特集号
招待論文

プログラミング言語ビスケットを用いた基礎としてのプログラミング教育の提案と実践

原田 康徳[†1]

[†1] 日本電信電話（株）

プログラミング教育を，実用的なプログラムを書くためのスキル教育だけでなく，情報化社会を生き抜く上で広く全員が知るべき教養的な位置づけで捉える．筆者は，ビジュアルプログラミング言語ビスケットを使って「コンピュータとは何か」を一般人（小学生から大人まで）に伝える活動（授業，ワークショップ，イベント）を行っている．その内容を紹介する．

1. はじめに

子どもへのプログラミング教育が社会的にブームになっており，実際にいくつかの国が義務教育の中にプログラミングを本格的に導入し始めた．我が国においても，民間レベルでの子ども向けプログラミング教室が盛んに行われ，国レベルでも少しずつであるが進展している．筆者も，ビスケットという子ども向けプログラミング言語[1],[2] を2003年に開発し，それを小中学校，科学館，児童館，イベントなどで子どもたちにプログラミングを教える活動をしている[3],[4].

筆者らの活動の目的は，子どもたちにプログラミングのスキルを身につけさせたいのではなく，「コンピュータとは何か」ということに関する直感を持ってほしいということにある．昨今のプログラミング教育ブームでは，役に立つアプリを作れる能力を早い時期から身につけさせるということが主なる目的であることと対照的である．

コンピュータやインターネットが広く普及し，もはやコンピュータと無関係に生活することは不可能である．プログラミングの仕事に就くか就かないかにかかわらず，コンピュータとは何かという，基本原理の理解がブラックボックスのままで良いわけはない．現代に生きるすべての人が知るべきと考える．

しかし，しばしば「コンピュータとは何か」という問いに対して，コンピュータの上で動いている「あるソフトウェア」の性質をもって，それをコンピュータだと誤解してしまう人がいる．たとえば「コンピュータは便利だけれども融通は利かない」というのは，コンピュータそのものの性質ではなく，単にそこで動いているソフトウェアがそういう性質だというだけである．技術者や研究者はこのような間違いはしないが，多くの一般人がこういった基本的な点から誤解している．

一方で，コンピュータの技術者・研究者たちは「コンピュータとは何か」という説明を一般の人向けにほとんどしてこなかったのではないだろうか．自分たちにとっては当たり前すぎることであるが，その中身に接していない人たちに対して，丁寧な説明が必要であろう．

そのため，筆者らの活動の基本的なパターンは，ビスケットによるプログラミングの体験だけでなく，体験の最後に「コンピュータとは何か」ということを解説する簡単な講義とセットになっている．講義は体験した子どもだけでなく，見学している保護者の方々も意識している．

本稿はそのような活動を通じて，参加者に伝えてきた「コンピュータとは何か」という筆者なりの解説を述べたものである．直接言葉で説明するだけでなく，純粋な体験の中から言葉を使わずに伝えている部分もある．体系的な評価は行ってはいないが，参加者・保護者の反応やアンケートなどを見ると，より高くプログラミング教育への重要性が理解され，その試みは成功しているようである．

2. ビスケットによるコンピュータ入門

ビスケットを用いた活動は，子ども向けから大人向けまで幅広く行われている．1人1台のタブレットを用いた体験が原則で，最も短いもので30名ほどの小学3年生から小学6年生の児童に対して25分，長いもので2時間の3回コース．大規模なケースでは，4人に1台のタブレット

で，一度に150人の高校生を対象にした講義などである．いずれの場合も単なる体験では終わらず，その体験から説明できる話題で，「コンピュータとは何か」という短い講義を行っている．これまでに，さまざまな体験と講義の組合せを開発してきたので，それらを手順に従って紹介する．

2.1　基本原理の理解

体験の時間や年齢層にかかわらず，必ず最初に体験するビスケットの最も基本的な部分を説明する．最初から簡単な絵が用意されている場合と，自分で絵を描いてから始める場合がある．

