Vol.13 No.4, 2011

原著論文

学習者のモチベーション向上のための好意的解釈を行うフィジカルコンピューティング環境のデザイン

宮下　芳明[*1*2]　中橋　雅弘[*1]

MOTIVATION: Physical Computing Environment Intended to Motivate the User

Homei Miyashita and Masahiro Nakahashi

Abstract - This paper describes MOTIVATION, the physical computing environment that is intended to motivate the user. By using this system the user can develop programs running under Windows, HTML5, and mbed. The compiler approves mistakes such as misspelling or missing variable declaration, and by using programming language HMMMML the user can make programs with a few lines.

Keywords: Physical Computing, HMMMML, HMMMBED, mbed, MOTIVATION

1. はじめに

近年，情報科学の進歩により，ソフトウェア・ハードウェアに関わらず何でもできる「高機能なツール」が増え，それは創造活動の一端も担っている．機能とは独立に UI を設計できるメリットを生かし，より使いやすいデザインはどうあるべきかについてこれまでも多くの議論がなされてきた．本論文は，ユーザがそのツールを使い始めるにあたって挫折することなく，最終的には使いこなせるようになるためにどのようなデザイン要件が必要なのかを考え，それをフィジカルコンピューティング環境に適用する試みを行ったものである．

ソフトウェア・ハードウェアに関わらず，広い意味でのツールを「使い始めたとき」の経験や感覚について，考えてほしい．はじめて車の運転席に座って発進させたときの感覚，はじめてスキーやローラースケートに乗ってすべったときの感覚，はじめて 3DCG ソフトや音楽制作ソフトでコンテンツを作ったときの感覚，はじめてホームページやブログを作ったときの感覚．あるいは，使い始めてすぐに挫折したときの経験や感覚…興味あって買ったソフトの使い方が難しくてマスターできなかった，そもそも環境構築で挫折した，楽器から思うように音を出すことすら難しかった，そういったときの感覚も思い出してほしい．紙面を割いて事例を列挙したのは，読者にも経験に照らして実感してほしいからである．

筆者らが考えるに，ツールを使い始めたときのユーザというものは，とても「心が折れやすい」状態である．そのツールに挑戦するだけの大いなる勇気を振り絞ってはいるが，小さなトラブルでくよくよしがちなナーバスな状態である．ちょっとした警告ダイアログが出るだけでも，叱られたような気持ちになる．本のとおりにやったのにそのとおりに動かない，そうしたちょっとしたトラブルでも大きな精神的ダメージを受け，「やっぱり自分には向いていない」と感じてしまうのである．ドライバのインストールや環境変数の書き換えなど，そのツールを使えるようにするための準備が煩雑なだけで辛いと思ってしまう．ましてや，ソフトのインストール作業のほかにレンタルサーバーの契約や設定など，他にすべきことがあったりするとすぐに心が折れてしまう．西本は，以下のように指摘している[1].

> …一般にこれらの道具の使用方法をマスターするためには膨大な時間と労力が必要となる．しかしながら，マスターしなければならない要素のすべてが，創造活動にとって必ずしも本質的であるわけではない．むしろ，本質的に必要な要素はほんのわずかしかなく，その他は本質に至るための付随的要素でしかない場合がきわめて多い．このような付随的要素の処理法に習熟するための時間と労力が，人々が創造活動に取り組むことを阻んでいる．たとえば，エベレスト山頂からの景色を写真撮影するために，まず高度な氷壁登攀の技能を身につけなければならないというのは，非常に大きな障害である．また，付随的要素の処理法を習得した者であっても，その実行のために認知的・身体的能力が消費され，本質的要素の処理に全力を注ぐことができない．氷壁登攀の疲労で，写真撮影に集中できないようでは本末転倒であろう．

*1 明治大学 大学院 理工学研究科 新領域創造専攻 ディジタルコンテンツ系
*2 独立行政法人科学技術振興機構，CREST
*1 Program in Digital Contents Studies, Program in Frontier Science and Innovation, Graduate School of Science and Technology, Meiji University
*2 JST, CREST

筆者らも，まずは基本的な考え方として，準備の大変さを取り除き，すぐに使い始められるようにすることが肝要であろうと考えている．その一方で，筆者らが注目したのは，ユーザがいったんそのツールを少し使えるようになると，「調子に乗っている」ともいえるような全能感を持ち，その後多少のトラブルや困難な壁があったとしても自力で乗り越えるだけの精神力を手にする，ということである．パソコン中級者がソフトのインストールで少々トラブルがあったとしても動じずに試行錯誤できるのは「試行錯誤を重ねれば自分はこの困難な状況をいつか打開できる」という自信を持っているからではないかと考えた．

ユーザを「心が折れやすい」状態から，全能感溢れ「調子に乗っている」状態に変身させるためのきっかけになるのは，最初の成功体験に他ならない．例えば自動車の運転のように，難しいものだと思い込んでいたものが，意外に簡単に動かせたとき，「ニヤける」体験がある．これは「嬉しい」「意外に簡単にできてしまって驚いた」，そして「もしかしたら自分には素質があるのかもしれない」と思えるナルシスト的な感覚が混在したような気持ちであると筆者らは考える．この「ニヤける」体験を誘発することができれば，そのあとは多少の困難があったとしても乗り越えられるだけの自信を，ユーザが持てるようになると考えた．ビギナーズラックによって最初に大勝してしまったギャンブラーは，調子に乗ってゲームを続け，その後に負けが何度か続いてもへこたれるようなことはなく，自分を信じて突き進む．ユーザを調子に乗せる（Pink のいうところの Drive[2]する）ことさえできれば，ユーザは自発的に使い方について学ぶにとどまらずに探求し，Csikszentmihalyi のいう Flow な状態[3]にも達しやすくなる．そういう精神面での大きな変化をいかに誘発するか，という観点でのシステムデザインを提案したい．

