ビジュアルプログラミング言語ビスケット（Viscuit）の紹介

原田康徳

1　はじめに

ビスケット [8] [9] [10] [11] [12] は子ども向けプログラミング教育用のツールとして 2003 年に開発された．絵の配置を書き換える規則を与え，対象を書き換えて行く様子をアニメーションで見る．これまで様々な実践を通した改良を重ね現在に至っている．

ビスケットの開発の歴史は，開発者の立場や興味の変化によって，大きく 3 つのフェーズに分かれている．一つ目はプログラミングを小学三年生程度に教える立場，二つ目はプログラミングが絵画や作曲といった芸術表現のメディアと同列のものとして扱おうという立場，三つ目はワークショップという実施形体を考慮したシステムとしての立場，である．単純なプログラミング言語からシステムやその教え方に至るまでが大きく変わってきている．

本稿では，まず，第 2 節ではビジュアルプログラミング言語としてのビスケットに付いて解説する．ここで述べている言語仕様に関しては開発当時からほとんど変わっていない（むしろ退化した部分もある）．第 3 節では，ビスケットが芸術表現のメディアとして，何を考慮されてきたのかについて述べている．プログラムを作るのが主ではなく，作品を作るためにプログラムを使うという立場である．第 4 節では，ビスケットがワークショップという複数人数での活動に広がったことに対して，教え方やシステムのサポート

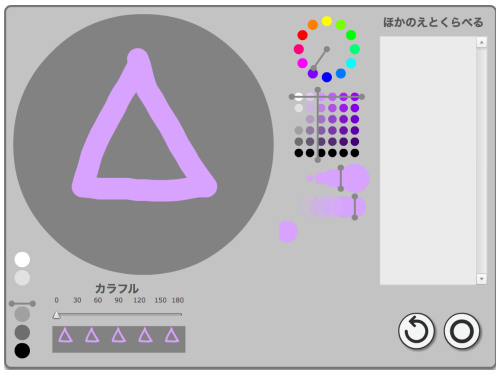


図 1　ビスケットでの描画

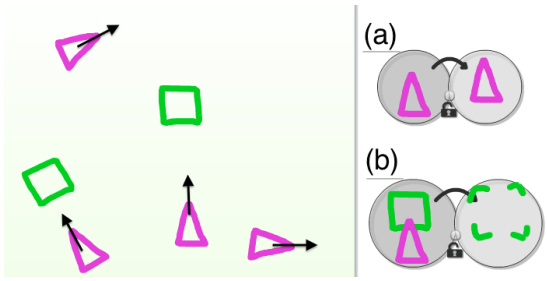


図 2　プログラムと実行画面

といった点について述べている．

2　ビジュアルプログラミング言語としてのビスケット

ビスケットは，書き換え規則に従って，絵の配置を書き換える，図形配置書き換え言語である．プログラムの実行結果は，図形の配置の時間的変化となり，それがアニメーションやゲームといった応用になる．

図 1 に描画ツールの様子を示す．このツールで描か

Introduction of Visual Programming Language Viscuit

Yasunori Harada, NTT コミュニケーション科学基礎研究所, NTT Communication Science Labs..

コンピュータソフトウェア, Vol.0, No.0 (0), pp.0–0.
[研究論文] 2013 年 12 月 15 日受付.

