教育用プログラミング言語としての「言霊」と「ことだま on Squeak」の試み

岡田　健*・杉浦　学*・松澤　芳昭*・大岩　元**
慶應義塾大学政策・メディア研究科*・慶應義塾大学環境情報学部**

我々は初心者に対するプログラミング教育において使用する環境として，日本語プログラミング言語を提案してきた．教育用プログラミング言語として開発されたプログラミング言語「言霊」と，Squeak を拡張する形で開発されたプログラミング環境「ことだま on Squeak」のそれぞれの特徴と問題点を解説する．そして教育用プログラミング環境として「ことだま on Squeak」が将来持つべき特徴を議論する．

1. はじめに

コンピュータは現代社会に不可欠な存在となり，それに伴いコンピュータ教育の必要性も叫ばれている．高校教育では２００３年より情報科目が必修となり，また中学校学習指導要領のなかでも技術・家庭で「情報とコンピュータ」が大きく取り上げられているのはその表れである．だが現実にはプログラミングを題材とした教育はあまり行われていない．

プログラミング教育の実践が難しい理由の１つに，プログラミング言語の可読性の問題がある．プログラミング学習の第一歩は，プログラムの読解から始まる．だがCやＪａｖａのプログラムは初心者には全く理解できず，それが学習の障壁となってしまう．

プログラミング言語の可読性の低さは，学習者がプログラムとシチュエーションを結びつけることを妨げる．例えばプログラミング言語の繰り返し文の学習で，コード１が学習者に提示されたとする．学習者は以後，何らかの繰り返し処理を記述するときにはこのプログラム例を参考にして記述することになる．

```
for( int counter=0 ; counter<4 ; counter++ ){
  fd(100); rt(90);
}
```

コード 1

だが多くの学習者は単純にある処理を何回も繰り返すプログラムは書けるようになるが，for文における初期化処理（int counter=0）・繰り返し条件（counter<4）・更新処理（counter++）が持つ意味は理解できない．学習者はそれらの意味を理解することなく，ただ繰り返し回数の部分を書き換え，繰り返す処理内容を｛｝の間に記述する．これでは繰り返しを理解できたと言えないだろう．

プログラミング学習を妨げるもう一つの要因は，プログラムの記述環境の貧弱さにある．プログラミング授業の多くにおいて受講者は，テキストエディタでプログラムを記述し，シェル上でコンパイラを使う．コンパイルエラーやデバッグプリントもシェル上で読む必要がある．

このような貧弱な記述環境では，プログラミングの論理的な思考を妨げてしまう．特に初心者教育の弊害になるのはコンパイルエラーである．初心者にはコンパイルエラーを除去することは難しく，プログラミングの論理的な思考を議論するまでに至らないことが多い．

2. 日本語プログラミング言語「言霊」

プログラミング言語の可読性を高めることを目的に開発されたのが，日本語プログラミング言語「言霊」[1][2][3]である．プログラムが日本語で記述されていて誰もが意味を読みとることが出来れば，プログラムの読解による学習を円滑に進めることが出来る．

日本語プログラミング言語「言霊」のコンパイラは，

Javaバーチャルマシンをターゲットマシンとして開発された．以下のコード２は，日本語プログラミング言語「言霊」のソースコード例である．コンパイルするとJavaクラスファイルが生成され，JavaVM環境で実行させることが出来る．

```
※各教科の点数を設定する
国語点数を作り、４９を代入する。
数学点数を作り、７３を代入する。
理科点数を作り、１００を代入する。
公民点数を作り、４５を代入する。
英語点数を作り、２５を代入する。

※５教科の合計を求める
合計点を作り、国語点数＋数学点数＋理科点数＋公民点数＋英語点数を代入する。
※５教科の平均点を求める
平均点を作り、合計点÷５を代入する。

※平均点を四捨五入する
平均点を１０倍する。
平均点を１０で割り、その余り≧５ならば｛
        平均点に１０を足す。
｝をする。
結果を作り、平均点÷１０を代入する。
```

コード 2「言霊」のコード例
各教科のテストの平均点を求めるプログラム

「言霊」の特徴を，以下に４点述べる．

2. 1. 日本語の語順と文法で記述されている

従来のプログラミング言語の多くは欧米で開発されたために，英語の語順（動詞が２番目，目的語が後ろに記述される）を用いてプログラムを記述することを強いられる．日本語の語順とは異なるため，日本人はプログラミングするときに頭の中で翻訳する必要が生じる．