筆者らは，その誘発のために 3 つの条件が必要であると考えた．ひとつは，そのツールが「プロっぽい」本格的なデザインであり，実際に到達可能性のレベルが高いインタフェースであることである．使いやすいが将来的にはたいしたことができない，また外見上も可愛くて使いやすそうなデザインの場合，簡単にできてしまったのはそういう「おもちゃ」だからだとユーザは思ってしまう．よって，インタフェースの「本格感」を最初の条件として挙げたい．

もうひとつの条件は，操作内容が本質的でシンプルなものになるようにすることである．一番はじめにそのツールを使うにあたって，多くの手順を踏まなければならないと，それだけでユーザにかかる負荷が大きくなる．また，「意外にも簡単な操作でできた」という部分を演出することも重要だと考えている．西本[1]は，以下のように述べている．

> …現在創造活動に用いられている道具がサポートしてくれる行為は，最終的に実行したい創造行為と比較して非常に原始的なレベルにとどまる．前節で示したユニバーサル・デザインのペンを例にとれば，サポートされているのは「線を描く」行為にすぎず，「風景画を描く」行為とは大きな隔たりがある．この両者の間には多くの付随的要素が存在し，それらすべてを人が処理しなければならない．これが，創造行為実施の難しさや抵抗感を招いている．ゆえに，可能な限り付随的要素の処理を道具が吸収し，人に余分な負荷を課さず，本質的要素の処理にのみ集中することを可能とする「創造活動のためのユニバーサルな道具」が実現できれば，より多くの人々が創造活動に取り組めるようになるであろう．

このように，付随的要素の処理を道具が吸収することによって「本質的な操作」が残るようにし，その操作内容をできるかぎりシンプルにすることが必要だと考えている．例えばブログのテンプレートを選択してタイトルを入力するだけで，とりあえずホームページが公開できてしまうように．もちろんその後カスタマイズなど細かな要求に対応できるだけの「本格さ」が必要なのは前条件のとおりである．

そして，最後に筆者らが加えたい条件は，付随的要素を道具が吸収するのに加えて，ユーザのミスをも吸収する「好意的解釈」の導入である．「心が折れやすい」状態になっているユーザにとって，エラーダイアログが出たり動かなかったりすることは，モチベーションを低下させる大きな要因となっている．そして，「ユーザが何をしようとしているか」を推測することは，ある程度可能であると考えられる．実際，五十嵐らの Pegasus[4]や増井らの POBox[5]は，ユーザの意図に応じた補正を越えて，次の行動の予測まで行って支援することに成功している．ツールを使いやすくするためというよりは，「ユーザを調子に乗せる」ためにシステムが支援する，というのが本論文で最も独自の主張である．

もちろん，こうした過保護なインタフェースデザインに甘んじることなくユーザがさらに向上意欲をもって学習し，最終的にはツールが吸収していた付随要素や好意的解釈という「補助輪」を自ら外していけることが理想である．そのためのキーとなるのが，他者の存在である．他者との競争が Flow 体験につながることを

Csikszentmihalyi も指摘しているが，増井は自己満足(自満)や自慢を支援するジマンパワー[6]も有効であると述べている．著者らも成果を他者と競ったり他者に自慢できたりする機構を設ける必要性があると考えている．Stephanidis が提唱するユニバーサル・デザイン[7]においても，想定されるあらゆる種類の利用者における機能要件を初期設計段階で織り込み，利用開始時に簡単・低コストで適応可能（adaptability）としたうえで，利用開始後に，細かい好みや経験に伴う変化に対して適応可能（adaptivity）にする 2 段階の設計が考えられている．

まとめると，ユーザのモチベーション向上を目的としたシステムデザインは，準備の大変さの軽減，本格感，本質的な操作，好意的な解釈に加え，ユーザがさらなる表現力向上を目指し補助輪を自ら外す仕組みを設ければ良いのではないかと考えた．

2. システム設計の思想

情報処理教育委員会は，情報処理教育に関する提言の中で，すべての国民が情報処理の仕組みを，体験を通じて理解し，それに基づいて ICT を有効に活用できる能力を備えるようにすると述べている[8]．プログラミング教育もその一環であるはずだが，プログラミングが流行らない理由として，プログラミングを始めるまでの敷居は高く，気軽にプログラミングしてみようという気持ちになれないと増井らは指摘している[9]．また山崎らの考察[10]では，プログラミングに全く触れたことがない人たちにとっては，プログラミングの具体的なイメージがわかず，専門的で難しそうという印象を抱いていると述べられている．そこで，ここでは，前章における考え方をフィジカルコンピューティング環境のデザインに適用するときにどうしたらよいかを考えていく．

2.1 準備の大変さの軽減

まずは環境構築のための初期障壁となる，準備の大変さを取り除くべきである．例えばフィジカルコンピューティングボードとして代表的な Arduino[11]は，簡単になったとはいえ，最初にドライバや開発環境のインストール，さらには COM ポートの設定などが必要となってしまう．一方で，最近流行となりつつあるフィジカルコンピューティングボード，mbed[12]は，USB メモリとして認識されるのでドライバは不要であり，中に入っているショートカットをダブルクリックすることで（ウェブ上の）開発環境が現れる．プログラミングを始めたいと思ったらすぐに始められる環境が理想であろう．