まず，自分で描いた絵（またはあらかじめ用意されている絵）を複数個ステージに入れる（図1（1））．ビスケットのプログラムは，メガネと呼ばれる書き換え規則によって作られる．メガネの左側にある絵を右側の絵に書き換える，という意味を持つ．メガネのそれぞれのレンズの中には，複数の絵を入れることができ，あいまいなマッチングを行いそれらの配置に近い場合に書き換えが起きる．図1（2）ではメガネの両側に同じ絵を少し左の位置に入れているので，ステージ上のそれと同じ絵が一斉に左に動く（図1（3））．

ビスケットには実行のモードがないため，図1（4）のようにメガネの中の絵の位置を変えるとステージの絵の動きが一斉に変わる（図1（5））．また，メガネから絵がでてしまうと（ここではメガネの左右で絵の数が同じという制約がある初心者モードであるため）そのメガネが無効となり（図2（6）），ステージの絵が一斉に止まる（図2（7））．

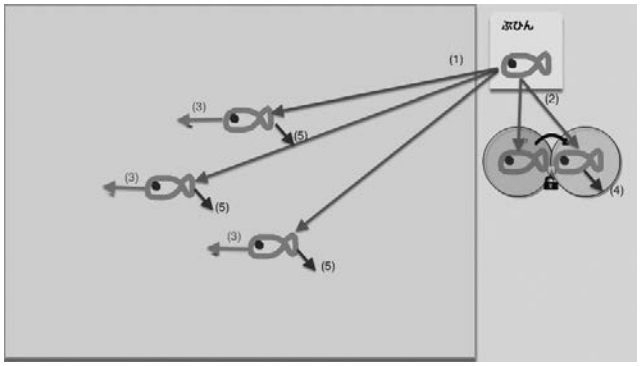

図1　ビスケットの基本

図2　プログラムが壊れた

ここまでがビスケットを触らせてから最初の5分で，すべての人が体験する部分である．ここに，コンピュータのきわめて本質的な説明が入っている．まず，1つのプログラムが複数の絵を動かすために何度でも使われている．そして，1つのプログラムを変更すると，すべての同じ絵の動きが一斉に変化する．プログラムが壊れるとそれらの動きが一斉に止まる．

このように一斉に動きが変わったり，一斉に壊れたりということは，情報システムならではの性質である．反対に，たとえば自動車などのリコールは故障する可能性があるから修理するが，必ず故障するわけではない．コンピュータでは問題が見つかれば確実に不具合となって現れ，一斉に故障する．自動車では不具合を1台ずつ修理しなければならないが，コンピュータでは一斉にソフトウェアのバージョンアップで修理することができる．そういった極端な長所と欠点を同時に持っているのがコンピュータなのである．

この部分に関しては明示的な説明をすることはあまりないが，すべての参加者が最初の5分でこの体験をし，コンピュータの最も基本的な性質を体験することになる．

2.2　論理と表現

体験が少し先に進み，開始10分～20分くらいで尺取り虫の例を出すことがある（図3）．これは，2コマの絵でつくられたアニメーションである．メガネの左右に異なる絵を入れた場合，書き換えによってステージの絵は切り替わる．書き換えは一定の時間間隔で行われるため，2つのメガネがあると，2つの絵が交互に提示される2コ

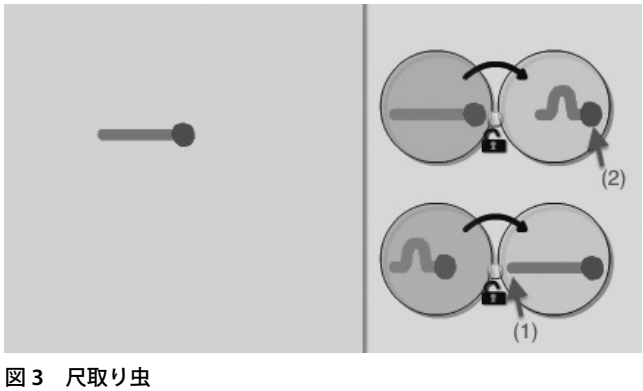

図3　尺取り虫

マのアニメーションが得られる．

しかし，2コマの虫の絵が動くアニメーションは簡単に得られても，その虫がきちんと歩いているようには見えない．歩かせるには，アニメーションの各コマの変化に対して，絵のどの部分がズレるのか，ズレないのか，ということを意識する必要がある．まず，縮んでいる虫が伸びるときを考えると，虫のしっぽは地面にくっついて動かない．そこで，メガネの左右で，しっぽの位置をそろえて置く(図3(1))．同様に，伸びた虫が縮むときは，頭が地面にくっついていて頭がズレないように置く（図3（2)）．このように絵のズレ方を意識してメガネをつくると，虫のアニメーションがリアルに見えるようになる．