「言霊」における文章表現は全て，正しい日本語の語順（目的語が文の真ん中，動詞が末尾に記述される）を用いて記述されている．語順が変わると，日本語で作りたいプログラムを考えて，その日本語をそのままプログラムに記述することが出来るようになる．筆者の個人的な体験になるが，日本語だけでプログラムを考えて記述すると気持ちよさを体験できる．それは従来の言語と異なり翻訳する必要がないために負担が減っているためではないかと推測される．

2. 2. 略記を避けて，正しい日本語表現を用いる

言霊における文章表現は全て，正しい日本語になるように配慮されている．プログラミング言語の表記にありがちな略記法は，可能な限り排除している．

例えばコード３は「言霊」で変数宣言をする際に使える記述である．どの記述例でも日本語として正しく読める表現になっていることが分かる．

```
国語点数を作る。
国語点数（整数型）を作る。
国語点数を作成する。
国語点数（整数型）を作成する。
```

コード 3「言霊」の変数宣言例

日本語として正しい表記をする理由を説明するために，比較対象として日本語プログラミング言語「なでしこ」[4]における変数宣言のコード例をコード４に紹介する．「なでしこ」は末尾に動詞が記述されていないため，正しい日本語として記述されているとは言えない．また動詞が記述されていないため，この文章の意味は読み取りにくい．

```
国語点数とは整数。
```

コード 4「なでしこ」の変数宣言例

プログラミング言語は，一般的にプログラミングが得意な人が迅速にコードを記述できるように設計されることが多い．「なでしこ」で動詞を省略しているのは，宣言文を記述するときに「作成する」などの動詞を記述するのが煩わしいからであろう．初心者から見たプログラムの読みづらさは，熟練者のための略記法を採用することによって引き起こされる．「言霊」は熟練者のための略記法を採用することを極力避け，日本語として正しく読解できる表現を採用した．また初心者が意味を理解しやすくなるような表現を採用した．

2. 3. 動詞などの活用語尾に対応している

「言霊」は活用語尾を活用しても正しく解釈できるような言語仕様になっている[3]．前ページのコード２の２行目では「～を作り，～を代入する。」という表現になっている．「作り」は連用形で記述されている．「言霊」における動詞は，終止形だけでなく連用形，命令形を用いて記述することが可能である．

```
国語点数を作る。４９を代入する。
数学点数を作る。７３を代入する。
理科点数を作る。１００を代入する。
公民点数を作る。４５を代入する。
英語点数を作る。２５を代入する。
```

コード 5 活用語尾が無かった場合の例

日本語の活用に対応することで，意味のまとまりを意識して文章を記述することが可能になる．例えばコード２の変数宣言が，コード５のような文の羅列だったとすると，短い文章が連続する不自然な文体になってしまう．普通，自然言語で文章を作るときには意味のまとまりを形作り，複数の文を読点で連結して文章として記述する．コード２の場合，変数を作ることと，初期値の代入は意味的にまとまりを作っていると考えられる．「言霊」では連用形を用いて記述できるため，この２つの文をまとめて１つの文章として記述することが可能なのである．

活用語尾の採用は，意味のまとまりを意識して読みやすい文章を作る上で必要な言語の機能である．活用語尾の採用が読みやすいコードを記述させ，それがコードの可読性につながるのである．

2. 4. 文脈が利用できる

「言霊」では文脈を用いた省略表現が可能である．文脈も自然な日本語表現を記述するためには不可欠な文法要素である．

仮に「言霊」の文法に文脈が無かった場合，コード２の２行目は以下のコード６のような表現になる．

```
国語点数を作る．国語点数に４９を代入する。
```

コード 6 文脈が無かった場合の例

表現がどこか不自然で硬い表現になっているのは，文脈を用いた省略を用いていないからである．「国語点数」という変数を作成して，その変数に数値を代入することは前後の文脈から明らかである．だから普通に表現しようとするなら，２回目の「国語点数」は省略されるべきである．だがここでは省略していないために不自然な表現になってしまう．

省略する代わりに代名詞「それ」を使用することも出来る．「それ」を使った表現を，以下のコード７に示す．同様のアイデアは「なでしこ」でも実現されている．

```
国語点数を作り，それに４９を代入する。
```

コード 7 代名詞「それ」を使った例

なお，前ページ「略記をすることなく，正しい日本語表現を用いる」では略記法を批判したが，文脈を用いた省略と略記法は全く別の問題である．略記法は，プログラム熟練者が記述の労力を減らすために略記するのであり，略記によって初心者がプログラムの読解が出来なくなることを問題視している．文脈による省略は，省略をしないと却ってプログラムの読解がやりにくくなる表現が出てくるための文法である．プログラムを読みやすくするために，文脈による省略を行うのである．

