By using infinite loop button on HMMMML Editor, the user can make interactive programs even in a single command-line string. Moreover, it is possible to mix other languages such as C, Java, and JavaScript in HMMMML.

1. はじめに

プログラミング能力は、いわば情報科学社会における「デッサン力」である。自分の構想を人に伝えるときに最も効果的なのは、そのプロトタイプを実装してデモンストレーションしてしまうことである。情報科学に携わる理系人間に関わらず、プログラミングは自己の表現能力を拡張する強力な手段であり、その力を手にすることは大きな喜びにつながるはずである。

本稿著者は情報科学科の教員として、学部 1 年生から博士前期課程 2 年生にわたるプログラミングの実習授業を担当している。言語は C, Java, Action Script (Flash), JavaScript, Processing¹⁾, HSP²⁾であり、学生たちの多様なニーズに対応させた指導を行

っている。高い意欲を持った学生たちは、こうした授業から多くを学びスキルを磨いているが、その一方で、プログラミングという行為に対して後ろ向きになっている学生たちも存在するのが現状である。

本稿で提案する言語 HMMMML は、こうした学生たちのモチベーションを高めるようなデザインを検討したものである。

1.1 モチベーション低下要因

学生たちのモチベーションを下げるひとつの大きな原因は、ソースコードの分量の多さではないかと考えられる。高度な処理を行いたい場合はもちろんのこと、基礎的な実習課題であってもそのソースコードは数十行にわたってしまう。プログラマにとって数十行は「短い」ものだが、プログラミングの初

心者にとっては、それでも長いものにとらえられているようである。

また行数だけでなく、その煩雑さもモチベーションを下げている要因なのではないかと考えられる。たとえば一般に「3回 Hello と書く」ためには、`int i; for (i = 0 ; i < 3; i++) { }` という書き方が一般的だが、「3回繰り返す」という意味を表すには些か大げさな印象があり、可読性も悪く、書く側としてもやや面倒でミスを誘う表現に思える。実際、情報科学科の学生たちですら、しばしばこの()内を正しく書けなかったり、変数 `i` を宣言し忘れることがある。

そもそも、プログラミング初心者が書くプログラムというのはなかなか動かないものである。変数の宣言忘れ、インクルードし忘れ、スペルミス、セミコロン抜け、`==`と`=`の混同、大文字小文字、`{ }`の不整合の間違ひは日常的である。不注意によるミスとはいえ、プログラムが動かずにエラーメッセージが出る体験を重ねてしまうと、何をしても怒られる子供のようなもので、次第にモチベーションが下がってしまうのは必然ではないだろうか。そしてもちろん、プログラムが暴走して応答しなくなるような経験は、さらに大きな低下要因になりうる。

典型的なケアレスミスのうち、`{ }`の不整合は多くのプログラム言語に内包される問題である。実習を行っている学生を観察すると、プログラムの最後に`}`がいくつも並んでいるソースコードがうまく動かないときに、勘に頼って`}`を増減させて帳尻を合わせる行動がよく見受けられる。考えてみると、多くのプログラム言語ではメイン文もコンストラクタもクラスもメソッドも`{ }`ではじまり`}`で終わる構造となっており、`if`文も`for`文も`while`文も大括弧を用いている。ソースコードにおいてはそれらが並列化・多重化し入り乱れているが、このような言語様式では、どの大括弧がどの大括弧に対応しているのかを見失ってしまうのも無理はない。

このように、モチベーションを下げる原因として、ソースコードの分量が多くなりがちである点、表現が煩雑になりがちである点、ちょっとしたミスでもエラーメッセージが出て動作しなくなる点、またプログラムが暴走して応答しなくなる点が考えられる。

1.2 モチベーション向上要因

次に、プログラミングに対するモチベーションを上げる要因について考察する。

おそらくかつては「自分が書いたソースコードの通りにコンピュータが動作する」ことだけで十分に興奮に値するものであったのかもしれない。著者が初めてプログラミングを行ったパーソナルコンピュータ SHARP MZ-80B は色表示すらできないものであったが、文字キャラクタによってグラフィック（いわゆるアスキーアート）を表示するだけで夢になれるものであった。

