Osaka Gakuin University Repository

Title	アニメーションプログラミング環境 Alice の日本語化とその教育的意義 Japanese Localization of Alice: a 3D Animation Programming Environment, and Consideration of its Educational Significance
Author(s)	淡　誠一郎 (Seiichiro Dan)
Citation	大阪学院大学　人文自然論叢（THE BULLETIN OF THE CULTURAL AND NATURAL SCIENCES IN OSAKA GAKUIN UNIVERSITY), 61-62：1-22
Issue Date	2011.03.31
Resource Type	Article/ 論説
Resource Version	
URL	
Right	
Additional Information	

大阪学院大学　人文自然論叢 ＜論　説＞
第61-62号　2011年３月

アニメーションプログラミング環境 Alice の日本語化とその教育的意義

淡　　誠一郎

Japanese Localization of Alice: a 3D Animation Programming Environment, and Consideration of its Educational Significance

Seiichiro Dan

Abstract

Alice is a 3D animation programming environment. Alice project, which was started at the University of Virginia, has been handed to Carnegie Mellon University and it is still under development.Though Alice was initially developed as a rapid prototyping system for virtual reality software, it is now regarded as an educational programming environment and has been used in many educational institutions. Alice can keep the novice programmers' motivation high by attracting them with its animation capability and reducing the complexity of details that they must overcome in programming. The current version of Alice supports only English. In Japan, this fact significantly reduces Alice's educational value, because many Japanese students tend to avoid English materials. Japanese localization must be necessary to spread Alice in Japan. We have developed a Japanese localized version of Alice and used it in some CS classes at our University and in summer seminars for Japanese junior high school students. This paper describes the Japanese localization and discusses the Alice's potential for the education of the computer science in Japan.

概要

Aliceは3Dアニメーションのプログラミング環境である。その研究プロジェクトはバージニア大学で始まり、カーネギメロン大学へと引き継がれて今も開発は継続している。当初は簡易VR（仮想現実）環境という位置づけで、VRソフトウェアの高速プロトタイピングが目的であったが、最近は教育用環境としての色彩が濃くなり、世界中の多くの教育機関で利用されている。Aliceの開発コンセプトは、プログラミング初心者の学習意欲低下をまねく要因を極力そぎ落とし、アニメーションという魅力的で理解しやすい題材により学習者のモチベーションを持続させ、成功体験を与えるということである。現バージョンのAliceは残念ながら英語のみにしか対応していない。日本人学習者にとっては英語システムであるということ自体が大きな障壁である。幸いAliceはオープンソースソフトウェアであり、自由な改変が許されている。そこで、ソースコードを解析し、かなりの部分を日本語化できたので、ここに報告する。このシステムは、本学（大阪学院大学）のゼミナールや中学生のサマーセミナー等で活用しており、日本語化によって我が国においても幅広い年齢層の生徒・学生に受け入れられることが実践的に確認できた。

はじめに

情報技術の発展に伴い、情報系の卒業生に要求される知識は増加・高度化する一方である。たとえば、オブジェクト指向プログラミングやイベント駆動プログラミングなどは、かつては大学院で初めて扱われたような内容であるが、今や学部レベルのリテラシと見なされている。しかし、現実には「オブジェクト指向」などの高度な抽象概念をプログラミング経験の浅い学習者に正しく理解させるのは容易ではない。学習内容が高度化する一方で、プログラミング学習に割ける時間は増えておらず、多くの学生が落ちこぼれているという現実がある。かといって、古典的な学習内容のままでいては、GUIに深く馴染み、高度なCGで目の肥えた今の学生には魅力が乏しく、興味をつなぎ止められない。このようなジレンマはプログラミングに限らず、コンピュータサイエンスの教育現場全体に広がっている。

こういった事情は我が国だけのものではない。UCLA大学教育センターの調査によれば2000年から2004年にかけて、コンピュータサイエンスを専攻する入学者の割合は6割も減少し、1980年代初頭よりも低下していると報告されている。この問題を契機として、セントジョセフ大学の研究チームは、カーネギメロン大学で開発されてきた3Dアニメーションプログラミング環境Aliceに注目し、その教育効果について調査研究を行っている。その結果、特に通常授業では学習意欲の低かった、いわゆる落ちこぼれ学生層が積極的に学習に取り組み、さらなる発展クラスの受講率も高まるなど、Aliceを用いたプログラミング教育は従来型のそれと比べて顕著な効果が認められている［1，2］。

2010年３月現在、Alice は世界80カ国以上、延べ2000以上の教育機関で利用されている（表１）。にもかかわらず、日本の教育機関の登録は、名桜大学（沖縄）、大阪学院大学など延べ６校が確認できるに過ぎない（上記２校以外は名称不明）。ちなみに、６校という数字は北朝鮮の登録数と同じである。日本での Alice の利用が極端に少ないことの背景には、Alice に日本語版がないという事情が少なからず影響を与えているものと推測できる。Alice の教育効果は、アニメーションを題材としていることに加えて、プログラミングの本質から離れたところで学習者の障害となる諸要因を極力排除した支援環境に依るところが大である。しかるに、多くの日本人学生にとっては、英語で記述されていること自体が大きな障害要因であり、学習者のモチベーションを著しく低下させ、教育効果を損なうことは想像に難くない。

表１　Alice を利用している教育機関数

順位	国名	登録教育機関数
1	USA	1492
2	Canada	93
3	United Kingdom	63
4	Mexico	49
5	Australia	46
6	Brazil	33
7	Philippines	28
8	China	19
8	New Zealand	18
10	Netherlands	17
−	Others	339
	合計	2197

※　公式ホームページ［３］に基づいて集計した。2010年４月18日現在。大学600、高校968、その他629（いずれも延べ数）が利用登録している。なお、利用・登録に義務はない。

1. 簡易 VR システムとしての Alice

Alice はカーネギメロン大学で開発中の、３D アニメーションを題材としたプログラミングの教育環境である。Alice プロジェクトは、中心的開発者の一人であった Randy Pausch（2008年没）がバージニア大学在職中に始めたもので、もともとはバーチャルリアリティ（VR）の高速プロトタイピング環境という位置づけであった［４］。Alice はヘッドマウントディスプレイやデータグローブを用いる環境埋没型の本格的な VR とは異なり、ほぼ同時期に同様の趣旨で仕様策定が始められた VRML（Virtual Reality Modeling Language）などと同様、一般的なパーソナルコンピュータシステムだけで成立する簡易型

の VR 環境である［5］。

VRML と Alice の大きな違いは、VRML が仮想世界のモデリングに重点をおいて開発されてきたのに対し、Alice は仮想世界におけるオブジェクトの振る舞い、つまりアニメーションに重点がおかれている点、そして VRML が言語仕様にすぎないのに対し、Alice は実環境であるという点である。VRML も仕様2.0（VRML97）［6］で動きの記述ができるよう拡張されたが、モデリングが中心であることに変わりはない。逆に Alice は仮想世界の構成部品を新たに定義して追加するような機能を持たない（唯一の例外は「人物」であり、バージョン2.2や現在開発段階にあるバージョン３では、人物のジェネレータが実装されている）。アニメーションを楽しむには当然３D モデルが不可欠であるが、Alice にはオブジェクトギャラリーと呼ばれる、３D 部品（背景、建物、家具、人、動物、乗り物、小物など）のデータベースが備わっており、面倒な３D モデリングなしにすぐにアニメーションのプログラミングを開始できる［7］。