2.2 本格感

次に，「ユーザを調子に乗せる」ための 3 条件をどのように適用すべきかを検討していく．まずは，本格感を醸しだす「プロっぽさ」についてである．今日，プログラミングの枠組みやスタイルは多様化しており，APP Inventor[13]，Lego Mindstorm[14]，プログラミン[15]，Squeak[16]，MAX/MSP/Jitter[17]，RecipeCollage[18]等のようにビジュアルなインタフェースを用いたプログラミングが，教育的目的などで推進されている．Scratch[19]は，ブロックを組み立てるようにあらかじめ用意された簡単なスクリプトを組み合わせてプログラミングを行うもので，子供でも条件分岐，繰り返し，オブジェクト指向といった考え方が効果的に身につくとされている．また，原田が開発した Viscuit[20]は，絵を描きマッチングを指定するだけでプログラミングを行うことができる．Fuzzy Rewriting を用いることによって柔軟な動きを実現し，子供でもプログラミングの楽しさを容易に知ることができる． 久保田が絵画的プログラミングとして提唱する Crowkee[21]も，キャンバスに絵を描くような感覚でプログラミングを行える画期的なものである．スマートフォンでプログラミングを行う環境として，ジェスチャで入力しソースコードを絵文字として表現する瀬戸らの研究[22]もある．こうしたプログラミング手法は，直感性やわかりやすさという意味で優れている面も多い．例示プログラミングの手法[23][24]も，実際にユーザが操作してそれを記録するというわかりやすい手法である．しかしながら，本論文のデザイン指針では，こうした手法は適さず，「キーボードで英文ソースコードを記述して実行するスタイル」が適しているといえる．日本語プログラミング言語「なでしこ」を用いて C 言語への導入を試みた井後らの研究[25]等とも方向性が異なる．

2.3 本質的な操作

また，西本[1]の指摘のように，付随的要素の処理を道具が吸収することによって「本質的な操作」を残すことは，言語仕様に大きく影響を与える部分である．例えば，#include <stdio.h> といった「おまじない的」な要素は，道具が吸収して動けるようにすべきであろう．長らは対象を大学生とし，Java への移行を考えたプログラミング言語 Nigari の開発を行った[26]．宣言を省略し，プログラムを自動的に可視化することで，プログラムの流れの直感的理解を促している．Reas らによって開発された Processing[27]も，Java をよりシンプルに洗練させた言語デザインが行われており，少ない行数で多様な表現が可能である．

筆者らは，「3 回繰り返す」というのが本質なら，カウンタ変数を宣言したり初期化式・ 継続条件式・再初期化式を指定したりするのは付随的であると考えた．さらに，「Hello と読み上げる」「指定された緯度・経度の Google マップを表示する」といった操作も，本質的には 1 命令で済ませられるべきではないかと考えた．

2.4 好意的解釈

好意的解釈については，まさにプログラミング環境に大いに取り込むべき要素だと筆者らは考える．前述のとおり，どんなツールであれ，エラーダイアログが出たり動かなかったりすることはユーザのモチベーションを著しく下げるが，プログラミングについても同様である．Programming2.0[28]を提唱する安村は，現在のプログラミング言語に対し，「構文指向で厳密であって、約束事が非常に多く、細かいエラーも許さないようになっている。」と指摘し，これに対して人間は「本来意味指向であって、構文的なことは忘れても意味内容は忘れない。また、人間同士の会話などにおいて、曖昧な表現を許容し、解釈にも自由性がある。 …（中略）…相手の人間が言ったことの中に曖昧なこと間違った言い方が含まれていてもお互いに許容している。」と指摘している．

これまでのプログラミング言語やその開発環境では，エラーとなっているところに下線を表示し「実行前に直させる」インタラクションデザインであった．前章の議論のとおり，「早く実行結果が見たい」というユーザの気持ちを優先させるのなら，間違いがあったとしても好意的に解釈して実行し，そのあとで正しい表現が何であったかを提示し学習を促す順序のほうがより適切であると考えられる．好意的解釈可能と考えられるミスは，例えば宣言忘れやセミコロン抜け，全角スペースはもちろん，スペルミスも最も一致度の高い命令を検索することによって推測可能である．さらに，解釈に幅があり複数通りの解釈ができる場合も，「どちらの解釈が正しいか」というダイアログは実行後に出す方がよいはずである．すなわち，複数通りの解釈があるなら複数の実行結果を示せばよい．極端な話，"Hello"とだけ記述した場合も，考えられる多くの候補を全部表示してしまえば良いと考えられる．これは'Hello World!' と記述するだけで文字表示が実行される APL[29]の考え方と少し近い．

2.5 ユーザが補助輪を自ら外す仕組み

最後に，ユーザがさらなる表現力向上を目指し補助輪を自ら外す仕組みをどのようにしてプログラミングに導入するかを検討する．ひとつは，前章で述べたジマンパワー[6]が発揮できるよう，ブラウザやスマートフォンで動作させて人に見せられるようにする機構である．もうひとつは，プログラムの正確さを競い合うランキングシステムの導入である．どれだけ支援を受けたかによってそのスコアが異なるようにすれば，高いスコアを出すために支援機構を自ら外していく動機となるはずである．

2.6 まとめ

以上の議論をまとめると，モチベーションを向上させるためのフィジカルコンピューティング環境のデザイン指針として，以下の要素を導入するとよいと考えられる．

A) インストールや実行が簡単なこと
B) キーボードで英文ソースコードを記述して実行するスタイルであること
C) おまじない的な部分が少ないこと
D) 本質的には 1 命令で済ませられるシンプルな言語体系であること
E) セミコロン抜けやスペルミスといった誤りは極力好意的に解釈し，まずは実行結果を見せること（指摘はその後のソースコード内で行うこと）
F) 複数の解釈が成り立つ場合も，どちらの解釈がよいかは複数の実行結果を提示後に尋ねること
G) ジマンパワーを発揮させるべく，ブラウザやスマートフォンで実行可能にすること
H) プログラムが正確なほど得点が高い／支援を受けていないほど得点が高いランキングシステムを導入し競える機構を設けること

3. 提案システム MOTIVATION

3.1 システム概要

提案するシステム MOTIVATION は，（フィジカル）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た言語 HMMMML (Homei Miyashita's Motivating Multi-Lingual Markup Language)を 2009 年 9 月に発表して以来，学会でのデモンストレーションと議論を繰り返し，フィードバックを反映させながらバージョンアップを続けてきた最終成果である[30][31][32][33]．第 1 章・第 2 章で述べてきたデザイン指針をすべて適用しプログラミング教育に活かそうとしている．現在，明治大学理工学部情報科学科の授業「ゼミナール 1」で，学部 1 年生が最初に触れるプログラミング言語として教育利用している．