しっぽや頭の位置が違ってもプログラムの論理的な構造はまったく同じで，2 つの絵が位置を変えながら交互に提示されているだけである．しかし，人間にとっては，プログラムのちょっとした違いで動き方が明確に違うと感じられる．コンピュータの視点(論理)と人間の視点(表現）がまったく異なっている．これがコンピュータを使う上で理解しておかなければならない重要な点である．

プログラミングとは関係ないが，このことは，ものが動くということはどういうことかを考えさせる仕組みでもある．この性質を利用して，ビスケットを水族館や動物園などでの生き物観察による作品づくりのツールとして使うことがある．この場合には，プログラミングの体験ということは一切言わずに実施している．

2.3　条件判定

図4は歩いている虫がボールを蹴るプログラムである．通常の動き（伸びる，縮む）のほかに，虫が歩いていてボールにぶつかると（図4（1)），虫が伸びてボールが遠くに飛んでいく（図4（2））という特別な動きが追加されている．コンピュータにとっての条件とは，動作に対する目のようなものである．条件がなければ虫にはボールは見えない．ボールの上を素通りするだけである．条件を導入して初めて虫にはボールが見えるようになるのである．

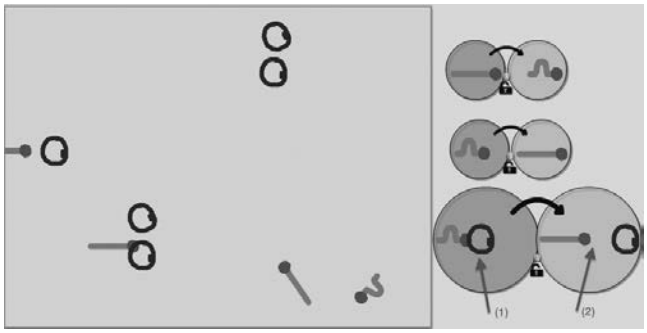

図 4　ボールを蹴るプログラム

たった1つの条件が入るだけで，虫たちは非常に面白い動きをする．1匹の虫が2つのボールを交互に蹴ったり，1つのボールを2匹の虫が奪い合ったり．動きは複雑になり，予測が難しくなる．

システムの複雑さはどこからくるのか．条件判定がないプログラムは，単なる記録再生機である．単純な動きしかせず，動きの予測も簡単である．複雑さは条件判断で生まれるのである．

このボールを蹴る虫のデモは，ビスケットの短い体験のバージョン（25 分～ 45 分）では最後に行い，長い体験のときでも途中に必ず行っている．その際には，次のような解説をしている．

> 1つ1つのメガネは単純なことしかしない．しかし，メガネを増やしていくと動きがどんどん複雑になっていく．最初に想定していなかったことが起きることもある．普段親しんでいるテレビゲームも複雑に動いているが，実は単純なメガネを1 万個とか100 万個とか大量に使って動いているだけである．プログラムを作るということは，このような大量なメガネを間違わずに作っていくという作業である．

どれくらいの参加者がこの説明を理解してくれているか分からないが，そのときの反応などを見ると，ある程度の説得力をもって伝わっているようである．

2.4　インタラクション

前節までの内容で1 時間の授業に収まるが，さらに時間がある場合は，次の1 時間でビスケットでのゲームづくりを行っている．プログラムの中にユーザの操作で反応する仕掛けを導入することで，インタラクティブな作品ができる．触ると箱が開くとか，好きなタイミングで弾を発射させるといったものである．

インタラクションが入ることで，ユーザがプログラムの実行に関与することができる．あらかじめプログラムの中に含まれていた複雑さに加えて，動的な制御が加わる．インタラクションがない中で，複雑に動き続けるシステムを作るのはそれなりに難しいが，インタラクションを導入すると比較的簡単に作れる．