同様の設計思想は、NHK が開発したテレビ番組制作システム TVML［8，9］にも見出せる。スタジオセットやキャストの部品があらかじめ用意されており、仮想世界でカメラマンやキャストを自由に動かそうというというアイデアは非常に似通っている。しかし、TVML のプログラムは文字通りの台本にすぎない。つまり時間に沿った動作の系列を記述できるのみである。これに対し Alice には Java 言語と類似した独自のオブジェクト指向言語が用意されており、プログラミングの自由度が高い。また、TVML は Player や Creator などの支援環境がバイナリ公開されているものの、基本的には VRML 同様言語仕様に過ぎないという違いもある。

VRML や TVML とは違い、Alice はコードエディタと実行環境を備えた統合環境である。Alice の３D モデルやプログラミング言語は独自仕様であり、現状では Alice 以外で利用することはできない。しかし、Alice をプログラミングの導入教育用の環境としてとらえるならば、このことは大きな欠点とはならない。

Alice はコンピュータグラフィックスの専門知識がない人にも、３D のアニメドラマを創作したり、仮想世界をデザインしてその世界を自由にウォークスルーするプログラムを開発したりする楽しみを提供してくれる。また、イベント駆動で動く仕掛けを簡単にプログラムすることもできるので、インタラクティブな３D ゲームを作成することもできる。これは今からプログラミングを学ぼうとする若者にはとても魅力的な特徴である。

2. 教育用プログラミング環境としての Alice

まず、プログラミングの教育において初心者の教育／学習を妨げる要因を挙げて説明し、各要因に対して Alice に期待される効果について考察する。

プログラミング導入教育における障害要素

プログラミング学習において、初心者の学習を妨げる要因には次のような事柄がある。

（1）構文エラーに代表されるプログラミング過程の落とし穴

タイプミスや予約語の覚え間違いなどによって思惑通りの結果が得られず、退屈なバグ取り作業に時間を消費してしまう。括弧のつけ間違いによるエラー体験が構造化という概念の理解につながることは稀で、煩わしさの印象しか残らない。こういった些細なミスはフラストレーションを高め、学習意欲をそぐ要因でしかない。前の課題ができないうちに次の説明が始まってしまうと、授業についていけなくなり、その時点で学習を継続する意欲が失せてしまう。初心者には失敗体験より成功体験を与えるべきである。

（2）プログラムの実行過程の不透明性

普通、プログラムの実行過程は目に見えず、想定通りの計算が実行されているという実感を持てない。また、結果が違っていても、結果からコードの間違いを推定するのは難しい。デバッガ等でテキストベースのトレースを行うことはできるが、初心者には覚えるべきことがらが増え、難解さを印象付け逆効果となることも多い。

（3）魅力のない例題、達成感の欠如

多くの学生は必須であるからという単純な理由でプログラミングの入門コースを受講しており、そのモチベーションは低い。そのような学生にやる気をおこさせるには魅力ある題材が必要である。「Hello World」に始まり、ソーティングでクライマックスを迎えるといった古典的な内容には、グラフィカルなソフトウェアを見慣れた最近の学生を惹きつけ、満足させる力はない。また、多くの学生は学習内容と、見慣れたソフトウェアの間に絶望的ともいえる距離を感じ、先へ進むことを断念してしまう。

（4）「抽象化」のとらえどころのなさ

プログラミング導入教育の一つのゴールは、学習者に「データ抽象化」、「制御抽象化」、そしてそれらを合わせた概念ともいえる「オブジェクト指向」という考え方を身に着けさせることである。抽象化を抽象的な言葉で説明しても初心者には理解できない。よい例題が必要であるが、簡単すぎる例題ではその意義が実感できないし、実感できるような例題は大規模で難しすぎる。

（5）アルゴリズムの設計技術の教授の難しさ

プログラミング言語の構文を理解し、既存のアルゴリズムの実装と動作確認ができたからといって、自分でプログラムを設計・実装できるようにはならない。構文や例題とその解決アルゴリズムを与えるのは簡単だが、与えられた問題に対しその解決アルゴリズムを学習者が自ら見つけ出せるように導くのはずっと難しい。

Alice の有効性

上にあげた各問題点に対する、Alice システムの有効性を挙げていく。

（1）GUI 操作の構造化エディタによる構文エラーからの解放

Alice のコードエディタはいわゆる構造化エディタであり、構文的に間違った文を入力できないように作られている。つまり、入力が完了した時点で構文的にはエラーフリーであり、構文エラーでフラストレーションが高まるという事態は起こらない。また、入力はマウスによるドラッグ＆ドロップとコンテキストメニューからの選択が中心であり、ポップアップメニューには正しい選択肢しか表示されないし、ドラッグ＆ドロップ時は構文的に正しい場所にしかドロップできない。また、ブロックは括弧の対応ではなく、明示的に色分けされたタイルで表現され、入れ子構造が一目瞭然である。

このような環境では過ちに基づく学習の効果が発生せず、構文的知識が身に付かないという指摘もある。しかし、構文エラーの訂正では、一つの誤り例とそれに対する一つの正事例しか学習できないのに対し、Alice ではプログラム要素をドラッグしたときに文法的に正しい落としどころとそうでないところが、ハイライト表示の有無ですべて明示的に示されるので、むしろ提供される情報量は多いという見方もできる。

（2）実行過程の透明性

アニメーションではプログラムの実行状態が文字通り見える。たとえば、符号の入力間違いはアニメーションではオブジェクトの進行方向の違いというような形で表れるし、停止条件の誤りはオブジェクトが目標地点に到達しないとか行き過ぎてしまうということですぐ気付くことができる。物理法則をシミュレーションしようとしているときに、データの数値的変化を読み取って間違いに気づくのは難しいが、目で見て動きが自然でないと知覚するのは容易い。Alice では、抽象化された数学的な表現ではなく、「前進」、「後退」、「右回り」、「左回り」、「～に顔を向ける」、「～に近づく」などといった身近な表現に対応したメソッドが用意され、その組み合わせでオブジェクトの動きをプログラムする。メソッドの意味と視覚的なイメージが直結しており、実行過程を見ればそれがプログラムのどの部分で引き起こされているのかを見出すのも容易い。