しかし、多様かつ高度な情報機器が身近に溢れている現代においては、指令したとおりに文字が表示されることからくる感動などはまずなく、"Hello World"の文字列から広大なプログラミングの「世界」を感じることは難しくなっている。むしろ文字列の表示どころか、「円を描く」「jpg ファイルを表示する」「ドレミの音を出す」「マウスやキーボードで操作できるようにする」ことすら当たり前のように感じられてしまう。にもかかわらず、既存のプログラミング教育ではこうした処理ですら十分に教えられていないのが現状である。

1.3 デザインポリシー

こうした議論から本稿では、モチベーションを向上させるための言語デザインを行うためには以下の5つの要件が必要なのではないかと考えた。

- (1) ソースコードの分量が少なくなること。たとえば、たった1行で充実したプログラミング体験が行えないだろうか？
- (2) 表現を単純化し、直観的なものにする。for文のような繰り返しはもっとシンプルな表現に変えられないだろうか？`{ }`に変わる表現はできないだろうか？
- (3) ちょっとしたミスくらいなら大目に見て動作してくれるようにできないだろうか？
- (4) プログラムが暴走して応答しなくなる状況を回避できないだろうか？
- (5) 上記を維持しつつ、グラフィックやマルチメ

ディア処理を包含し、インタラクティブリティを付与した「派手な」プログラムが簡単に実現できないだろうか？

本稿ではこれらの条件を優先するにあたり、他の事柄は犠牲になってもよいと考えた。例えば、処理速度は多少遅くなってもかまわない、汎用性が多少損なわれてもかまわない、他の言語への橋渡しについても考えるべきだが優先順位は下げる、とした。また、直感性を追及してグラフィカルな言語やインタフェースまでは検討せず、あくまでテキストを入力してコンピュータを動作させるプログラム言語の範疇でデザインを行うこととした。

言うまでもなく、将来的には長いソースコードに対する耐性も身につけなければならないし、複雑なプログラミング表現もできるようにならなければならないし、派手なものだけが善であるわけでもない。プログラミングを極めるなら、いずれは汎用的な言語で実装を行う技術が必要になってくる。しかし、そういった鍛錬が持続的に推進されるためには、まずプログラミングの可能性や面白さを十分に体験しておく必要があると考えた。

これらの要件を満たす言語として開発した HMMMML (Homei Miyashita's Motivating Multilingual Markup Language) は単純かつ直観的な文法表現を工夫したものである。エディタの「無限ループボタン」によってたった1行でも多様な表現が行える。セミコロン抜けや宣言忘れなどのミスも許容して実行される。{} を完全廃止し、ループや条件判定のためのタグ命令が用意される。動作は緩慢になるものの、プログラムの応答がなくなることなく、ESC キーによって常にプログラムは終了可能である。これらによって上記(1)(2)(3)(4)の要件を達成した上で、音声合成による文字列の読み上げ、インターネットを利用した検索や翻訳、MIDI音の出力、Google マップの表示といった、(5)の条件であった「派手」な処理が一命令で実行可能である。

また、次の優先事項であった他言語への橋渡しについては、C や Java や Javascript など他言語と混在・混同して書いても実行できるようにすることで実現した。

2. 提案言語 HMMMML

2.1 エディタ

提案言語 HMMMML はテキストファイルであるが、HMMMML エディタ(図1)上での編集を想定している。このエディタはテキストエディタとしての基本機能に加え、実行/停止と無限ループオン/オフを切り替えるボタンを持っている(キーボードショートカット F5, Ctrl+L でも使用可能)。

内部的にはソースを HSP に変換し、HSP コンパイラで実行している。変換前にタグの開閉チェックを行い、どちらかのタグが不足している場合にはその旨を表示する。

無限ループボタンは HMMMML エディタにおける大きな特徴であり、このボタン(トグルボタン)をオンにすると、エディタ内のコードを実行し続ける(後述する<while>タグで挟むのと実質的に同じ動作となる)。その際には自動でウェイト(ESC キーによるイベント)が入り、プログラムの応答がなくなることはない。ESC キーを押すか、実行ウィンドウの閉じるボタンをクリックすることでいつでも終了可能である。