MOTIVATION は，mbed[12]と☆board orange 内[34]（図 1 ）に収録された Windows プログラムである．PC の USB 端子に接続すると USB メモリとして認識され，

開発環境が立ち上がる．制作したアプリケーションは(1)Windows アプリケーション，(2)スマートフォン上でも動かせる HTML アプリケーション，そして，(3)mbed マイコン単体で動作する mbed アプリケーションの 3 種類の書き出しが可能である．

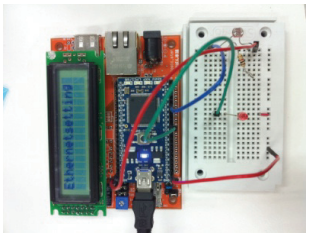


図1 MOTIVATION がインストールされた mbed[12]と☆board orange[34]，および接続されたブレッドボード（動画は[35]参照）

3.2 好意的解釈

開発環境 MOTIVATION は，スペルミスをも許容する超好意的解釈コンパイラとその度合いを制御するスライダ等の機能を搭載している．実際の動作は，超好意的解釈のプレゼンテーション動画[36][37]を参照されたい．

提案システムは「間違いがあったとしても，好意的な解釈によってとりあえず実行し，その後でコンパイラがどこを修正したのかをユーザに教える」ものとなっている．例えば if 文の条件式内で「a=1」（言語 HMMMML では代入表現）と書いた場合，提案システムではこれを「a==1」と好意的に解釈して実行した上で，ソースコードには「a=1 を a==1 に修正しました」とコメント文が入るようにした（図 2）．

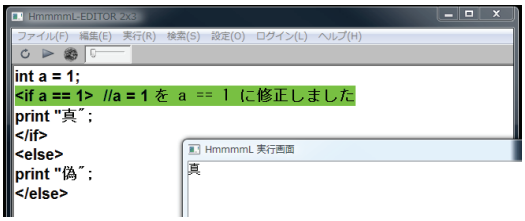


図2 解釈結果の実行後提示

提案システムでは，プログラムに直接関係ないと思われる全角スペースや 2 バイト文字は全てコメントとして好意的に無視するようにした．これにより，通常コメントを書く際に必要な//や/*～*/を忘れたとしても，それが 2 バイト文字であれば問題なく動作する．

また，予約語にない命令，例えば ptint "Hello"という文字列があったとき，これは print を意図して記述されている可能性が高いので，コンパイル時に print と置き換えて実行するようにした．コンパイラ内部のアルゴリズムとしては，予約語との一致率を計算して類似した予約語に置換している．

言語仕様では，p ではじまって文字列が続く命令が print しかないため，p "Hello"と記述しても print "Hello"と解釈される．これに対し，s "Hello"と書いた場合は，say(人工音声で読み上げ）と show(ダイアログ表示)の 2 通りが考えられる．提案システムでは，複数通りの解釈が成り立つ場合は，その全ての解釈を実行することとした．タブを切り替えることで実行結果を見ることができ，「採用」ボタンを押してその解釈を選択できる（図 3）．

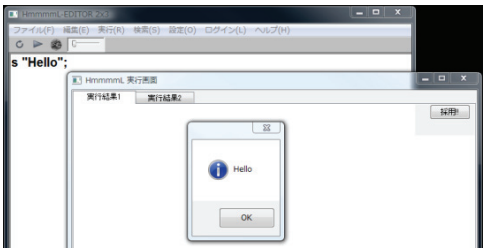


図3 複数解釈に対するタブ表示

例えば図 3 において，実行結果 1 を選択すると「Hello」が読み上げられ（say としての解釈），実行結果 2 を選択すると「Hello」というダイアログが表示される（show としての解釈）．タブ 1 を選択して採用ボタンを押すと，ソースコードは

say "Hello"　//s を say に修正しました

という内容に修正されている．

さらに，現状のシステムで好意的に解釈できない特殊なエラーが発生した場合は，そのプログラムをサーバー上にアップロードする機能を備えた．これにより集められた多くのプログラムからエラーの典型を導き出し，コンパイラの解釈機能強化に役立てることができる．

提案システムはプログラムの正確さの評価を行い，スコアとして換算する．スコアは命令数に比例，修正数に反比例する換算式を用いている．

さらに，エディタのオンラインスイッチをオンにすることで，ユーザの成績をサーバーにアップロードする機構を設けた．図 4 のランキングサイトで全国のユーザのプログラミング成績のランキングが閲覧できる．エディタに twitter のアカウントでログインしていれば，ユーザ名にはそのアカウント名が表示され，それを用いてユーザ同士で健闘を讃えあうこともできる．また，これによりプログラマのコミュニティが容易に形成できるのではないかと考える．

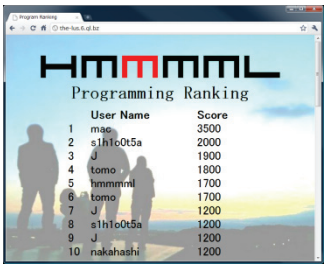


図4　ランキングサイト

提案システムでは超好意的解釈のみに依存しないシステムとして，超好意的解釈- 超頑固解釈を調整できるスライダを用意した．これにより，ユーザはコンパイラの解釈度合いを超好意的解釈，好意的解釈，超頑固解釈の三段階に調節が可能となった．好意的解釈では命令文のスペルミスを除くケアレスミスを許容し，超好意的解釈ではそのスペルミスをも許容してくれる（図 5 ）．

図5　超頑固解釈でのエラー画面

超頑固解釈では，エラーがある場合それを知らせるダイアログのみを表示し，実行結果は表示しない（図 6 ）．スライダを好意的解釈に依存しない位置に設定するほど，スコアで高得点を得ることができる．こうすることでいわばユーザが自ら好意的解釈の補助輪を外し，より高度なプログラミングへと挑戦できる．