（3）アニメーションという魅力的な題材

アニメーションは楽しく解りやすい。単なる数値計算やデータ処理よりアニメーションの方が多くの学習者にとって魅力的であるという事実には議論の余地はないだろう。

学習者のモチベーションを高めるためには達成感や向上感が不可欠である。自分

の作成しているプログラムの意味が理解でき、その効果が目に見える形で確認できるということは、プログラミングの学習を続けようという意欲に直結する。コードの効果が実感でき、想像できると、次にはもっと複雑な機能を自分で作り出してみたいという意欲が湧き上がってくる。

（4）オブジェクト指向とアニメーションの親和性

アニメーションは近代的なプログラミングの主流をなしているオブジェクト指向と親和性が高く、この難解な概念を説明するのに最適な題材である。しかし、一方で、高度な数学とグラフィックライブラリの知識が必要であるため、通常初心者向けのプログラミング教育でとり上げるには無理がある。これに対し、Alice は（2）で述べたように身近な言葉に対応したメソッドがあらかじめ用意されているし、高度な数学的知識が要求される描画機能は環境に埋め込まれ隠ぺいされているので、初心者の障害とならない。

（5）ドラマ制作プロセスからの類推によるアルゴリズム設計プロセスの自然な理解

初心者に「処理の流れ」や「逐次実行」という、プログラムの基本概念を理解させるのは意外に難しい。アニメーションであれば映画やドラマ制作と対応させて説明できるので、「処理の流れ」や「逐次実行」、さらには「並列実行」という概念も説明しやすく、受け入れられ易い。アルゴリズムの設計はシナリオやシーンの設計、演出という言葉で説明でき、プログラム作りとは何なのかということで学生が悩むことはない。学習者は、自らシナリオを発展させたり、視覚効果を追加・工夫したりとあれこれ試行錯誤する。それがすなわちアルゴリズムの設計のプロセスである。

以上、プログラミング導入教育における Alice のメリットを挙げてきたが、システムで使われている言語が母国語でないと、せっかくの期待される効果が半減してしまう。幸い Alice はいわゆるオープンソースプログラムであり、日本語版を独自制作可能である。次章では、Alice の機能を説明しながら、その日本語化について述べていく。

なお、Alice は Java 言語を用いてマルチプラットフォームで開発されており、Windows 版、MacOSX 版、Linux 版がそれぞれ存在し、現在2.0, 2.2, 3.0ベータのバイナリ、2.0, 2.2 のソースが公開されている。本稿では、このうち Windows 版バージョン2.0の日本語化について述べる。（開発開始当時公開されていたのが2.0であったためであるが、2010年３月現在のバージョン2.2にはマルチバイト文字が表示されない不具合がある。）

3. Alice システムの日本語化

3.1 日本語環境における既知の問題

現在公式リリースされている Alice 2.0, 2.2は、日本語 Windows 環境で次のような不具合が確認できている。Alice が英語システムであるということに加えて、日本において Alice があまり普及しない原因であると考えられる。

（1）日本語パス名が使えない問題（2.0）

Alice はインストール不要で、展開するだけで使えるソフトウェアであるが、「デスクトップ」に展開すると起動すらしない。

（2）デフォルトの保存ディレクトリの問題（2.0, 2.2）

Alice の各種デフォルトディレクトリは、

ユーザのホーム￥Desktop

となっている。ところが日本語版 Windows Xp では「デスクトップ」という仮名表記が採用されており、Desktop というフォルダ自体が存在しない（自分で作成すれば別）。このため、なにか設定を変えようとするたびに毎回警告ダイアログがポップアップする。一方、Vista 以降では「デスクトップ」のターゲットは日本語環境でも Desktop となっているため不都合は生じない。Xp に合わせてソースを改変し、デフォルト保存フォルダを単純に「デスクトップ」にしてしまうと、Vista 以降の環境と整合性がとれない。

（3）変数名、メソッド名、パラメータ名等に日本語が使えない問題（2.0, 2.2）

Alice は文法的な誤りは入力時に排除するように設計されている。２バイト文字は設計上不正文字であり、入力しようとした時点で赤字表示され、受け付けられない。変数名などにマルチバイト文字が使えないことはプログラミング言語としては特殊事情ではないが、そう訓練されていない日本人学習者はたいてい日本語を使おうとする。初心者の障害となる要因を極力排除するという Alice の設計方針に従うならば、使用文字に対する制限はない方がよい。

（4）日本語オブジェクト名を使用したファイルが読み込めない問題（2.0, 2.2）

オブジェクト名や部品名は少し事情が異なる。変数名やメソッドとは違って、入力時に日本語入力が拒否されるということはない。特にソースレベルの改造をせずとも、オブジェクトやその部品の名前を日本語にリネームすることが可能であり、実行もできる。一見問題なく動作するが、完成したプログラムを一旦保存して次に開こうとすると読み込めない。読み込みルーチンがマルチバイト文字を想定していないためである。せっかく作ったプログラムが開けず、一から作り直しになってしまう。

（5）日本語文字が文字化けする（2.2）

Alice は基本的にマルチバイトに対応していないが、３D テキストや print 文、コ

メントなどに日本語文字を入力することは可能である。ところが、バージョンによって日本語文字が内蔵エディタやコンソールでも表示されないことがあり、その事情がバージョンが変わるたびに変動し安定しない。

3.2 日本語化の基本方針

Alice にはグラフィカルなアイコンが豊富に使われており、操作も直感的で分かりやすい。Alice で作るプログラムはアニメーションに限定されているので、英語が理解できなくても試せば意味は一目瞭然という部分も多い。一通りの機能、組み込みメソッド等を試して意味を掴んでしまえば、英語表記は実はあまり大きな障害ではない。問題は導入初期である。

日本人にはプログラミングやアニメーションに興味があっても、英語に対して苦手意識や抵抗感をもつ人が多い。そういう人は入り口で拒絶反応を起こしてしまいがちである。日本語化の目的は、その最初の拒絶反応の要因を取り除くことであり、隅から隅まで無理に日本語化する必要はない。自然に目に入るメニューや、頻繁に使用される基本メソッドを日本語化すれば目的は達成される。

めったに使わない機能は無理に日本語化する必要はない。下手にわかりにくい日本語訳をつけるより、あえて英語のままで残す方がよい場合もある。英語と日本語が混在する場合、英語の苦手な日本人の目は、日本語化してある部分にのみ向けられるので、多すぎる機能に戸惑い圧倒されるという別の拒絶感を和らげる効果が期待できるからである。