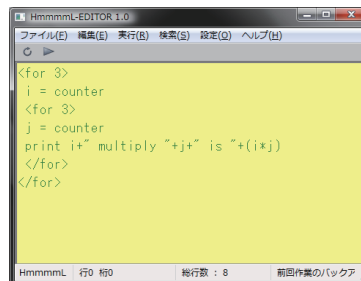


図1. HMMMML エディタ

詳細は3章で述べるが、この無限ループボタンによって、1行のプログラムでも多様かつ「派手な」プログラムを実践することが出来る。

2.2 基本文法

プログラムのソースコードは、HMMMML エディタで作成されたテキストファイルである。命令ごとに改行してあれば、行末にセミコロンがなくても動作する。// の後ろや/* */の間はコメント文として処理される。

変数は宣言せずに使用可能であり、整数型か実数型か文字列変数かは初回代入操作で判別している。すべての命令・変数において大文字小文字は区別しない。配列変数についても、実数の一次配列であれば宣言無しで使用可能であり、たとえば以下のようなプログラムが書ける。

```
a[100]=1 //宣言無しで配列を使用可
a[101]=2
b = a[100]+a[101]
print b //3 と表示される
```

三角関数 `cos()`, `sin()`, `tan()`, `atan()`, 平方根 `sqrt()`, 絶対値 `abs()`, 文字列変数の整数化 `int()`, 乱数 `random()` 等の関数は標準で用意されている。

特殊な変数としては、まず時間取得に関連する `now.second`, `now.minute`, `now.day` があり、常にその時点での時刻情報が代入されているものとして扱うことが出来る。また後述する `for` タグにおけるカウンタ変数 `counter`, マウスカーソルの現在位置 `mouse.x`, `mouse.y`, 自分の IP アドレスを返す `computer.ip`, `get.color` 命令で取得した色情報が格納される `color.red`, `color.green`, `color.blue` 等がある。

通常の言語に見られない変数として `cursor.x`, `cursor.y` がある。これは、カーソルキーによる座標を意味する変数で、プログラム起動時からのカーソルキーの打鍵累積値が格納されている。これにより、たとえばマウス操作によるプログラムがあったとき、そのソースコードにおける `mouse.x`, `mouse.y` を `cursor.x`, `cursor.y` に置換するだけで、キーボード操作によるプログラムに変更することが出来る。またこれらの変数は読み出し専用の変数ではなく変更することも可能で、たとえば `cursor.x = 0`, あるいは `mouse.x = 0` といった代入を行って位置をリセットすることが出来る。

コンパイル時には典型的なスペルミスや `==` と `=` の混同が起きても好意的に解釈するように多くの読み替え処理を行っている。

2.3 一般命令

HMMMML における一般命令は、英語における他動詞を使った（前置詞を用いない）命令文を基本

としてデザインし、その記法は

述語.目的語（パラメータ群）

となるようにした。例えば、(100,100)の座標を中心にして半径 50 の円を描画する命令は

```
draw.circle (100, 100, 50)
```

となる。

一般命令として他に描画に関する命令を列挙すると、四角を描く命令 `draw.box` (始点 `x`, 始点 `y`, 終点 `x`, 終点 `y`) , 線を描く `draw.line` (始点 `x`, 始点 `y`, 終点 `x`, 終点 `y`) がある。

`draw` を `fill` に置換した表現, `fill.circle` (中心 `x`, 中心 `y`, 半径 `r`) , `fill.box` (始点 `x`, 始点 `y`, 終点 `x`, 終点 `y`) を用いれば、円や四角を塗りつぶしたりすることが出来る。この際の描画色は `set.color` (赤輝度 `R`, 緑輝度 `G`, 青輝度 `B`) で指定する。また, `set.color` ("purple") のような色名指定も可能である。

点を描画する `fill.pixel` (座標 `x`, 座標 `y`), `draw.pixel` (座標 `x`, 座標 `y`) は同義の命令として用意されている。

`draw` 命令は一般的な言語における描画命令以上に拡張しており, `draw.screen` (ウィンドウ幅 `w`, ウィンドウ高さ `h`) によってウィンドウを生成できるほか, `draw.browser` (幅 `w`, 高さ `h`, "URL") を使うと, 画面内に指定サイズの IE コンポーネントを配置し, 指定した URL のウェブサイトを表示することが出来る。IE コンポーネントが表示される位置は `set.position` (座標 `x`, 座標 `y`) で指定する。