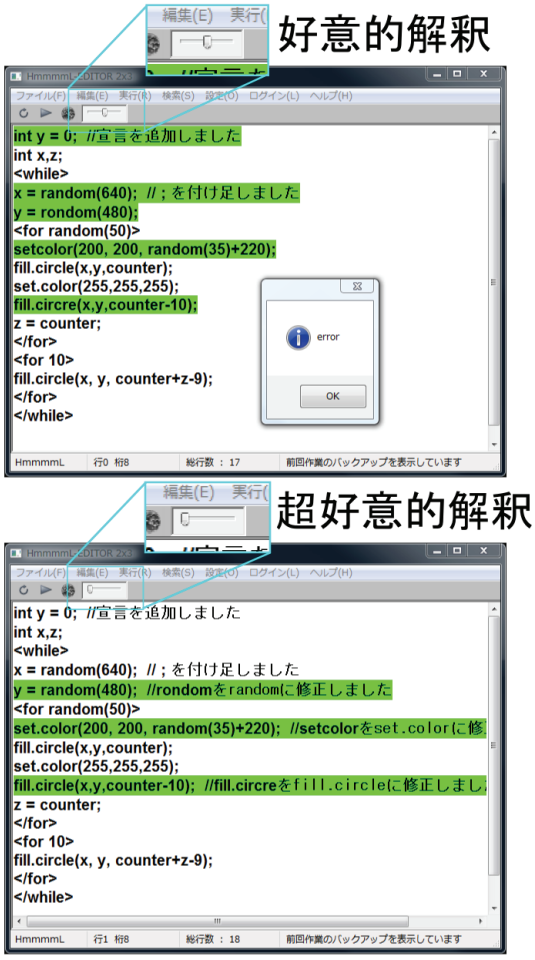


図6　好意的解釈，超好意的解釈の比較

4.　システムと言語の仕様について

システムは mbed[12]内に入っており，USB ドライブとして認識される．中にあるエディタを起動すればそのままコードを記述することができる．ドライバやソフトウェアのインストールは一切必要ない．以下に，採用されている言語 HMMMML の仕様を述べる．デモンストレーションについては動画[38][39][35]を参照されたい．

例えば HMMMML で 3 回 Hello と表示するときには，「for タグ」を使用して以下のように書ける．

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

カウンタとしての変数の宣言や初期値設定;条件式;増分処理を指定する必要がない．

マウスカーソル位置に半径 20 の円を描画するには

draw.circle(mouse.x, mouse.y, 20);

と書ける．エディタに備わっている無限ループボタンをオンにして実行すると，ペイントツールのようなプログラムを 1 行で実現できる．ESC キーやウィンドウの

閉じるボタンでループから脱出することができる．

say "Hello";と実行すると"Hello"と読み上げ，google "Hello";と実行すると"Hello"の検索結果をブラウザに表示する．tweet "Hello";と実行すると"Hello"と twitter 上に投稿する．MIDI で音を鳴らすには play.note(ノート番号)， Google マップを表示するには draw.map(横幅，縦幅，緯度，経度)と指定する．動画ファイル・音楽ファイルも全て play "ファイル名"で実行する．

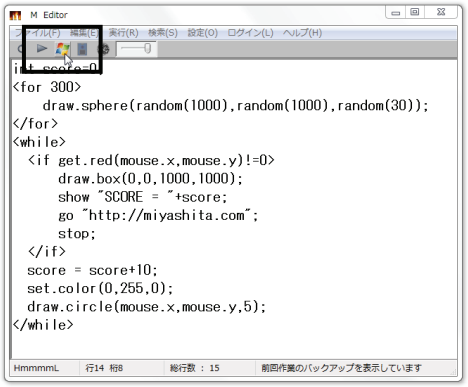


図7 3 種類の書き出し方法
(Windows アプリケーションを選択)

提案システムは，Windows アプリ，HTML5 アプリ，mbed アプリの 3 種類の書き出しを行える（図 7 ）．Windows アプリケーション書き出しによる主要機能は，動画[38]を参照されたい．また，このモードでは，Km2Net 社の USB-I/O[40]に対応している．0 番のピンに通電したいときは，on(0)と実行することで通電し，切断したいときは off(0)を実行すればよい．例えば，マウスの座標によって 0 番ピンに接続された LED を点灯させるプログラムは，以下のとおりである．

<if mouse.x>100 > on(0) </if>
<else> off(0) </else>

出力ポートは 0～7 番のポート番号で指定できるが，8 番ポートを指定すると，6 章で後述する仮想ブレッドボードソフトウェア HMMBB-V に通電することができる．(バーチャル／リアルの LED を交互に動かすデモ動画[41][42]参照)

Windows アプリケーションの多くの命令は HTML5 を用いて実行可能である．例えば say 命令の場合，Google Translate API の読み上げ機能(非公開 API)を用い，ここから返ってくる mp3 ファイルを HTML5 の<audio>要素で再生することで実現した(say 命令専用の画面を html の<frame>タグを用いて隠し，そのページで API を使っている)．エディタは同一で，実行ボタンを押すと Chrome が立ち上がって実行する仕組みになっている．

完成したプログラムは，アップロードボタンを押すだけで，特定の URL へとアップロードされる．プログラムは実行ボタンからも実行できるが，ブラウザにその URL を打ちこむことでも実行できる．スマートフォンを所持していれば，自作のプログラムをいつでも，どこでも見せることが可能である．このようにプログラムを容易に他人に見せられる環境を提供したことで，ユーザは自分が作ったプログラムを用いて誰かを楽しませる体験ができる．プログラムを HTML5 で実行するデモは動画[39]を参照されたい．

最後の mbed アプリとしての書き出しを行えば，mbed マイコン単体で実行することも可能である．実際に内部で行っていることは，パーサによってコードを変換し，IE コンポーネントを用いて mbed のサイトにアクセスし，マウスイベント，キーボードイベントを送り込むことでそのソースの貼り付け，コンパイル，mbed へのコピーまで全自動で行っている．すなわち，実行ボタンを押したのちに mbed のボタンを押すだけで，マイコン自体が実行を開始する．（mbed でコンパイルまでの一連の操作は動画[35]を参照されたい）