インタラクションが入ることは，箱庭的（実験室的）なプログラムから製品的（作品的）なプログラムに進化させることでもある．

通常はこの内容を小学3年生以上に対して行っている．しかしゲーム性にこだわらなければ，小学1年生であっても，インタラクティブな作品を作ることはできている．

2.5　他人を意識したソフトウェアづくり

ゲームづくりの最中や最後に遊び合いの時間を設けている．遊び合いをすることで，自分が作りたいゲームから他人に遊んでもらうゲームへと意識が変わる．難しすぎるゲームも簡単すぎるゲームも他人には面白くない．ちょうど良い難しさにするのが重要である．一般的には絵が移動する速度を遅くすると，ゲームは易しくなるし，速くすると難しくなる．ビスケットでは絵の位置を動かすだけで速度の調整が簡単にできるため，ゲームの難しさのバランスを簡単にとることができる．

他人に遊んでもらって感想をもらったり，遊んでいる様子を自分で観察したりして，フィードバックをじかに作品に反映させる．こういった開発はまさにモノづくりの基本であるが，作文，工作，絵画といった，一般的な小学校での創作活動においてこれほど短時間で，改良のサイクルが実践できるメディアはあまりないのではなかろうか．

小学校におけるゲームづくりの授業は，すでに30回以上実施している．どの回も，楽しそうに自分たちの作品を自慢して，友達や見学に来た人たちに遊んでもらって楽しんでいる．ある学校では，自分のゲームを校長先生に遊んでもらう順番待ちができ，その学校の良い雰囲気が現れていた．

2.6　シミュレーション

風邪感染を例としてシミュレーションをつくる実践も行っている．図5 に示すように，健康な人は右に動く (1)，風邪をひいた人は上に動く (2)，健康な人と風邪をひいた人がぶつかると風邪がうつる (3)，という3つのメガネの動作を設定する．これを確認した後，ステージに，健康な人を10人いれ，風邪をひいた人を1人いれる．最初はゆっくりと感染が広がり，やがて一気に感染が広がる様子が見られる．次に，風邪を治す病院を建てるシミュレートを行うために，図5 (4) に示す，病院に入ると風邪が治るというメガネを作る．まず，病院を1軒建てて様子を見るが，風邪をひいた人は病院になかなか当たらないため，治すのは難しい．病院を4軒くらいに増やしても，簡単には風邪を撲滅することはできないことを体験させる．

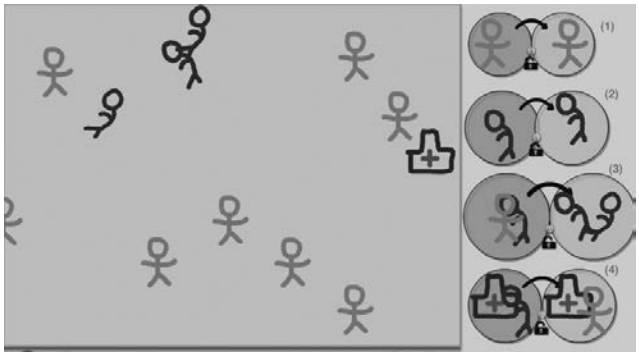

図 5　感染のシミュレーション

これは，小学生向けの2時間の授業の最後にデモとして見せている．また，中高生の授業ではこれを自分たちで作らせることもある．いずれも子どもたちから大歓声があがり，大人気の内容である．

ここでは，次のような解説をしている．

> 2つのメガネ「風邪がうつる」，「風邪が治る」はぶつかると相手の絵が変わるという点では似ている．しかし一方で風邪の人が倍々のように増えていくのに対して，他方では病院は増えない．この違いが非常に大きいことが分かる（これは，相手が大学生であれば，指数関数と線形関数の違いと説明できる）．またこれは，情報特有の性質である「複製で劣化しない」ということにも関係がある．この劣化することなく倍々で増えていく現象がコンピュータのすごさであり，怖さでもある．コンピュータが急に遅くなったり，突然フリーズしてしまう原因にもこのような増え方が関係していることがある．
>
> また，このようなコンピュータの中で行う実験をシミュレーションと呼んでいる．人の歩く速度を変えて感染の強さをコントロールし，感染が広がる速さを調べるなどの実験ができる．シミュレーションは天気予報などにも使われ，コンピュータの重要な応用の1つである．