2. 5.「言霊」の効果と問題点

日本語プログラミング言語「言霊」は，初心者を対象にプログラムの読解を容易にすることを最大の目的として作成された言語である．「言霊」を用いて慶應大学で４回のプログラミング授業を実施したところ，コードの読解が容易になり文法教育のコストが下がったことは実感できた．

だが一方で「言霊」は書きにくい言語である．日本語表現を使ったプログラムが記述出来ると言っても，全ての日本語表現を解釈できるわけではない．コンピュータを使っている以上は解釈可能な表現には限りがあり，人間を相手に話しかけるようにプログラムが記述できるわけではない．

日本語プログラミング言語の書きにくさは，日常言語における表現の幅は広いのに対して，日本語プログラミング言語の表現の幅は狭いことに起因する．例えば関数名として「現れる」という動詞を使うとき，代わりに「あらわれる」という平仮名表現を使ったり，「現われる」と異なる送り仮名を使ったりする．このよう

に慣れ親しんできた日本語が使えると，ついつい日常的な癖で様々な表現を使ってしまう．これらを全て適切に解釈することは，現在のコンピュータには難しい．

3.「ことだま on Squeak」

日本語プログラミング言語の書きにくさを補うためには構文エディタを持ったプログラミング環境を用意することが最適である．構文エディタとは，文法的に間違ったプログラムが書けないように設計されているエディタである．

構文エディタを持ち，かつプログラミング教育に最適な環境として近年注目を集めているのが，Alan Kayの提唱するSqueak[5]である．SqueakではSqueak eToys（以後，eToys）と呼ばれるビジュアルプログラミング環境があり，構文エディタを用いてコーディングを行うことが出来る．

我々はeToysの利点を生かしつつ，目指すべき初心者プログラミング環境として「ことだま on Squeak」を設計・実装した．その特徴を以下に３点述べる．

3. 1. 構造エディタ環境によるプログラム記述環境

「ことだま on Squeak」は命令タイルを貼り付ける“タイルスクリプティング”によってプログラミングすることで，「言霊」の時に生じた書きにくさを解決している．

タイルスクリプティングの模様を図１に示す．ユーザは命令タイルを貼り付けることでプログラムを記述していることが分かる．

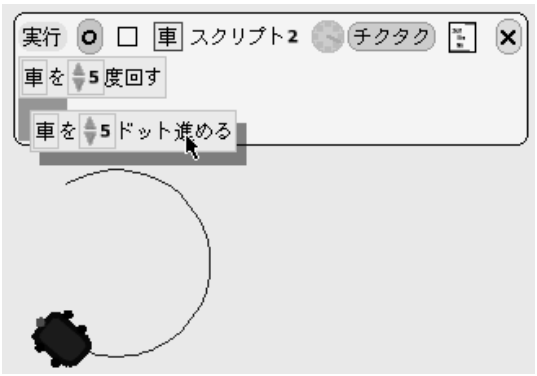

図 1 タイルスクリプティングの模様

タイルスクリプティング環境は，構造エディタを用いるために，２つのメリットが得られる．一つはユーザが自分で日本語を記述する必要がないため，２．５節で述べた日本語プログラミング言語特有の書きにくさは解決される．もう一つは，構造エディタであるためコンパイルエラーと無縁であり，初心者でも様々なコードを試行錯誤しながら記述することが可能になる点である．

3. 2. 日本語の語順と文法で記述されている

「ことだま on Squeak」は，正しい日本語の語順（動詞が末尾に来る）を採用している．図２は「ことだま on Squeak」における命令タイルの例である．「言霊」にの読みやすい表記を受け継いでいる．

図 2「ことだま on Squeak」における命令タイル

3. 3. 略記を避けて，読みやすい日本語表現を用いる

「ことだま on Squeak」は簡略化を一切排し，初心者がタイルを読むことで出来るだけ意味を分かるようなプログラム表現を採用している．**エラー! 参照元が見つかりません。**３に「ことだま on Squeak」における条件分岐タイルのプログラム表現を示す．

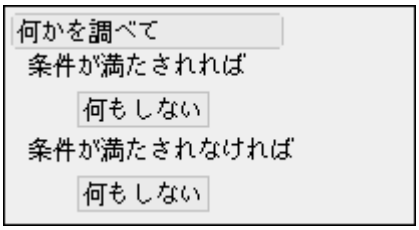

図 3「ことだま on Squeak」における条件分岐タイル

「ことだま on Squeak」は，プログラム初心者が読んでも意味を理解できるようなプログラム表現を採用している．図３の条件分岐タイルを読めば，“とにかく何かを調べる”ためのタイルであることが理解できる．

