ビスケットにおけるプログラムの美しさについて

原田康徳（NTT　コミュニケーション科学基礎研究所）

概要

ビスケットというビジュアルプログラミング言語の開発を通じて，プログラミングを万人に伝えるという活動などを述べる．

ビスケットの開発の目的

ビスケットの開発の目的は，プログラミングの楽しさと可能性を万人に伝えたい，ということである．それは，文化的に豊かな情報化社会を実現するためには，現在のようなプログラミングが特別な人だけできればよいという態度ではなく，すべての人がプログラミングとはどういうことかを理解する必要があるからである．料理にたとえると，世界の素晴らしい料理のいくつかは，家庭料理がベースとなって発展したものである．普段料理をしない人であっても，料理とはどういうものかくらいは誰でも知っている．その程度にプログラミングを知ってほしいということである．

現在，コンピュータは様々なものに使われ，我々の生活に無くてはならないものとなっている．しかし，コンピュータとはどういうものかという基本的な事柄は，誤解されている．多くの人は，コンピュータの性質ではなく，コンピュータの上で動くアプリケーションの性質から，コンピュータとはどういうものかを想像しているだけのようである．さらに，残念なことは，この誤解している多くの人たちが，この国の政治や社会を動かす重要なポジションにいるということである．そして，彼らは自分が誤解しているということに気付かないまま，たとえばコンピュータを教育に利用するという政策を実行し，まったく頓珍漢な方向にすすんでしまっている．

コンピュータとは何かを理解するのは，コンピュータを直接触る以外に，理解することができない．なぜなら，コンピュータは何かに例えられるような装置ではないからである．ほとんどの発明は，それまで世の中に存在していたものとの差分で説明できる．写真はすごい絵だし，映画は動く写真である．テレビは家でも見ることができる映画である．しかし，コンピュータはあまりにも独特で，何かのたとえでは説明できない．強いて説明するなら，すごく計算が速い，ということかもしれないが，この「すごい」というのが桁違いである．自動車は速い馬車と言ったとき，速いと言ってもせいぜい 10 倍程度である．世の中の大半の発明は目に見える程度のすごさでしかない．しかし，コンピュータのすごい速いというのは，本当に説明ができないくらい速い．こういうことも説明が難しい．つまり，何かに例えて説明ができないということは，本を読むだけで理解できる知識とは根本的に違うということである．もう，直接コンピュータを触って，こういったコンピュータの性

質を体験する以外には理解できないのである．ここが，プログラミングをやったことがある人と無い人で根本的にコンピュータに対する考え方が変わる理由である．

ビスケットとはどういうものか

ビスケットについての解説とデモは Web サイト www.viscuit.com を参照していただく．簡単に性質だけあげると，絵の配置を書き換える書き換え言語である．書き換えルール（これは形状からメガネと呼んでいる）の集合がプログラムである．

ビスケットには，プログラミングの理解において重要だと思われている，変数，制御，条件分岐，といった概念は存在しない．これらは，実用的なプログラミングにおいては重要であるかもしれないが，コンピュータとは何かを体験で理解してもらう，という目標に置いては重要ではない．言語仕様はシンプルであるべきである．

コンピュータにとって，理解してほしい非常に重要な性質とは，小さい変化しかできない，その変化が非常に正確に高速に実行する，ということにつきる．プログラミング言語のスタイルの違いは，小さな変化をどのようにしてコンピュータに指令するかの違いとなる．手続き型言語は変化を列として表現している．一方，ビスケットのようなルールベース型言語は，条件と変化の組をルールとして表現している．特に書き換え型言語では，二つのパターンで，条件と変化を記述できるので，非常にシンプルである．

教育用言語において重要なことは，我々がプログラミングの過程で体験する様々なことが，同じように体験できるのがよい．ある種の便利な機能は大きなプログラムを作る上で必須であるけれども，重要な体験を奪う可能性もあるので，導入は慎重であるべきである．くだらないバグやエラーに悩まされることは重要な体験ではないが，全体を設計して，部分的に動作を確認しながら構築しないと，大きなシステムは作れない，ということを体験するのは重要である．

コンピュータに対する誤解

ビスケットをいろいろな人にデモをしていると，相手の反応によってその人がどのようにコンピュータをとらえているのかがよくわかる．理系であるからといって，コンピュータの基礎を理解しているとは限らないのが，非常に残念なところである．たとえば，メガネの中の配置をいろいろと変えることで，様々なアニメーションができるというデモをしているときに，「このソフトは何種類のアニメーションが入っているのか」という質問を受けたことがある．最初この意味が理解できなかったのだが，どうやらこの質問をした人は，あらかじめ動きのパターンがプログラミングされていて，メガネの中の配置をパターンマッチして，用意された動きのプログラムを呼び出している，と勘違いしたらしい．よく考えると，たとえば音声対話システムなどはこのように作られている．何か音声で質問すると非常に気の利いた返答が音声で返ってくる．しかし，本当の人間なら当然応えられるような，基本的な質問にまったく応えられないこともある．コンピュータとはこういう音声

対話システムのようなものだという理解なのである．この人はプログラミングを知っているのかもしれないけれど，非常に狭い理解である．プログラミング言語は自分で作れる，ということがまったく理解できていないようだ．

別の人の反応で，2 コマのアニメーションのデモのとき．A と B という二つの絵があり，これが交互に変化するアニメーションを作りたい．A→B という 1 方向の変化のメガネを作ってみせる．画面上の A はすべて B に変化する．ここで，本当は B→A というもう一つのメガネを作れば，交互のアニメーションが得られるのであるが，それを自分で発見させるということをやっている．このとき，ある人が，「メガネの上で右クリックすると，メニューが出てくるので，そこに繰り返すという項目があるはずだ」と答えた．この人も，コンピュータをよく知っているけれど，そもそも，コンピュータには沢山の機能があらかじめ用意されていて，それをどの順番で呼び出せばよいか，という理解しかしていない．小さな機能を組み合わせて複雑な機能を作る，とは理解していない．