なお，3 種類の書き出し方法では，それぞれ互換性のない命令（mbed で音声合成命令 say など）が存在するが，それらｈ好意的解釈によってコメントアウトし，非対応命令である旨のコメントがつけられる．

5. 作例

提案システムは，これまで多くのデモンストレーションの機会を得てきたが，その過程でできあがったプログラムの例を２つ示す．ひとつは，スマートフォン上で動作する 14 行のゲームプログラムである．このプログラムは，ランダムな場所に球を配置し，その球にぶつからずに長い軌跡を描くほど得点が高くなるゲームである．ゲームオーバーになると，指定した URL にジャンプするようになっている（図 8 ）．

```
score=0;
<for 99>
  draw.sphere(random(400),random(400),random(20));
</for>
<while>
 <if get.red(mouse.x,mouse.y)!==0>
   draw.box(0,0,400,400)
   show "SCORE = "+score
   go "http://miyashita.com"
 </if>
 score = score+10;
```

```
  set.color(0,255,0);
  draw.circle(mouse.x,mouse.y,3);
</while>
```

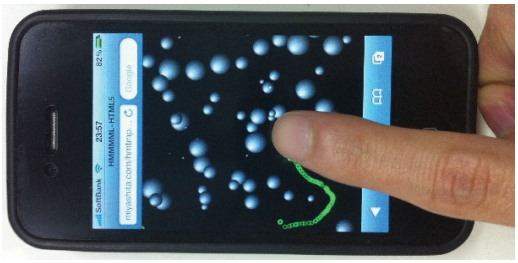


図8　スマートフォンでの実行

次に，mbed で動作する，光量によって twitter 上につぶやくプログラムの例を示す．図 1 のように 15 番ピンに CdS セルをつなぎ，下記のように 8 行のプログラムを記す．PC から接続をはずしても，携帯充電バッテリーなどで USB に給電し，☆board orange に LAN ケーブルをつなげば単体で動作する．

```
set.position(0,0);
print(in(15));
set.position(0,1);
<if in(15)<0.1 >
on(4);
tweet"Hello World!!";
print"transmitted";
</if>
```

6.　「ゼミナール１」授業での成果報告

6.1　授業概要

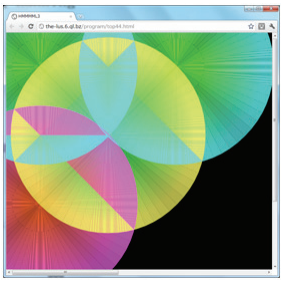
MOTIVATION を用いたプログラミングの授業を，情報科学科 1 年生 27 名を対象に行った．対象者のほとんどがプログラミング初心者で，簡単な繰り返しや条件分岐を習い始めた段階である．授業は 70 分程度のものを週に 1 度，計 4 回行った．1 回目の授業で MOTIVATION の一行命令をいくつか体験してもらい，図形を描く命令とループ命令を用いた演習を行った．2 回目の授業では<if>タグを用いた条件分岐の説明をし，それを用いた演習を行った．3 回目の授業では，自由にプログラミングをしてもらい，最後に発表会を行った．4 回目の授業では MOTIVATION と MBED を用いてフィジカルコンピューティングの体験をしてもらった．4 回目の授業の最後に簡単なアンケートに答えてもらった．

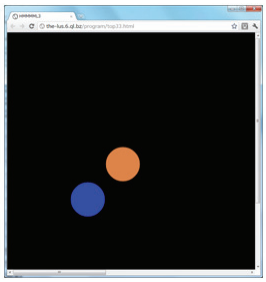
6.2　授業風景

3 回目の授業で，自由にプログラミングをしてもらったが，当初設けた時間を越えてもまだプログラミングしたいという生徒が多く，実習時間を延長した．このように積極的にプログラミングに取り組む姿勢が，授業を通して感じられた．図 9 の(c)に生徒が作ったプログラムの例を示す．左は無限ループを用いて 4 つの円が次第に大きくなっていくプログラムである．右の例は等速移動する円が壁に当たると反射し，さらにその円の色が変わるというプログラムである．このほかにもアニメーション表現を用いた作品が多く，マウスカーソルの動きに合わせて変化するインタラクティブなプログラムもあった．

(a)プログラミング風景

(b)MBED の回路作成





(c)オリジナルプログラム

図9　授業風景

6.3　アンケート結果

表 1 にアンケートの集計結果を示す．ここではアンケートの中に書かれていた意見の一部を示している．①の「プログラミングに対する印象は変わりましたか」という質問に対して，「変わった」「少し変わった」という回答が多く得られ，その理由として「難しそうに見えても以外に簡単に書くことができること」，「簡単に作れるので楽しいと感じられた」などの意見が得られた．②の「プログラミングは楽しいと思いますか」という質問に対しても「思う」という回答が多く得られ，その理由として「自分の打ち込んだことが，実際に目に見えるところが楽しいと思う」，「何でもできそう」などの意見が得られた．「あまり思わない」と回答した人の理由としては「難しい」，「苦手だから」という意見が得られた．③の「プログラミングに対するモチベーションは上がりましたか」という質問に対して，「上がった」「少し上がった」という回答が多く得られ，その理由として「自分で

もやればできると思ったから」,「もっと知りたいと思った」などの意見が得られた．④の「好意的解釈はあったほうがいいですか」という質問に対して「あったほうがいい」という回答が多く，理由としては「便利だから」,「つまらないミスで頭を悩ます必要が無く，別のことに集中できるから」などの意見が得られた．反対に「なくていもいい」と回答した人の理由としては「よくわからなかった」という意見と,「甘えてしまうから」などの意見が得られた．