「ことだま on Squeak」における条件分岐タイルの読みやすさは，条件節に調べるべきタイルを入れたときに，その効果が発揮される．図４は色判定タイルを条件節に入れたときのプログラム表現である．

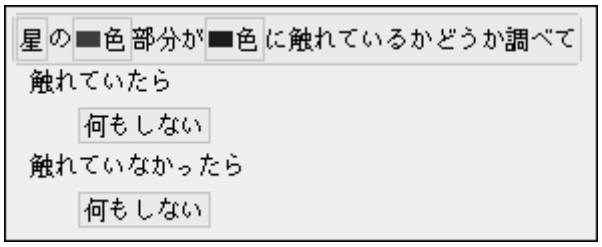

図 4「ことだま on Squeak」における
色判定タイルを条件節に入れたときの
条件分岐タイル

「ことだま on Squeak」の条件分岐タイルは条件をセットすると，条件分岐タイルの２行目と４行目において命令タイルの実行条件を細かく解説する．図４の例では「触れていたら」「触れていなかったら」という表現になっている．

3. 4.「ことだま on Squeak」の効果と問題点

「ことだま on Squeak」は，日本語プログラミング言語「言霊」が持つコードの可読性を受け継ぎつつ Squeak-eToy の構造エディタなどのプログラミング環境を利用することで，読みやすく書きやすい日本語プログラミング言語を目指して開発された．

「ことだま on Squeak」を用いた実験授業を神奈川県藤沢市立村岡中学校で実施した．その結果，既存の Squeakを用いて行われた授業と比較して，文法の質問が減ったことが確認された．ちゃんとした日本語でタイルが記述されているため，受講者は特に質問をしなくてもタイルの意味を理解できるのである．その結果，受講者は本質的なアルゴリズムに関する議論に集中することが可能になった．

だが，一方で「ことだま on Squeak」の読みやすさ，書きやすさは初心者を対象にしたときの話しである．少しでもプログラムが出来る人にとっては，「ことだま on Squeak」の日本語表記は長すぎて，本質的な部分を読み取るのに煩わしい要素が多い．またタイルスクリプティング環境も，初心者にとってはミスが起こらないために記述しやすいが，ミスを自分で解決する能力がある熟練者にとっては操作が面倒な環境である．

「ことだま on Squeak」は初心者を対象としてプログラムの読みやすさ・書きやすさを追求したが，中級者以上からは使いやすい環境とは言えないのである．

4. 教育用プログラミング環境として
将来「ことだま on Squeak」の持つべき特徴

「ことだま on Squeak」を，初級者だけではなく中上級者を対象として活用するためには，プログラムの読解・記述環境の切り替えを可能にするべきである．プログラミング学習者の技術レベルや環境に対する好みは様々であり，学習環境も様々な学習者に対応することが望ましい．

4. 1. 受講者レベルに応じたコード読解環境

プログラムは読みやすくあるべきだが，読みやすさは受講者のレベルによって異なる．我々は初心者にとっての読みやすさを追求するために日本語プログラミング言語を開発してきたが，この読みやすさはプログラムの概念を全く理解していない初心者が対象である．一方でプログラムの概念を身につけた中上級者にとっては本質的でない記述は邪魔でしかない．

プログラミング学習者も，最初は初心者であるが，学習と経験を重ねることでいずれ中・上級者へと変わっていく．その時にプログラミング学習環境も，学習者のレベルに応じた環境を提供するべきである．

初心者にとってコードの見た目は，未学習の内容があってもコードの大意が掴めるような文章になっていることが重要である．図５（現在の「ことだま on Squeak」と同じ）のように，コードには多くの説明的な言葉が入り，学習者の理解を助けるようなビューになることが望ましい．

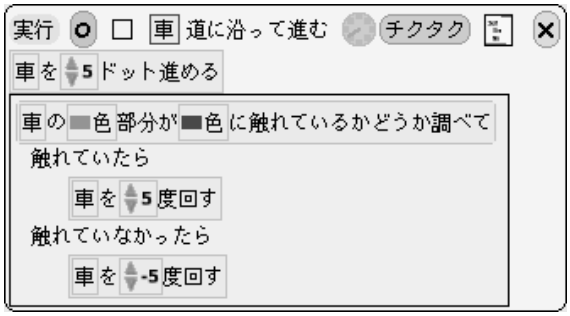

図 5 説明的なコードビュー

一方，中上級者にとってコードの見た目は，プログラミングの本質に無関係な記述はなるべく廃した表現になることが重要である．図６（詳細な文法などは検討中である）のように，図５と比べると簡潔な表現になる．2. 2節で述べた簡略表現も取り入れるべきであろう．