小学6年生の授業でこれを作らせたときに，ある班では誰が一番最初に全員に感染させられるかという競争をしていた．別の児童は，シミュレーションのルールを変更して，健康な人と病気の人の数のバランスが自動的にとれるようなルールを見つけ出していた．そのクラスは1年前（5年生のとき）に1時間でゲームづくりをし，6年生でこのシミュレーションを1時間やっただけであるが，このようなシミュレーション的な遊びを，こちらからの指示なしに自発的にやれるようになっている点はビスケットを使ったプログラミング教育の1つの成果であると言える．

このシミュレーションを子どもたちに見せていて気づくのは，彼らはプログラムが作れるだけでなく，プログラムが読めるようになっているということである．2時間程度のビスケットを体験した後，最後にこのデモを見せる．健康な人を10人動かすあたりから，子どもたちの

反応がざわつき始める．これはプログラムを実行させる前から，感染が伝搬して広がるということに気づいていると言える. そして, 実際に走らせて予想通りになったり，ならなかったりすることを体験することも重要である.

プログラムが読めることは，非常に重要なリテラシーである. プログラムを読める力が広く一般に浸透すると，ディジタル機器の操作マニュアルなどの記述の仕方も変わるかもしれない.

シミュレーションの面白さは，最初に定義したルールだけからは，それがどのような結果になるのか予想ができないということでもある．これは実際に講座の中で発見したものであるが，ジャンケンのシミュレーションがその1つの例である．絵を3つにして（グー，チョキ，パー）それらが三つ巴になるようなルール（異なる絵がぶつかった場合，ジャンケンで負けた方の絵が勝った絵に変わる）というものである．中学校の授業で実施した際に，結果を予想させてみたところ，強さが対等になっているので，バランスが取れたまま動き続けるであろうという返答が担任の先生も含めて返ってきた．しかし，実際に走らせてみると，簡単にバランスがくずれ，最初に相手を滅ぼしてしまった手が最終的には負けるという予想外の結果になった．たとえば，グーが頑張ってチョキを滅ぼすと，パーの天敵のチョキがいなくなるため，最終的にグーが滅び，パーが勝つのである．これは，ルールを冷静に考えてみれば分かるはずのことであるが，シミュレーションをやるまではなかなか気がつかない，という例にもなっている.

現在では，シミュレーションはコンピュータのプロの世界では非常によく使われている手法ではあるが，まだまだ家庭や教育の分野には普及していない．プログラミングが一般化されたとき，コンピュータの真の力を利用した重要な用途として，シミュレーションが広く行われるようになるであろう.

2.7　2 進法と 10 進法

「コンピュータは0と1で動いている」ということは知識としては広く知られている．しかしなぜ0と1なのか，その理由はほとんど知られていない．体験を通じてそれを理解する授業も行っている．ここでは, 計算とは何か，人間とコンピュータの計算の違いとは何か，ということにも触れている.

図6のように，「1」と「0」という絵を描き，1と0が重なっていると，それが1に変わるというメガネを用意する（図6（1））．これを「1と0を足すと1になる」と読む．加えて，1と1が重なっていると，それらの1が消え，元の1 の場所に0が，0の左隣に新しい1が生成されるというメガネを追加する（図6（2））．これは「1と1を足すと10になる」と読む．メガネの左の1と右の0 の場所を一致させていることで桁を表現している．次に一番下の位に1を連続して生成するためのメガネを作る（図6（3））．この絵が口を開くたびに，その先に1が生成されていく．この絵は同じ場所でパクパクさせているので，一番下の位となる同じ場所に1が生成される．その結果，1が生成されるたびに，1を足すという計算を行うために，2進法でカウントしていくことになる（図6（4））.

2進法のカウンタは，一番下の位の生成部を除けば，わずか2つのメガネだけで作られる．他のほとんどのプログラミング言語が最初から数値計算の機能が入っているのに対して，ビスケットは計算をする機能は持っていない．そのため，このような2進法の原理的なことを説明するのに適している.