3.3 Alice の概要とその日本語化

以下、Alice の画面構成や機能を紹介しつつ、その日本語化の概要を説明していく。まず、システムの主要機能について、その後でチュートリアル機能について説明する。

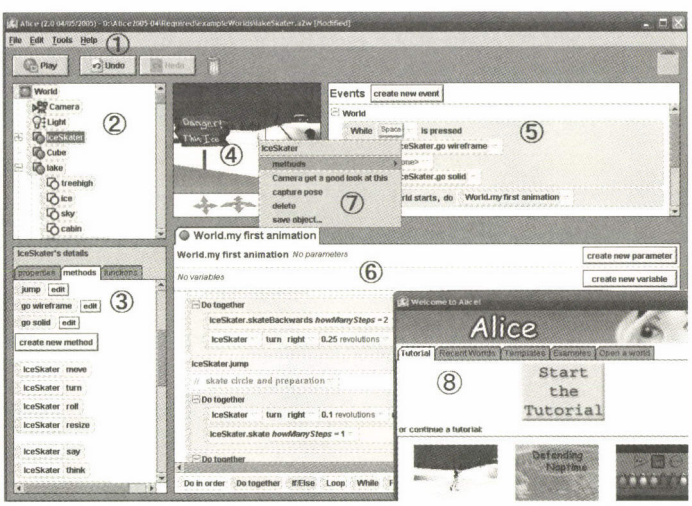


図 1　Alice の画面構成（英語版）

主要機能の日本語化

Alice のメインウィンドウを図１に示す。以下、各エリアの働きを簡単に解説しながら、日本語化の概要を説明する。

①メインメニュー

メニューはソフトウェアの印象を大きく左右するので、プルダウンで表れる部分も含め、後述する一部を除き極力日本語化した （図２）。メニュー項目やボタンのテキストはソース内に直接埋め込まれている。該当文字を見つけ、日本語に置換するだけで日本語化できる（以下、同様の方法で日本語化できる要素は「**置換**」と記す）。

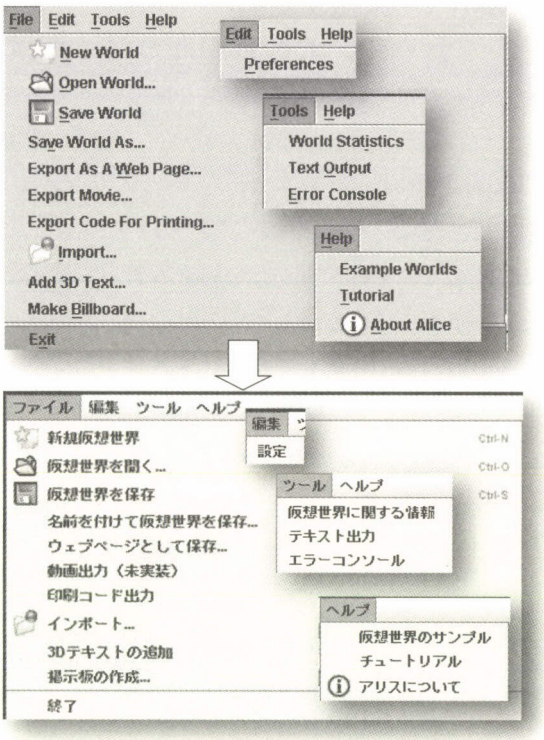


図２　メインメニューの日本語化

②オブジェクトツリーエリア

シーンの階層構造をツリー表示するエリア。この部分は文字コードの種類に依存するような処理はない。表示される内容は仮想世界を構成する部品の名称なので、オブジェクト名に日本語が使えるようになれば、このエリアは日本語化できたことになる。

③詳細エリア

オブジェクトツリー上でフォーカスされたオブジェクトの詳細情報が表示されるエリア。次の３つのタブから成る。

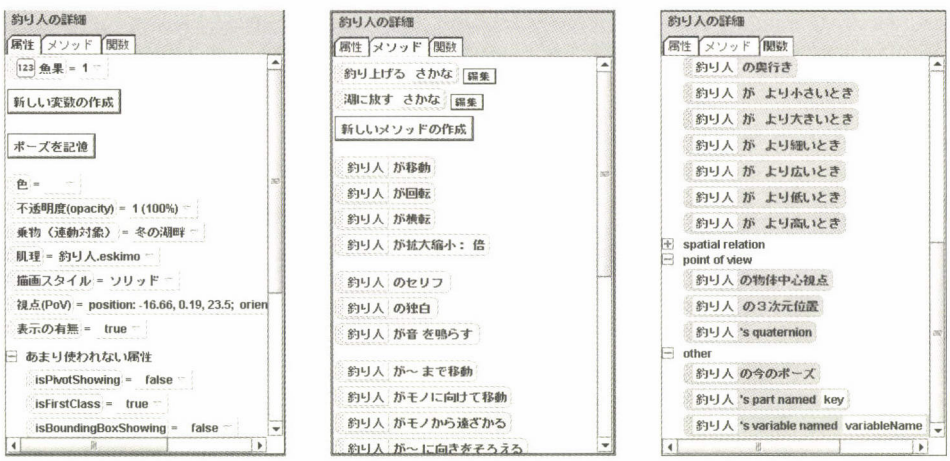
・属性（**properties**） － 色、透明度、テキスチャ、連動対象、位置など。ユーザが新しい属性（変数）を定義して使うこともできる。

・メソッド（**methods**） － 移動、回転、拡大、属性変化など。一連の動作や振舞いをユーザが定義して使うこともできる。

・関数（**functions**） － そのオブジェクトで値の決まる関数。大きさ、相対距離など。ユーザが新しい関数を定義して使うこともできる。

Alice にはあらかじめ多数の３D 部品が用意されているが、それらはオブジェクト指向言語における組み込みクラスに相当する。そのほとんどは３次元物体であり、共通の属性、メソッド、関数を持つ。また、特別なクラスとして World（仮想世界全体）、Camera（カメラ）、Light（光源）があり、一般のクラスが持たない属性やメソッド、関数を持っている。

組み込みで用意されている属性、メソッド、関数のうち、あまり使われない項目や日本語表現しにくい一部の項目を除き、大半を日本語化した（**置換**）。また、メソッド名や関数を新規作成する際に日本語が使えるようにソースを改良した（以下、「**改変**」と記す）。図３に日本語化後の詳細エリアの例を示す。



「属性」タブ　「メソッド」タブ　「関数」タブ

図３　詳細エリアの３つのタブ（日本語化後）

④ワールドビューエリア

３D シーンの視覚イメージを表示するエリア。マウス操作で直接オブジェクトの配置や視点を変更できる。詳細編集モードへの切り替えで、オブジェクトギャラリー（既成３D 部品のデータベース）から部品を呼び出してシーンに追加し、たくさんのオブジェクトから構成された複雑なシーンを簡単に作れる（図４）。シーン編集モードでのボタン、右クリックで表示されるポップアップメニュー等を日本語表記にした（**置換**）。