アンケートの結果を整理すると，授業を通して多くの生徒を「自分はプログラミングの才能があるのではないか」と良い意味で勘違いさせることに成功したといえる．また，発表会を行うことでプログラムを作り，他人に見せる楽しさも体験させることができた．

表 1 アンケート集計結果

①HMMMMLでプログラミングをしてみてプログラミングに対する印象は変わりましたか？

変わった	少し変わった	それほど変わらない	変わらない	
9	12	5	0	

②プログラミングは楽しいと思いますか？

とてもそう思う	思う	あまり思わない	全く思わない	未回答
11	11	3	0	1

③HMMMMLでプログラミングに対するモチベーションは上がりましたか？

上がった	少し上がった	変わらない	少し下がった	下がった
12	9	5	0	0

④好意的解釈はあったほうがいいですか？

あったほうがいい	なくてもいい	未回答
19	6	1

⑤無限ループボタンはあったほうがいいですか？

あったほうがいい	なくてもいい	
23	3	

⑥命令の書きかたはわかりやすかったですか？

わかりやすかった	ふつう	わかりにくかった
14	12	0

⑦命令の最後にセミコロンはあったほうがいいですか？

あったほうがいい	なくてもいい	
22	4	

⑧<if>タグや<for>タグはわかりやすかったですか？

わかりやすかった	ふつう	わかりにくかった
19	7	0

7. おわりに

フィジカルコンピューティングを容易にするアプローチとして Phidgets[43]，Arduino[11]，Gainer[44]といったモジュールは広く使われている．Gainer はメディアアートにおける支援をひとつの目的としているが[45]，インスタレーション作品[46]や新楽器開発[47]にも応用されている．高松らは，フィジカルコンピューティングを用いた授業支援の研究を行っている[48]．森らは，造形教育プログラムに応用することを試みている[49]．姉崎らの提案する開発環境　Intuino では，タイムライン上に出力値を描くシーケンサ的なインタフェースによってその動きの調整を容易にしている[50]．他にも Visible Bread board[51]や react3D Electric[52]，そして秋田らの LED Tile 電子ブロック[53]のように，AR 的に電流を可視化するシステムが挙げられる．

8. 謝辞

最後になったが，これまで 1 年半にわたり，HMMMML1，2，3 について多くのフィードバックをくださった方々に感謝したい．それらのフィードバックと改良のおかげで，ここまでの機能と議論を実現することができた．一連の機能をひとつに統合しひとつの成果とすることで，さらなる議論を行えたらと考えている．そのうえで，好意的解釈構文解析の理論構築，好意的解釈の科学的定義，人間の間違いやすさを定量化した設計などについて考えていきたい．

9. 参考文献

[1] 西本一志. 創造活動のためのユニバーサルな道具とは, エンタテインメントコンピューティング 2006 予稿集, pp.7-8, 2006.

[2] Daniel H. Pink. モチベーション 3.0 持続する「やる気！」をいかに引き出すか, 講談社, 2010.

[3] Mihaly Csikszentmihalyi. フロー体験 喜びの現象学. 世界思想社, 1996.

[4] 五十嵐健夫, 松岡聡, 河内谷幸子, 田中英彦. 対話的整形による幾何学的図形の高速描画. 情報処理学会論文誌, Vol.39, No.5, pp.1373-1384, 1998.

[5] 増井俊之. インターフェイスの街角(5) - 予測型テキスト入力システム POBox. Unix Magazine. Vol.13, No.4, 1998.

[6] 増井俊之. 「界面潮流」第 3 回 ジマンパワー http://wiredvision.jp/blog/masui/200707/200707131029.html

[7] Stephanidis, C.: Adaptive Techniques for Universal Access, User Modeling and User-Adapted Interaction, Vol.11, pp.159-179, 2001.

[8] 情報処理学会情報処理教育委員会："日本の情報教育・情報処理教育に関する提言 2005", http://www.ipsj.or.jp/12kyoiku/proposal-20051029.pdf/

[9] 増井俊之, 塚田浩二. 全世界プログラミング, 情報処理学会夏のプログラミング・シンポジウム, September 2006.

[10] 山崎充, 渡辺和大, 藤岡直矢, 皆月昭則. プログラミング教育における導入期の苦手意識の変化に関する一考察. 第 8 回情報科学技術フォーラム(FIT2009)論文集，Vol.4, pp.519-520, 2009.

[11] Arduino. http://www.arduino.cc/ ,2010.

[12] Philips. 'mbed' - Rapid Prototyping for NXP LPC Microcontrollers in Minutes http://ics.nxp.com/literature/presentations/microcontrollers/pdf/mbed.pdf