また `draw.map` (幅 `w`, 高さ `h`, 緯度 `la`, 経度 `lo`) を用いると, 指定した緯度と経度の付近の Google マップを表示することができる。

提案言語ではグラフィック関連だけでなく, サウンド関連でも命令を拡充している。特に MIDI 制御については一般の言語に比べて大幅に簡単に行えるような命令の実装を行った。MIDI デバイスのオープン/クローズを行う必要もなく, またノートオン/オフのメッセージを対で送信する必要もなく, 指定された高さの音を鳴らしたいときには

```
play.note(ノートナンバー n)
```

を実行しさえすればよい。また、音色の変更も `change.program` (音色ナンバー `v`) だけで行えるようにしている。

2.4 文字列命令

文字列変数を使う命令については、前節で示した一般命令の基本文法をさらに簡略化でき、目的語や `()` を使わなくてもよい表現としている。

```
print "hello"
```

画面への文字列表示を行う `print` は、`print a` のような表示だと変数の内容を表示し、`"` でくくられた場合はその文字列を表示する。連結して表記する場合には以下のように `+` を用いる。

```
print "a is" + a + "!"
```

`show` 命令は、`print` と全く同じ文法でダイアログを表示する命令である。`show "hello"` を実行すると、`hello` と記されたダイアログが現れ、OK ボタンをクリックすると閉じる。

`say` 命令は、`print` と同じ文法によって、Microsoft Speech API³⁾ を用いた音声合成でその内容を読み上げるものである。数値変数の場合は英語でその数値を読み上げる。

`translate` 命令は、`print` と同じ文法で、その内容を翻訳表示する命令である。これは URL リクエストによった結果を RSS で取得する Yahoo! Pipes⁴⁾ の翻訳 API を使用している (そのためインターネットに接続された環境からしか実行できない)。

`google` 命令、`yahoo` 命令は、その文字列の検索結果をブラウザに表示するという命令である。例えば

```
google "宮下芳明"  
yahoo "宮下芳明"
```

と実行すると、タブブラウザの場合は二つのタブに `google` による検索結果と `yahoo` による検索結果

をそれぞれ表示する。

`browse` 命令はその文字列が URL であると仮定しブラウザからそのページにジャンプする命令であり、`mail` 命令は、その文字列がメールアドレスであると仮定してメーラーを起動させる命令である。

画像ファイルを読み込み表示する場合は、前節の `draw` 命令を使用し、`draw.picture` "ファイル名" と記すだけでよい。対応する画像ファイル形式は `bmp`、`jpg`、`gif` 等である。表示される位置は `set.position` (座標 `x`、座標 `y`) によって指定する。

また `play` "ファイル名" を用いると、`mp3` ファイルや `wav` ファイルの音声を再生することができる。

2.5 タグ命令

提案言語では `{ }` の廃止を目指し、ループや条件判定のための「タグ命令」を用意した。タグを用いた言語としては、飯島が提案する XML で文法を与えるプログラム言語⁵⁾や、XML 形式の文法を持つ XML 用プログラム言語 Xi⁶⁾ などがあるが、提案言語においては大括弧が必要となる命令についてのみ、タグを用いてその開閉を表現することとしている。例えば、`HMMMML` を用いて 3 回 `Hello` と表示するには以下のように「`for` タグ」を使用する。

```
<for 3>  
  print "Hello"  
</for>
```

カウンタを使用する場合には、システム変数である `counter` を使用する。

```
<for 3>  
  print counter*2 // 0, 2, 4 と表示される  
</for>
```

多重化されたループを使いたい場合は、例えば以下のような記法を用いる。

```
<for 3>  
  i = counter  
<for 3>  
  j = counter
```

```
print i+" multiply "+j+" is "+(i*j)
</for>
</for>
```

条件が満たされている限り実行する処理は、while タグを用いて以下のように書ける。条件を省略して <while> ~ </while> と記した場合には、無限ループを作り出すことができる。

```
<while (a<4)>
  a = a+1
</while>
```