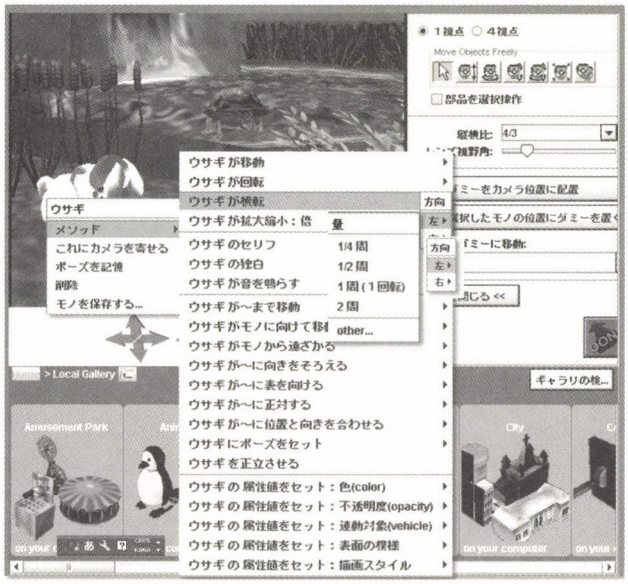


図４　ワールドビューエリアの日本語化（詳細編集モード）

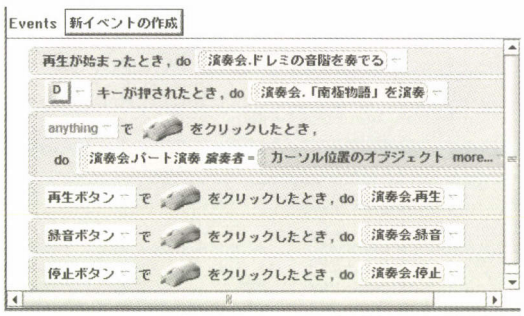


図５　イベントエリアの日本語化

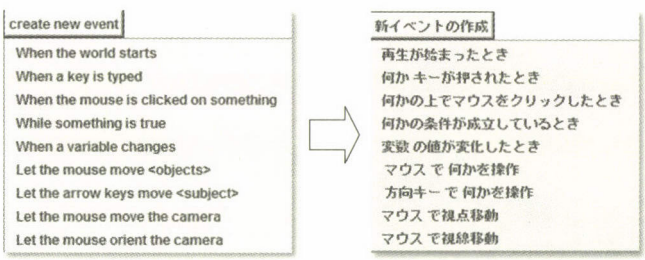


図６　イベントエリアのプルダウンメニューの日本語化

⑤イベントエリア

イベントとそれによって駆動されるオブジェクトの振る舞いを定義するエリア。プルダウンメニューとイベント部品を日本語表記にした（**置換**）。（図５，６）

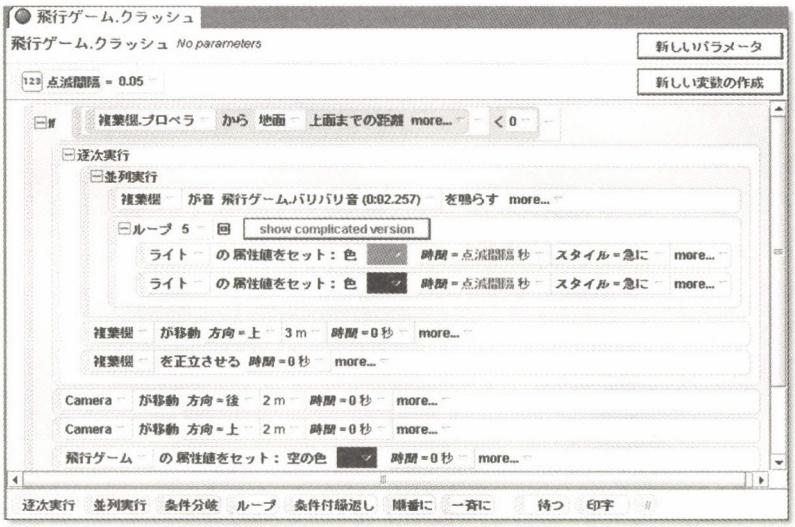


図７　コードエディタの日本語化

⑥コードエディタ

メソッドのプログラムコードを記述するエリア。図７にプログラムの作成例を示す。画面下に配置された制御部品（ブロック、条件分岐、ループ等）や、オブジェクトツリー、詳細エリアからオブジェクトやメソッド、関数をマウスでドラッグし、このエリアにドロップすることでプログラムを作成できる。行の入れ替え、削除、複製、パラメータの選択などもマウス操作で行う。このエディタはいわゆる構造化エディタであり、文法的に正しい場所にしか部品をドロップできない。部品をドラッグすると、正しい落とし所がハイライトされる。ポップアップメニューには適切な項目しか表示されないように作られている。

制御部品を表２に示すように日本語化し、ポップアップメニューを日本語表記にした（**置換**）。また、変数やパラメータに日本語が使用できるようソースを修正した（**改変**）。

表２　制御部品の英日対応

Do in order	逐次実行
Do together	並列実行
IF/ELSE	条件分岐
Loop	ループ
While	条件付繰返し
For all in order	順番に
For all together	一斉に
Wait	待つ
Print	印字

⑦コンテキストメニュー

オブジェクトツリーとワールドビューエリア内でオブジェクトを右クリックすると対象に応じた内容のメニューがポップアップ表示される。このメニューを日本語化した（**置換**）。図４で表示されているメニューは「ウサギ」に関連付けられて表示されるポップアップメニューである。オブジェクトツリー上で「ウサギ」を右クリックした場合にも同じメニューが表示される。

⑧入出力ダイアログ

入出力ダイアログの表記を日本語化した（**置換**）。特に図１の⑧に示すダイアログはAlice 起動時に毎回表示され、チュートリアルや各種サンプル、最近使ったファイルへのポータルとなっている。本質的な部分ではないがソフトウェアの印象を大きく左右する。また、ファイル読み込みのルーチンをファイル名だけでなく、内包するオブジェクト名、メソッド名、関数名、変数・パラメータ名が日本語である場合にも対応できるよう改良した（**改変**）。

⑨その他のダイアログとポップアップ・ヘルプ

学習者に指示を与える、あるいは情報の入力を促すためのポップアップ・ダイアログ、マウスポインタのある位置に現れるポップアップ・ヘルプは、単純にソース内の該当テキストを日本語に置き換えれば日本語化できる（**置換**）。ダイアログとポップアップ・ヘルプの例を図８に示す。