さて，2進法のカウンタが簡単につくれるのは加算に必要な知識が2つでよいからである（正確には，0＋0のプログラムも必要であるが）．計算とは何かを考えてもらうために，今度は３進法の計算を考えさせてみる．まず,「2」という絵を追加して，他の計算のためのメガネ（1+1->2，2+0->2，2+1->10，2+2->11）を追加する．これで3進法の加算ができる．ここで必要なメガネがぐっと増えていることを伝える．さらに4進法5進法と増やして行き，10進法の計算を考えさせる．これらは実際に制作せずに思考実験として行うだけで十分である．まず, 絵が「0」から「9」までの10個必要である．加算はすべての数の組合せが必要であるが，可換であるため実際にはその半分でよく，大体50個くらいのメガ

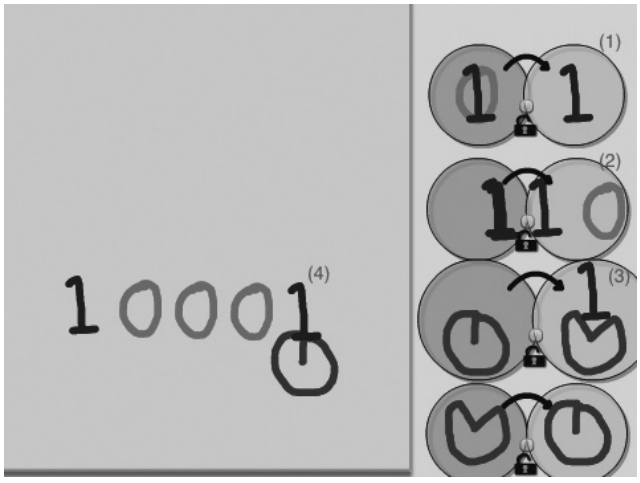

図6　2 進法

ネが必要ということが分かる．

2進法では2つのメガネが必要であり，3進法では4つのメガネ，10進法では50個のメガネが必要である．コンピュータが2進法で計算する理由は，2進法が一番簡単な計算だからである．それに対して私たちは，10進法の方が簡単な計算だと勘違いしている．しかし，それは小学校のときに50個のメガネを暗記したからであって，計算そのものが簡単だったからではない．これを実感させることが重要である．

また，単なる知識としてだけ「コンピュータは0と1で動いている」と理解させることは「0と1で動くからコンピュータは冷たい」，「論理的で感情は持たない」といった誤解を生む元であり，このような体験により，不思議でも難しいことでもないことを感じさせることが重要である．

この2進法から10進法の計算は，実際に子どもたちに作らせたことはそれほど多くはないが，講師がデモとして何度か見せている．2進法を学校で習っていなくても，桁が増えるごとに数が2倍になってゆくことに気付く小学生もいる．それ以上に見学に来ている大人たちにとって，コンピュータはなぜ0と1が使われているかという明確な理由が分かり，非常に面白がって話を聞いている．

2.8　プログラミング言語

コンピュータの上で，大規模で複雑なプログラムを作ることができるのは，プログラミング言語の階層という発明があるからである．プログラミング言語が別のプログラムで作られて，それがまた別の言語で作られ，という階層である．ここでは，中間言語・仮想機械なども1つの言語として扱っている．このことは，まさにコンピュータの中の常識ではあるが，他に類似の例を見ない，独特な仕組みである．この一端をビスケットで体験する．

図7はビスケットの上でオルゴールをつくったものである．絵と音階を対応させ，ある絵にぶつかると特定の音階がなるというプログラムとなっている．絵を適切にならべ実行すると，曲を演奏させることができる．

さらに，プログラムを工夫すればフレーズを繰り返したり，条件判定を利用して途中を変えたりすることが可能である☆1．オルゴールのプログラムはそのままで，ステージ上の絵の配置を変えることでさまざまな曲を演奏させることができる．絵の配置をプログラムとみなすと，オルゴール言語のインタプリタを作ったことに相当する．

このような，自分でプログラミング言語を作った体験の後で，ビスケット自身の実装の話をする．まずビスケットの内部構造（XML 表記）を見せ，それを実行するビスケットのソースコード（ActionScript）を見せる．さらに，その下にある swf, FlashPlayer, C++, llvm, 機械語といった言語の階層を説明する．