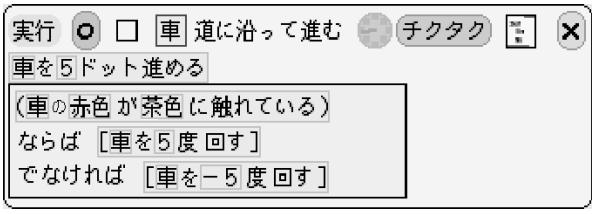

図 6 簡潔なコードビュー

4. 2. 受講者レベルに応じたコード記述環境

プログラムの記述環境も，受講者のレベルによって異なる．

初心者がプログラミングを学習する際に一番問題になるのが，コンパイルエラーである．初心者はコンパイルエラーの対処方法を知らないため，エラー文に遭遇すると無力感と共にプログラムを学ぶ気力を失ってしまう．そのため「ことだま on Squeak」はeToyの構造エディタを用いることでコンパイルエラーが生じない図７のような環境を提供している．

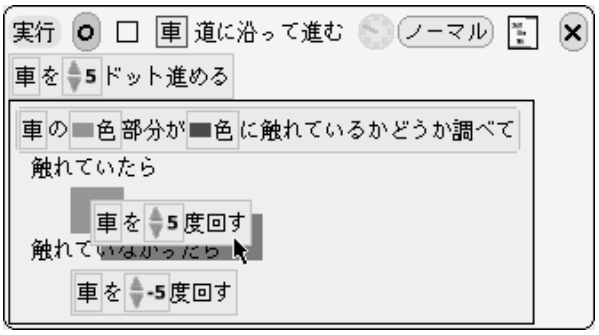

図 7 初心者のためのタイルスクリプティング環境

一方，中上級者にとってeToyの構造エディタによるプログラミングは煩わしさを感じてしまう．マウスによるタイルの貼り付けは，キーボードによる直接入力と比べると記述時間が長くなってしまう．また中上級者はコンパイルエラーへの対処方法を知っているため，コンパイルエラーを避けるための構造エディタを用いる必要性もない．そこで図８のように，キーボードからコードを直接入力するような環境を提供するべきである．

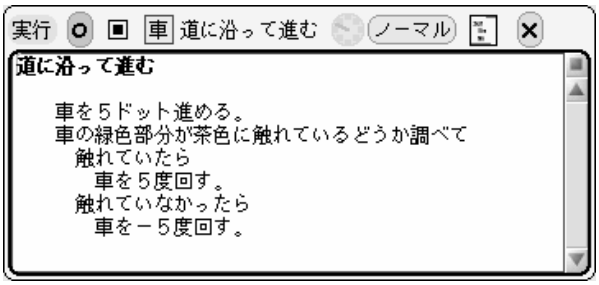

図 8 中上級者のためのプログラミング記述環境

もちろん中上級者でタイルスクリプティングを好む人は，図７の環境を用いればいい．記述環境も読解環境も，個人の好みで使い分けるべきである．

5. まとめ

我々は初心者に対するプログラミング環境を模索し，日本語プログラミング言語「言霊」と，「ことだま on Squeak」を開発した．その知見を生かして，初心者だけではなく様々な習熟度の学習者が使用できるようなプログラミング学習環境を提案した．ここで述べた環境は提案に過ぎないので，実際に実装して実証実験を行うつもりである．

参考文献

[1] 岡田健，中鉢欣秀，鈴木弘，大岩元(2002) 日本プログラム言語「言霊」，情報処理学会2002FIT

[2] 岡田健，大岩元(2002) プログラミング言語としての日本語 慶應義塾大学湘南藤沢学会．Keio SFC Journal, Vol.2 No.1 pp114-134

[3] 岡田健, 大岩元(2005)日本語プログラム言語「言霊」におけるメソッドの記述方法．情報処理学会第４６回プログラム・シンポジウム

[4] 日本語プログラミング言語「なでしこ」 http://nadesi.com/

[5] Dan Ingalls, Ted Kaehlei, John Maloney, Scott wallace, and Alan Kay (1997). Backto to the Future: The Story of Squeak, A Practical Smalltalk Written in Itself, Proc. of ACM OOPSLA '97, pp.318