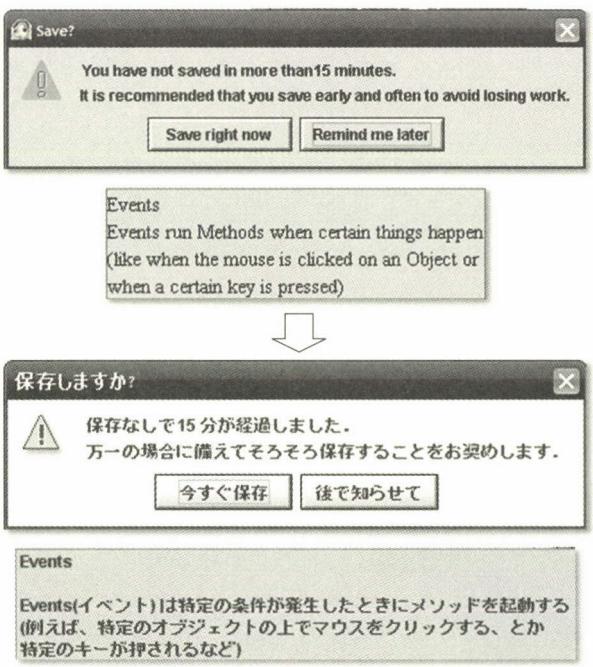


図８　ポップアップ・ダイアログとポップアップ・ヘルプの例

ここまでに説明した日本語化実施後の画面スナップショットを図９に示す。

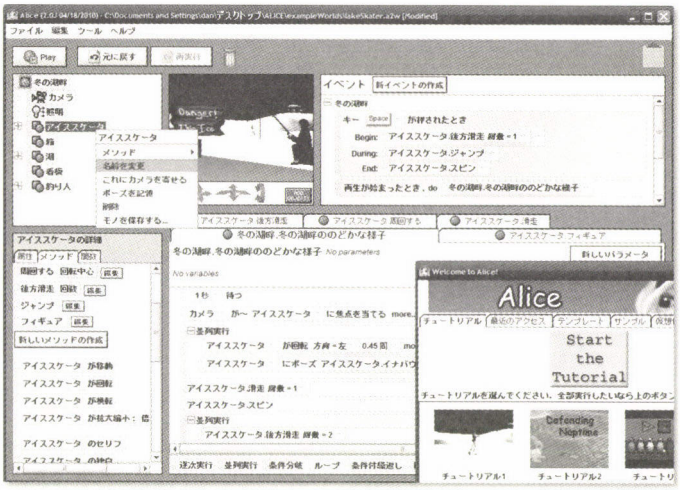


図９　日本語化 Alice

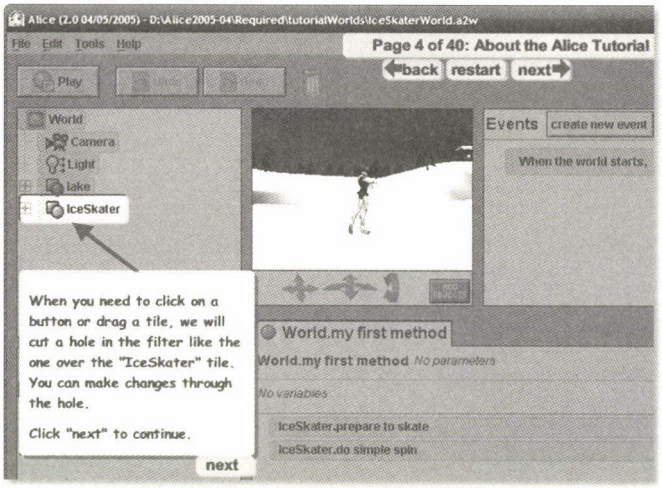


図10　チュートリアル（英語版）

チュートリアルの日本語化

Alice にはシステムの概要を紹介する４つの導入チュートリアルが用意されている（表３）。このチュートリアルは教育用プログラミング学習システムとしての Alice のセールスポイントの一つである。単なる動画やハイパーテキストによる機能紹介などとは異なり、実際の Alice の操作画面に解説やインストラクションが重畳表示される。インストラクションに従って Alice を操作していけば、基本的な使い方は理解、マスターできるよう設計されている。Alice そのものを使わせながらも、操作対象以外はマスク処理され、学習者

が迷わないような親切設計となっている（図10、図11）。Alice による教育プログラムは、このチュートリアルから始まるといってよい。

メニューやボタンのような単語レベルではなく、文章レベルの英語力が必要となるので、英語の苦手な日本人学習者には、このチュートリアルは役に立たない。

表３　Alice のチュートリアル

	内　容	ステップ数
チュートリアル１（クイックツアー）	チュートリアルの操作方法、Alice の画面構成と各エリアの役割、プログラムの実行方法、既存メソッドの組合せによるプログラミング	40
チュートリアル２（メソッドの作成）	メソッドについて、新メソッドの作成方法、作成例、メソッド実行時間（スピード）の調整法。	69
チュートリアル３（イベント処理）	イベントについて、起動時イベント、キーイベント、マウスイベント。	70
チュートリアル４（シーンの設計）	シーン編集モード、視点の変更方法、オブジェクトギャラリー、オブジェクトの追加、配置、回転、拡大縮小、複製	34

図11に日本語化後のチュートリアルの画面スナップショットを示す。網掛け部分は操作不能、網掛けのない部分は自由に操作できる。インストラクションから外れた操作をして、次のステップへ進む条件が満たされない場合は警告文が現れ、操作前の状態に戻される。

Alice のチュートリアルのスクリプトは、チュートリアルごとに独立したデータファイルとなっており、xml 形式で記述されている。図11のステップ付近のスクリプトを図12に示す。メッセージ部分は単に英文を日本語文に置き換えればよい（**置換**）が、それだけではチュートリアルは正しく動作しない。正常操作がなされたかどうかの判定条件やマスク情報に、具体的な座標値やコードが埋め込まれてしまっているからである。コードエディタ内のプログラム部品の日本語化はもちろん、フォントを変更しただけでも成否判定が狂ってしまうので、チュートリアルを１ステップずつ実行して確認しながら、条件テンプレートを書き換える必要があった（**改変**）。