条件判定については if タグを用いる。その条件が満たされていない場合を記述したいときは else タグを用いる。

```
<if a = 0>
  print"無"
</if>
<else>
  print"有"
</else>
```

プログラムの実行ウィンドウに表示されるタイトルを指定するときには <title> タグを用いる。

コメントとして /* ~ */、// ~ 以外に <!-- ~ --> のタグを用いることも出来るが、このタグはラベルとしても機能し、link <!--ラベル名--> によってそのラベルにジャンプさせることができる。

また draw.button "ボタン名", <!--ラベル名--> によってボタンを作成することが出来る。このボタンが押されたときには指定したラベル名にジャンプする。

```
draw.button "PUSH", <!--ラベル名-->
```

実行時に無視されるタグとして、head タグ、body タグ、html タグ、hm タグ、main タグを用意している。これにより、例えば下のソースコードのように初期設定を head タグに整理し、body タグにメインルーチンを記述するといったことが可能になる。

```
<hm>
<head>
<title>サンプルプログラム タイトル</title>
  x = 10 //初期設定
</head>
<body>
  <!--メインルーチン-->
</body>
</hm>
```

2.6 他言語への対応

提案言語において、モチベーション向上だけでなく他の言語への橋渡しについても検討した。通常は、他の言語と親和性の高い文法をデザインすることで橋渡しを実現するものであり、例えば Processing は Java との親和性が高い文法であることから、Java への導入としての利用も可能である。しかしながら、HMMMML は大括弧を廃止してタグ命令を導入するなど、他言語にないコンセプトを中心としていることから、このようなかたちで親和性を実現することは実質不可能である。そこで、全く違うアプローチとして、「他言語における表現と混在できる」仕様を考案した。

たとえば、HMMMML の他に、Java, C, JavaScript, ActionScript, Processing, BASIC, HSP において Hello World の文字列を表示するためには、以下のような表現となるが、HMMMML エディタで上記のどの表現を用いても、print "Hello World" と同義と解釈し実行されるようにした。

```
HMMMML  print "Hello World"
Java     System.out.println("Hello World");
C        printf("Hello World");
JavaScript document.write("Hello World");
ActionScript trace("Hello World");
Processing println("Hello World");
BASIC    PRINT "Hello World"
HSP      mes "Hello World"
```

また、大括弧廃止の原則と一見矛盾するようだが、次のソースコードのように、大括弧とタグが混在し

たり、一部が C 言語であったり JavaScript であったりしても柔軟に理解して実行される。

```
if (a == 1) {
    printf("it is one");
}
<else>
    document.write("it is not one");
</else>
```

本質的な構造の違いに関わる命令については同居不能であるが、出来る限り多くの言語の表現を理解できるようにシステムを改良している。将来的には、異なる言語によるアルゴリズムをコピー&ペーストしても動作するようにしていきたい。

また、HMMMML は HSP コンパイラで動作しているため、HSP 標準命令によるプログラムはすべて HMMMML エディタにコピー&ペーストして実行できる。

3. 使用の実際

本章では、HMMMML を用いてどのようなプログラムが制作できるか、具体例を紹介していく。

次のプログラムは 10 からのカウントダウンを行うものである。「10」というダイアログが現れ、OK ボタンを押すと「ten」と読み上げた後に「9」というダイアログが現れ、その OK ボタンを押すと「nine」と読み上げ、…以下これをゼロまで続ける。

```
<for 11>
    show (10-counter)
    say (10-counter)
</for>
```

次のプログラムを実行すると、ブラウザが起動し、44 つのタブに「第〇代大統領」のインターネット検索結果が表示される。

```
<for 44>
    google "第"+(counter+1)+"代大統領"
</for>
```

次のプログラムを実行すると、1024 × 768 のウィンドウを作成し、明治大学の地図と公式ページを横に並べて表示する (図 2)。この例での記法は<head>タグと<body>タグを使用して html 風にしてある。

```
<head>
    <title>明治大学の情報</title>
    latitude=35.612394
    longitude=139.548758
</head>
<body>
    draw.screen (1024,768)
    set.position (0,0)
    draw.map (512,768,latitude,longitude)
    set.position (512,0)
    draw.browser (512,768,"http://www.meiji.ac.jp")
</body>
```