考えてみれば，これまでの世の中のアプリケーションはほとんどがこのような作りである．プログラミング言語的な，つまり小さな機能だけ用意して，それらを組み合わせる，という操作法を提供しているアプリケーションはほとんどない．ときどき，あるのはプログラミングがバリバリできる人むけのマクロ言語であるが，あれはあまりにも本物すぎる．もっと，アプリケーションのレベルでプログラミング的な要素があってもよいかもしれない．現状が，このようなアプリケーションにあふれているのだから，コンピュータを誤解する人が居ても仕方がない．

プログラムの美しさ

プログラムの美しさの議論には，3 つの視点がある．

1. プログラムを実行した結果の美しさ
2. プログラムコードの美しさ
3. プログラム言語仕様の美しさ

ビスケットはビジュアルプログラミング言語であるので，１．の視点も重要である．たとえば，美大の授業で美しいアニメーションを作るという課題でビスケットを使用した例がある．一般にはメディアアートのような分野がここに相当する．

２．の視点は，メガネをどのように配置するかという純粋に GUI 的な問題と，できるだけ少ないメガネで複雑な動きを作る（冗長なメガネを削除する）という論理的な問題がある．ビスケットは教育用という性格があるため，メガネの配置に関して特にソフトウェア的な配慮（たとえば，メガネを自動的に美しく配置する）は何もしていない．その結果，メガネをわかりやすく並べると，後から見やすい，ぐちゃぐちゃに並べると，後からわかりにくくなる，ということを体験できる．プログラムが単純であるうちは，整理して見やすくするということは重要ではない．プログラムが複雑になって初めて重要になってくる．初心者に向けて，つまり単純なプログラムしか作っていない段階で，整理のことを教える必

要はない．２．の論理的な視点は，たとえばステップ実行や，発火したメガネの可視化などによって，無駄なメガネを発見することはできると思うが，現在のところそのような機能はない．

３．の言語仕様の美しさは，ビスケットの言語仕様の単純さに表れている．ルールベースの言語であるから，言語の基本構文は一つだけである．様々な追加機能は，ルールの条件側に入れるセンサーか，アクション側に入れるアクチュエータとして実装される．教育用言語において言語仕様の単純さは非常に重要である．この言語が伝えたいメッセージがより明確になる．言語仕様が複雑であると，多くの挫折者を生み，本来の目的である「プログラミングの楽しさと可能性を伝えたい」ということを万人に伝えることができなくなってしまう．また，「可能性」を伝えるための演出も重要である．最初から，システムが高機能であった場合，複雑な動きができるのは，システムに組み込まれた機能を利用しているだけという勘違いをしてしまう．たとえば，文部科学省のサイトで公開されている「プログラミン」には，右に動く，左に動く，といった動く方向の違いで 10 種類の命令が存在する，そのうちの 6 種類についてはビスケットでは一つのメガネによって実現可能であるし，他の種類もメガネを増やしたり絵を増やしたりして対応することができる．絵を動かすというのは，プログラミングの入門において，もっとも基本的な部分であると思うが，この時点で，「コンピュータとは小さな機能を組み合わせて複雑なものをつくることができる」，と思わせるのか，「コンピュータとはあらかじめ用意されている機能を組み合わせて複雑なものをつくることができるものだ」と思わせるのか，どちらの方が可能性を感じさせられるだろうか．

会場での質疑応答など

Q．何歳ぐらいの子どもができるのか

A．ただ動かすだけなら 3 歳．2 コマの繰り返しは 6 歳くらい．衝突判定は 9 歳くらい．

Q．美というのは面白がれる能力があって伝わる．小さい子でも面白がれるのか．

A．誰かが作った新しい動きは目立つので，そこから動かし方への興味が広がり，伝搬する．

Q．ビスケットはどうやって思いついたのか．

A．先行研究はいろいろあった．KidSim(現在の StageCast), AgentSheets など．ビスケットの独自性は，曖昧なマッチングを導入した点．

Q．単純なものの組み合わせは vi に通じるものがある．

Q．アナログな VisuLan か．グリッドはないのか？

A．グリッドがないと，動き続けて壊れてしまうので，グリッドは大きなシステムを作るうえでは重要．最初に壊れるのがあって，壊れない方がうれしいと思わせて，壊れない方

法を教えるのが筋．たとえば，積み木で線路を作って，電車を転がして遊ぶとき，積み木の線路はずれるのがあたりまえ．でも，いまのコンピュータは壊れないのが普通なので，自然な感覚とはづれている．

Q．音は？

A．音は鳴ります．プログラミングで音を鳴らすというのは，作曲マシンを自分で作るということになり，新しい音楽教育に使えるのではないか．

Q．来年から学習指導要領がかわり，プログラミングが必須になるが，レゴマインドストームのような制御が中心になるのはどう思うか．

A．5 年生ぐらいの授業の最後に，チェーンメールの仕組みの例をビスケットで見せると，なぜチェーンメールを出してはいけないのかということを規則ではなく，体験として教えることができる．これが情報のすごさであり怖さであるのだけれど，これは早い時期に教える必要がある．中学の技術の時間に教える計測と制御の一つとしてのコンピュータのとらえ方はちょっと狭いのではないか．

Q．ビスケットの開発体制

A．他人と一緒に仕事ができない．どこかはオープンソースにしないとだめですよね．

Q．メガネの配置をボタンで整列にしないのか．

A．教育的な視点など，よくよく考えてやりたい．