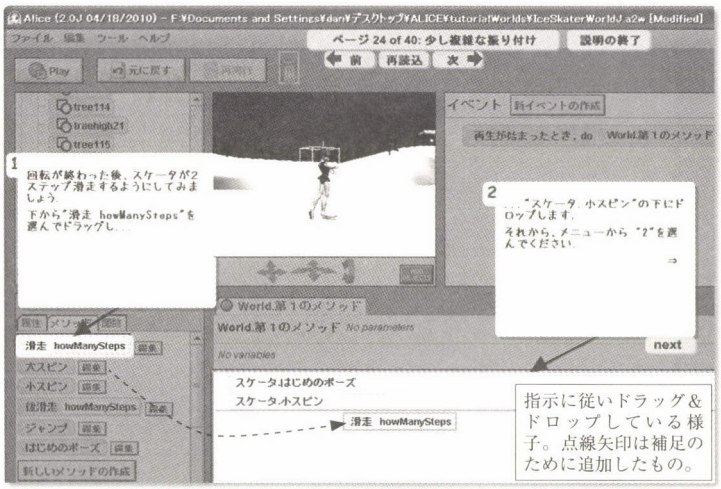


図11　日本語化チュートリアル

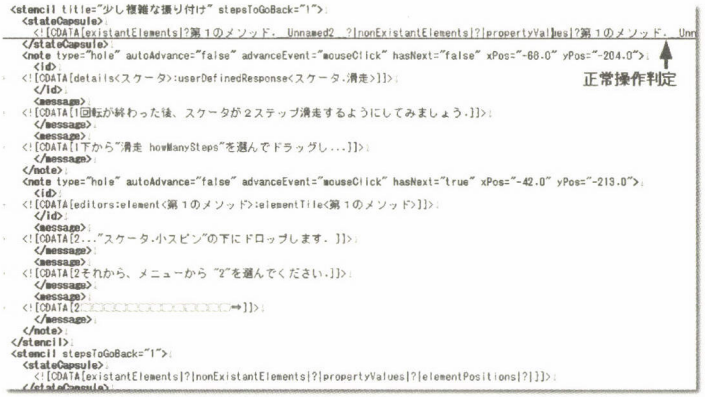


図12 チュートリアルのスクリプト（日本語化後）

3.4　Alice 日本語化のまとめ

最近のソフトウェアは多国語対応しているものが多くなってきている。最初から多国語対応を想定して作られたソフトウェアの場合、メッセージやメニューなど、言語に依存して変化する部分はリソース化し、ソースから独立させるのが一般的である。しかし、残念なことに Alice はそのような設計にはなっていない。属性や関数名など、一部はテンプレートとして独立リソース化してあるが、そういう部分もマルチバイト文字のことはまったく考慮されていない。リソースは加工して用いられたり、連想記憶に格納して間接参照されたりするが、そういった部分でシングルバイトを前提としたルーチンが組まれており、しかも関連するコードは局在化していない。このため、日本語化にあたっては、処理プログラムの深くまで解析して改変する必要があった。

最終的に若干英語の部分が残されてはいるが、初心者が接する部分はほとんど日本語化

できた。残された日本語化できていない部分をまとめると以下のとおりである。

① 簡単な英単語やメリハリのために英語のまま残した方がよいと判断した部分

簡潔で分かり易く、プログラム部品としてはそれ単独で特殊な意味を持つような英語表現はあえて日本語化せずにそのまま残した。大半が日本語化された環境においては、英語の部品は見た目に変化を与え、それが効果的な位置にあれば、構造を明瞭化するので、プログラムの構造の理解を助ける、という効果が期待できるためである。

例）more, other, do, Nothing, anything, No parameter, など

② めったに使われない属性や専門用語など

Alice にはもともと Seldom Used Properties という項目がある。この部分は過度に専門的な用語が多く、日本語化しても意味が通じにくい。文字通りめったに使われないので、あえて日本語に直す必然性はないと判断した。

例）quaternion, isPivotShowing, isFirstClass, eventStopAscending, opacityMap

③ ソースが公開されていない部分

Alice はほとんどの部分が Java 言語と Jython で記述されており、ソースプログラムも公開されているが、３D のレンダリング部分だけは速度を重視して Windows の DirectX７をネイティブで呼出しており、この部分の改変は困難である。残念なことに、この描画ルーチンがマルチバイト文字に対応していないため、アニメーション再生時にセリフの日本語が文字化けしてしまう。この問題は残されたままである。

セリフや独白内で日本語が使えないという問題は、プログラミング学習やアニメーション学習という観点では、本質的な問題とはいえない。しかしながら、学習者の多くが最初に手がけるアニメーションは登場キャラクター同士のたわいないやり取りであることが多い。母国語のセリフが使えないということは、日本人学習者のモチベーションを損なう要因となることは間違いない。サイレントで味のあるアニメーションや楽しい動く仕掛けを作れるセンスのある学習者はそれほど多くはない。

Alice には機能を単純化し、物語作りに特化した Storytelling Alice というバージョンが存在する。Storytelling Alice は中学・高校の初等中等情報処理教育、あるいは創作力開発のための教材として利用され、特に女子学生に歓迎して受け入れられることが報告されている。セリフなしでは物語は成り立たないので、日本の中学生、高校生に Alice を広く受け入れてもらうためには、セリフの日本語化は不可欠であろう。

現在 CMU での Alice プロジェクトはバージョン2.2の開発を終了し、3.0が開発途上にある。次のバージョンでは多国語対応、あるいは多国語対応を前提とした設計になっていることを期待したい。

4. 日本語化 Alice の教育的効果とその意義

4.1 教育効果について

2009年度に情報学部の大学２年次生と大学３年次生を対象としてそれぞれ半年間 Alice を用いたプログラミング教育を実施した。また、2006年度と2009年度に大阪府の中学生を対象とした夏休みセミナーで日本語化 Alice を用いたアニメーション教室を開催した。すべて少人数クラス（８～12人）であり、統計的なデータを得るには至っていない。

大学生向けの授業では、はじめは日本語版を用いてチュートリアルなどをこなさせた後、慣れてきたところで英語版に移行させた。一旦基本的な使い方をマスターしてしまえば、英語版に切り替えても、特に不自由する様子はなかった。Alice（アニメーションプログラム）は視覚的に理解できる部分が多いので、最初の導入にさえ成功すれば、あとは英語版でも不自由しないようである。

２回実施した中学生向けのセミナーは２日間であるが、Alice を用いた学習は１日のみ、時間にして５時間程度であった。通常のプログラミング言語を用いて、分岐と繰返し、逐次実行と並列実行といったプログラミングの基本概念を実習込みで１日で済ませるのはほぼ不可能に近いが、Alice であれば、中学生でも無理なく１日で理解でき、それなりのアニメーション作品を作り上げることができる。イベント駆動という比較的高度な概念も、特に混乱なく受け入れてもらえた。

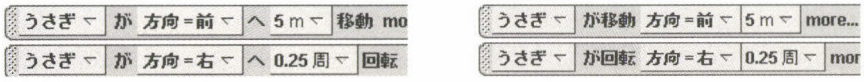
米国での調査研究でも報告されているが、Alice を用いたプログラミング学習では、学生が自主的にプログラムを改良しようとする積極的な姿勢が見られる点が特徴的である。通常のプログラミング学習では与えられた課題以上のことをやろうとする学生は少ない。これに対し Alice では、ほとんどの学生が与えられた課題を発展させようとしたり、オリジナルのアニメーション作りにチャレンジしたりして、学習を楽しむ様子が見られる。

これはもちろん日本語化による効果ということではなく、Alice そのものの効果である。しかし、英語版のままではそもそもシステムに馴染もうとしない学生が出るだろう。

4.2 日本語プログラミング環境としての日本語化 Alice

前章で説明したように、今回作成した日本語化 Alice では、コードエディタも日本語化してある。ここでいう日本語化は単にメニューやヘルプメッセージの日本語化にとどまらない。Alice は Java 風の擬似オブジェクト指向プログラミング言語を扱うシステムであるが、その言語の日本語化という意味も含まれる。前述したように、Alice のコードエディタはキー入力を極力排し、プログラム部品のテンプレートとメニューの選択が中心である。その主だった部分を日本語化し、変数・パラメータ・メソッド・関数に全角文字が利用できるようにしてあるので、利用者が積極的に日本語文字を使用すれば、見かけ上日本語プログラミング言語的な利用が可能である。

そもそもオブジェクト指向言語は名詞先行の日本語文法とは相性が良いので、予約語とユーザ定義語の日本語化だけでも日本人にはずいぶんわかりやすい表現となる。さらにAliceはテンプレートを差し替えるだけで見かけの語順を自由に改変できるので、工夫すればより文法的に日本語に近い表現を作ることも可能である。テンプレートはリソース化されており、入れ替えに再コンパイルは必要としない。