図 2. draw.map と draw.browser を用いた例

次のプログラムは、HMMMML エディタでの無限ループボタンをオンにして実行すると、マウスカーソルを中心として円を描き続けるため、簡易的なペイントツールとして動作する (図 3 左)。

```
fill.circle(mouse.x, mouse.y, 20)
```

このソースコードの半径のパラメータを乱数を用いて次のように書き変えると、筆の書き味のような効果を得ることが出来る (図 3 右)。

```
fill.circle(mouse.x, mouse.y, random(20))
```

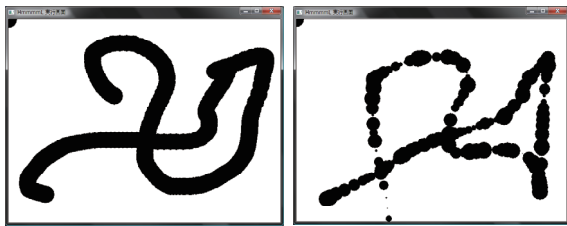


図 3. 1 行プログラムによる簡易ペイントソフト

さらに、このソースコードにおける `mouse.x`, `mouse.y` を `cursor.x`, `cursor.y` に置換して改造すると、マウスではなくカーソルキーで操作できるようになる (図 4)。

```
draw.circle(cursor.x*5, cursor.y*5, random(20))
```

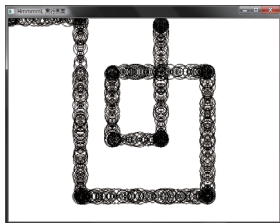


図 4. カーソルキーによる制御

無限ループボタンをオンにして次のプログラムを実行すると、次々と音色を変えながら半音階で上昇する旋律を奏でる。

```
i++
change.program(i)
play.note(i)
```

次のプログラムを無限ループがオンの状態で実行すると、(0, 0) - (100, 100) の範囲でマウスカーソルの位置がランダムに変わり、コントロールが奪われる。

```
mouse.x = random(100)
mouse.y = random(100)
```

次のプログラムを実行すると、PUSH と書かれたボタンが現れ、これをクリックするたびに `song0 ~ 9.mp3` の音楽ファイルがランダム再生される。

```
draw.button"PUSH",<!-- randomplay -->
stop
<!-- randomplay -->
play "song"+random(10)+".mp3"
```

4. おわりに

本稿では、プログラミングに対するモチベーション向上を目的とし、新しい言語 HMMMML を提案した。無限ループボタンやタグ命令といった特徴を持ち、シンプルなソースコードとしてデザインされ、1 行でも多様な表現が可能となった。

2009 年 9 月 29 日に、明治大学理工学部情報科学科 1 年生を対象とした授業「プログラム実習 2」の一貫として、本システムのデモンストレーションとアンケート調査を行ったところ、「この言語でプログラムしてみたい」「プログラミングが楽しく思えてきた」といった感想を得ることができた。現状では Microsoft Speech API 環境の問題等でコンピュータ室に HMMMML をインストールできていないが、今後はこの言語を洗練化するとともに、実際のプログラミング教育に用いることによってその効果を検証していきたいと考えている。また、次の改良点として、デバッグ過程に対するモチベーションを向上させる工夫を検討している。

参考文献

- 1) Processing <http://processing.org/>
- 2) Hot Soup Processor <http://hsp.tv/>
- 3) Microsoft Speech API
www.microsoft.com/speech/default.msp
- 4) Yahoo! Pipes
<http://pipes.yahoo.com/pipes/>
- 5) 飯島 正, XML で文法を与えるプログラミング言語. 情報処理学会 ソフトウェア工学研究会報告, IPSJ SIG Notes 98(64) pp.53-60, 1998.
- 6) 川道亮治, 佐藤誠. XML プログラム言語 Xi(ザイ)の完全な仕様策定とその実装 - 美しいツリー言語の提案 -, 平成 15 年度未踏開発ソフトウェア創造事業, 情報処理推進機構, 2003.
<http://www.ipa.go.jp/SPC/report/03fy-pro/mito/15-1319d.pdf>