コンピュータの中身は，プログラミング言語（と仮想機械）が何段にも積み重なってできており，今回はその一番上にオルゴール言語をつくった．コンピュータは高速化大容量化しているが，この階層のおかげで，非常に複雑なシステムを簡単に作れるようになった．また階層化によって，さまざまな種類のハードウェアに同じプログラムを走らせることができるようになった，という説明をする．

この講義は，大人を対象に「ビスケットによるコンピュータ入門」という講座で実施した．どの参加者もコンピュータの奥深い世界を知ることができて満足度の高い講座となった．

3. まとめ

以上のような，プログラミングを通じて「コンピュータとは何か」ということを，さまざまな層に伝えてきた．

コンピュータは部分的に見ると単純なことしかしていない．しかし，その組合せによって複雑さが生み出されるのである．単純な機能の中には，絵を次々と複製するような本質的に危険な操作もある．そういった直感が身につくと，コンピュータを取り巻く生活が変わって見えてくるのではないだろうか．

ここに述べた例は，体系立てて「コンピュータとは

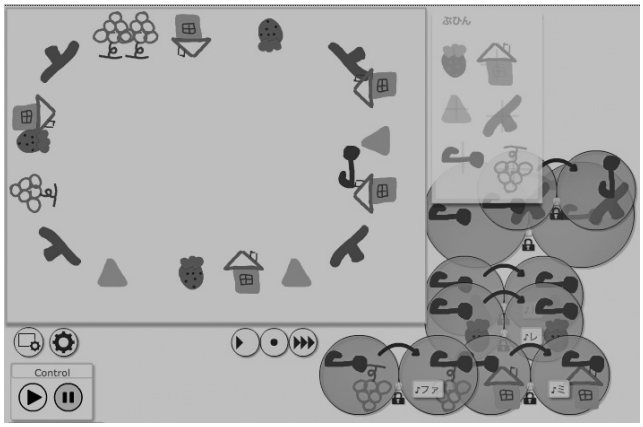

図7　オルゴール

☆1 ビスケットは曖昧な判定が特徴であるが，この応用の場合には厳密な判定で動作するモードに切り替える．曖昧なままだと，動かしていてどんどん画面が壊れていくからである．

何か」を説明しているわけではない．どちらかというと，ビスケットを用いて，体験と同時に簡単に説明できることを説明しているに過ぎない．ほかにも説明すべきことはたくさんある．たとえば，数値の配列で表現される，さまざまな計算や精度のよいシミュレーションなどは，中学生にとって非常に重要なコンピュータの応用であろう．

コンピュータのあるアプリが我々の生活を変える，ということだけではなく，コンピュータそのものの本質的に持っている力を広く開放し，一般の人がそれを自由に使いこなせる時代，それを目指したい．

参考文献

1）Harada, Y. and Potter, R.: Fuzzy Rewriting–Soft Program Semantics for Children–HCC 2003，IEEE（Oct. 2003）．
2）原田康徳．加藤美由紀．Potter, R.: Viscuit：柔軟な動作をするビジュアル言語，WISS（2003）．
3）原田康徳：体験型ワークショップ用ソフトウェアの開発，第 50 回プログラミングシンポジウム，情報処理学会（2009）．
4）原田康徳，勝沼奈緒実，久野　靖：公立小学校の課外活動における非専門家によるプログラミング教育，情報処理学会論文誌，Vol.55, No.8（2004）．

原田 康徳（非会員）viscuit@gmail.com
1963 年生．1992 年北海道大学大学院情報工学専攻博士後期課程修了．同年日本電信電話（株）NTT 基礎研究所．2000 ～ 2015 年 NTT コミュニケーション科学基礎研究所．1998 ～ 2001 年 JST さきがけ研究員．2004 ～ 2006 年，2010 ～ 2013 年 IPA 未踏ソフトウェアプロジェクトマネージャ兼務．2015 年 4 月より合同会社デジタルポケット代表．博士（工学）．ワークショップデザイナー．

採録決定：2015 年 2 月 26 日
編集担当：西田知博（大阪学院大学）