(a) (b)

図13 テンプレートの違いによる語順の違い

例えば、図13の（a）（b）は同じプログラムを、テンプレートを変えて表示してみたものである。この両者を単純比較すれば（a）の方が日本語として自然である。しかし、「移動」というメソッドには、オプションの修飾表現（例えば「時間」「変化スタイル」「回転中心」等）が存在し、それらは文末に順次追加されていく。このことを考慮して、日本語化Aliceでは（b）の表現を採用している。（a）と比べて（b）が了解性で著しく劣るわけではないし、修飾語が複数付属してきた場合、（b）の方が表現の一貫性を保てるからである。もし、自然な日本語にこだわるなら、語順や表現自体を適宜変更するという方法も考えられなくはないが、それには単純なテンプレートでは対応不可能であり、本格的なシステムの改変を要する。

現在実用化されているほとんどのプログラミング言語は英語ベースである。学習にそれほど際立った英語力は不要とはいえ、多くの日本人学習者にとって英語が障害の一つであることは間違いない。特に中高生にプログラミングの諸概念をつかんでもらうのが目的であれば、実言語を利用する必然性はなく、母国語が使える環境の方が学習効果は高い。

このため、プログラミングの導入教育用に、日本語プログラミング言語やその教育用環境が開発・利用されている例も多い［10］。大阪学院大学の西田研究室と大阪市立大学の松浦研究室で共同開発されているPEN［11］や慶應義塾大学の大岩研究室で開発されている「ことだまon Squeak」［12］などがその例である。

PENは大学入試センターの入試科目「情報関係基礎」で用いられる日本語による疑似手続型プログラミング言語DNCLを拡張したプログラミング言語xDNCLの実装環境である。言語が日本語に基づいていることに加え、機能がプログラミング導入教育用に絞り込まれているので初心者には分かり易く、またAlice同様、多くの入力操作をマウスで行えるような入力支援機能が実装されている。

「ことだまon Squeak」は子供向けオブジェクト指向プログラミング言語Squeakとアルゴリズム教育用に開発された日本語プログラミング言語「言霊」とを組み合わせた環境で

ある。「言霊」はプログラムの読み易さを追求した言語であり、語尾変化を伴う正しい日本語表現が特徴である。残念ながら読み易さは書き易さには直結しない。日本語表現には揺らぎがあるからである。この欠点を補うため「ことだま on Squeak」では、キー入力ではなくタイルと呼ばれるテンプレートを貼りつけるという、Alice と似たプログラミングスタイルが採用され、読み易く書き易い環境が実現されている。

初心者にとって、正しい日本語で書かれたプログラムが読み易いことは否定できないし、とことん正しい日本語表現でプログラムするということへの技術的な興味も理解できるが、それと実用性とは別である。初心者の時期はわずかであり、その時期を過ぎればいずれ実際の言語と向かい合わねばならぬ時が来る。その時のギャップはあまり大きくない方がよい。例えば PEN の xDNCL は、代入文など日本語化されていない部分も多いが、全部日本語化したら可読性が高まるかというとそうでもなく、むしろ日本語表現と英語や数式が混在している方がプログラムの構造がつかみ易くなることも多く、実際のプログラミング言語へ移行したときの戸惑いも少なくてすむ。

大事なことは、初心者の心のバリアを解くことであり、それには PEN や Alice の日本語化の程度で十分な効果があると考えられる。

おわりに

かつて、計算機が特別な人達だけのものであった時代は、計算機に触れるとか、プログラミングが体験できるというだけで学生のモチベーションを高く維持することができた。当時の学生は、学習内容がいかに素朴であっても、それが先端であるという自負を持ち、満足感を得た。それに比べ、今の情報系学生は不幸である。家庭や学校にコンピュータが当たり前に入り込み、高等学校では「情報」が必修科目となって久しい。大学では専門にかかわらずすべての学部の学生が情報教育を受ける機会をもつ。情報系学部に入っても、自分たちが特別なのだという優越感は得られない。それどころか、技術が進みすぎ、授業内容と現実のギャップの大きさに絶望感さえ感じる。授業内容は理解できてなくても、普段目にするアプリケーションや情報システムまでの道のりが気の遠くなるほど遠いことは分かる。

どんなに時代が進もうとも、基本から始めるしかなく、基本は変化しないから、従来通りのプログラミング入門でよろしい、ついて来ない学生は勝手にせよと突き放すのは容易いが、ゴールとのギャップが大きすぎてやる気が起こらないという気持ちは理解してやらねばならない。Alice でそのギャップが埋まるかどうかは問題ではない。錯覚であってもゴールまでの道がつながりそうだという手ごたえを感じさせてやることが大事なのである。最初の一歩を踏み出さなければ、何も始まらない。Alice の最大の意義はその一歩を拒むバリアを取り除くことであり、その意味で日本においては日本語化が不可欠である。

参考文献

［1］S. Cooper, B. Conover, and W.Dann: “Alice: A Fresh Approach to Teaching Computer Science”, http://visualization.sju.edu/ .

［2］B.Moskal, D. Lurie and S.Cooper: “Evaluating the effectiveness of a new instructional approach”, Proceeding SIGCSE '04（2004）.（http://www.alice.org/publications/EvaluatingTheEffectivenessOfANewApproach.pdf）

［3］Educational Institutions Using Alice（Alice 公式ページ内）: http://www.alice.org/index.php?page=alice_users/alice_users

［4］R.Pausch, T.Burnette, A. C. Capehart, M., D. Cosgrove, R. DeLine, J. Durbin, R.Gossweiler, S. Koga, J. White: “Alice: Rapid prototyping system for virtual reality”, IEEE Computer Graphics and Applications, 15（3）, pp.8-11（1995）.

［5］広内哲夫: “Web３D グラフィックス― VRML で創るバーチャルワールド”, ピアソン・エデュケーション（2001）.

［6］“The Virtual Reality Modeling Language” http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/.

［7］Dann, S. Cooper and R.Pausch: “Learning to Program with Alice”, Prentice Hall（2006）.

［8］NHK TVML 公式ホームページ : http://www.nhk.or.jp/strl/tvml/index.html

［9］林正樹：“めざせ！テレビ番組クリエイター―パソコンと番組記述言語 TVML で実現 !!”, 技術評論社（2004）.

[10] 兼宗進：“教育用プログラミング言語の動向”, IPSJ Magazine 48（6）, pp.4-7（2007）.

[11] 西田知博 , 原田章 , 中村亮太 , 宮本友介 , 松浦敏雄：“初学者用プログラミング学習環境 PEN の実装と評価”, 情報処理学会論文誌第48巻第８号 , pp.2736-2747（2007）.

[12] 岡田健 , 杉浦学 , 松澤芳昭 , 大岩元 : “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”, cybermedia forum, No.7, pp.17-22（2006）.

（2010年12月6日受理）