[13] App Inventor http://appinventor.googlelabs.com/about/index.html/

[14] Seymour Papert. Mindstorms: children, computers, and powerful ideas. ACM classic books series. Basic Books, Inc., New York, NY, USA, 1980.
[15] プログラミン http://www.mext.go.jp/programin/
[16] Cardelli, L. and Pike, R. Squeak: a language for communicating with mice. In Proceedings of the 12th Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH '85. 1985.
[17] Max/MSP/Jitter http://cycling74.com/products/maxmspjitter/
[18] 吉川祐輔，宮下芳明. 料理プログラミングの為の枠組みについて, 情報処理学会夏のプログラミング・シンポジウム報告集, No.2010, pp.29-36, 2011.
[19] M. Resnick, J. Maloney, A. Monroy-Hernandez, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: Programming for all. Communications of the ACM, Vol.52, No.11, pp.60-67, November 2009.
[20] 原田康徳. 子供向けビジュアル言語 Viscuit とそのインタフェース, ヒューマンインタフェース研究会報告, pp.41-48, 2005.
[21] 久保田秀和, 西村拓一. 絵画的プログラミング, 第 51 回プログラミング・シンポジウム論文集, pp.105-114, 2010.
[22] 瀬戸優之，宮下芳明．ジェスチャ操作で入力する絵文字ソースのインタラクティブメール，インタラクション 2011 論文集, 2011.
[23] 増井俊之. インターフェイスの街角(47) - 例示プログラミング. Unix Magazhine. Vol.16, No.11, 2001.
[24] Hartmann, B., Wu, L., Collins, K., and Klemmer, S. R. Programming by a sample: rapidly creating web applications with d.mix.In Proceedings of UIST2007, 2007.
[25] 井後宏康，原佑輔，松江信太郎，吉末千紘.「なでしこ」によるプログラミング言語の導入, http://mis.edu.yamaguchi-u.ac.jp/kaisetu/sotsuron2007/n-ihmy-resume.pdf
[26] 長慎也，甲斐宗徳，川合晶，日野孝昭，前島真一，筧捷彦．Nigari-Java 言語へも移行しやすい初学者向けプログラミング言語，コンピュータと教育研究会報告，pp.13-20, 2003.
[27] Casey Reas and Benjamin Fry. Processing: a learning environment for creating interactive Web graphics. In ACM SIGGRAPH 2003 Web Graphics (SIGGRAPH '03), 2003.
[28] 安村通晃. Programming2.0：ユーザ指向のプログラミング, 情報処理学会夏のプログラミング・シンポジウム報告集, No.2006, pp.115-122, 2007.
[29] APL http://japla.sakura.ne.jp/
[30] 宮下芳明. 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, 第 51 回プログラミング・シンポジウム論文集, pp.57-64, 2010.
[31] 中橋雅弘, 宮下芳明. HMMMML2：超好意的に解釈するコンパイラ, 情報処理学会夏のプログラミング・シンポジウム報告集, No.2010, pp.107-110, 2011.
[32] 宮下芳明．フィジカルコンピューティングへのモチベーションを向上させるブレッドボード，夏のプログラミング・シンポジウム報告集，pp.1-4，2010.
[33] 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 論文集, 2011.
[34] ☆board Orange： mbed 評価用ベースボード http://mbed.org/users/logic_star/notebook/star_board_orange/
[35] mbed での実行デモ http://www.youtube.com/watch?v=HBpAyfcQ5Bg
[36] 超好意的解釈のプレゼンテーション http://www.youtube.com/watch?v=7h0EedR5C3Y
[37] 超好意的解釈のプレゼンテーション http://www.youtube.com/watch?v=dpOg6bIsU1k
[38] HMMMML & HMMMML2 を用いた Windows アプリのデモンストレーション http://www.youtube.com/watch?v=2x38UGp5iIA
[39] HTML5 での実行デモ http://www.youtube.com/watch?v=mZ19RC09Pqs
[40] Km2Net USB-I/O http://km2net.com/usb-io/index.shtml
[41] HMMBB+AV Cable 1/2 プレゼンテーション 1 http://www.youtube.com/watch?v=CAHaTn8PPjA
[42] HMMBB+AV Cable 1/2 プレゼンテーション 2 http://www.youtube.com/watch?v=N_uke5_6Jog 中村亮太, 西田知博, 松浦敏雄. 高等学校での「プログラミング」教育の導入：PEN を用いて, 情報処理学会研究報告, pp.41-47, 2008.
[43] Saul Greenberg and Chester Fitchett. Phigets: easy development of physical interfaces through physical widgets. Proceedings of the 14th annual ACM symposium on User interface software and technology, pp.209-218, 2001.
[44] Shigeru Kobayashi, Takanri Endo, Katsuhiko Harada, and Shosei Oishi. Gainer: a recongurable i/0 module and software libraries for education. Proceedings of the 2006 Conference on New Interfaces for musical expression, pp.346-251, 2006.
[45] 原田克彦, 小林茂:GAINER: メディア・アーティストのための再構成可能な I/O モジュール, 情報処理学会研究報告 [音楽情報科学], vol.2005, No.129, pp.7-11,2005.
[46] 長嶋洋一. サウンド・インスタレーションのプラットフォームについて, 情報処理学会研究報告 Vol.2007, No.50 (2008-MUS-75). (2008-HCI-128), 情報処理学会, 2008.
[47] 長嶋洋一. シーズ指向による新楽器のスケッチング, 情報処理学会研究報告 2009-MUS-80, 情報処理学会, 2009.
[48] 高松 智弥, 加藤 美和, 小濱 隆司, 宮川 治. フィジカルコンピューティングを用いた授業支援システムの提案, 情報処理学会研究報告. コンピュータと教育研究会報告, pp.1-8, 2010.
[49] 森 公一, 有賀 妙子. Sensory vision フィジカル・インタラクションによる造形教育プログラムの開発, 情報処理学会研究報告. グラフィクスと CAD 研究会報告, pp.1-5,2011.
[50] 姉崎 祐樹, 脇田 玲. Intuino: GUI によるフィジカルコンピューティング開発支援環境の構築, 情報処理学会研究報告. HCI, ヒューマンコンピュータインタラクション研究会報告, pp.1-4, 2010.
[51] 落合陽一.「電気がみえる」デバイス Visible Breadboard, 日本バーチャルリアリティ学会論文誌 Vol.15, No.3, 2010.
[52] Frank Uhling. react3D Electric - Tangible User Interface. http://www.youtube.com/watch?v=6qTTQKWfV8Q
[53] 秋田純一, LED Tile Block: パターン描画による機能設定可能な相互接続型ブロック型デバイス, インタラクション 2011 論文集, pp.699-702, 2011.

（2011 年 5 月 10 日受付，8 月 8 日再受付）

著者紹介

宮下 芳明（正会員）

2001 年 千葉大学工学部画像工学科卒業，2003 年 富山大学大学院教育学研究科音楽教育専修修了，2006 年 北陸先端科学技術大学院大学知識科学研究科博士後期課程修了，博士(知識科学)．2007 年より明治大学理工学部情報科学科に着任，2009 年より准教授，現在に至る．研究室サイト http://miyashita.com

中橋 雅弘

2011 年 明治大学理工学部情報科学科卒業．同年より，明治大学大学院理工学研究科新領域創造専攻博士前期課程に在籍．