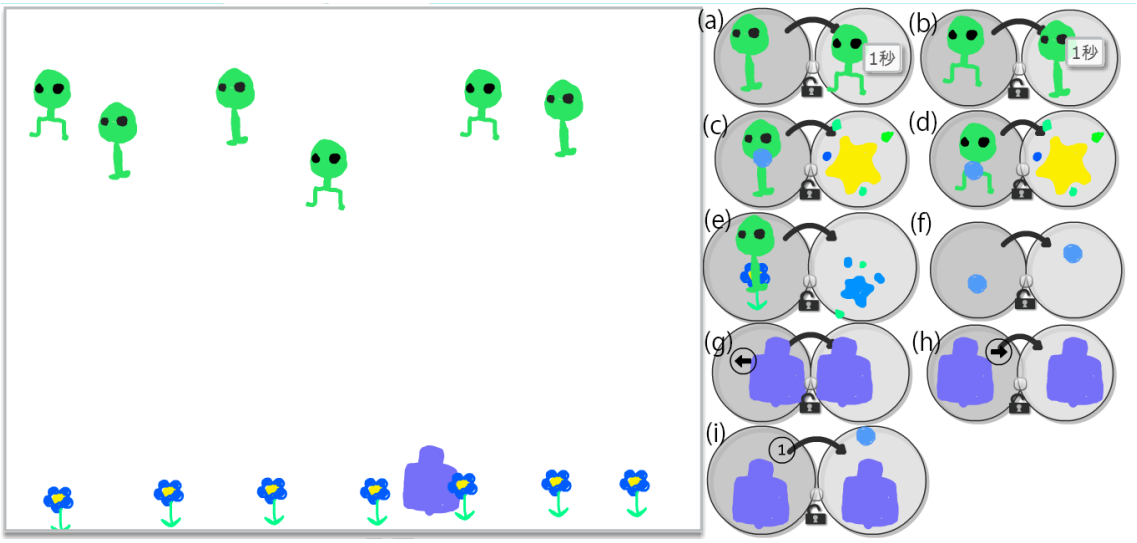


図 3　ゲーム（小２女子が１年目に制作）

れた絵がビスケットのプログラムの「部品」となる．ここでは三角形を描いている．図 2 はプログラム（右側）とその実行画面（左側）である．プログラムは「めがね」と呼ばれる書き換え規則で定義される．めがねに部品を入れることで，書き換え規則が作られ，実行画面におかれた部品が書き換えられる．書き換え規則 (a) では，左の円（書き換え前）と比べ右の円（書き換え後）の方が三角形の位置が上にずれているため，この規則が繰り返し適用されることで，画面内の三角形は矢印のように移動していく．

ビスケットでは，複数の絵を含んだめがねを用いることで，複雑な書き換えを実現できる．たとえば図 2 の (b) のめがねは，「三角形が四角形に刺さると，四角形が分解し，三角形は消滅する」という動作を表す．

図 3 に制作当時小学校 2 年生の女子（ビスケットの経験 1 年）の作品を示す．各めがねの役割は以下の通りである．

(a),(b)　宇宙人が足を開→閉，閉→開と変化させながら下降してくる（「1 秒」はそこで 1 秒待つ命令）

(c),(d)　宇宙人に球が当たると爆発する（宇宙人の二つの絵に対してそれぞれ必要な点に注意）

(e)　宇宙人が花に接触すると花が枯れる

(f)　球は上に向かって移動する

(g),(h)　砲台は左右矢印ボタンにより左右に動く

(i)　砲台は「1」ボタンで球を発射する

このプログラムは，単純な構造ながら，宇宙人から花を守るゲームというストーリが表現され見事である．このプログラムが示す通り，単純な動きを組み合わせることで大きな作品が簡単につくれることがわかる．

ビジュアルプログラミング言語のなかで，書き換えを基本にしたものは幾つかある．BITPICT [3], VISULAN [6] はビットマップ書き換え言語である．BITPICT は GUI のアニメーションをビットマップ書き換え規則で記述できることを示した．VISULAN はマウスのボタン情報を表現するピクセルの導入によりマウスイベントによるインタラクティブな書き換えに成功している．ChemTrains [1], VISPATCH [4] はベクトルグラフィックの書き換え言語である．ChemTrains は図形の包含関係をマッチングで判定できるようにし，論理的な問題を記述している．VISPATCH はメタレベルの書き換えを許し，自己拡張可能な図形エディタを記述した．Kidsim [2] (後の StageCastCreator [7]) は格子状のアイコンの配置書き換え言語である．純粋な書き換えだけでなく，アイコンの内部状態や条件，アクションを導入し複雑なシミュレーションを記述できる．AgentSheet [5] は厳密には書き換え言

語ではないが，アイコンの配置で発火する規則を決めている．規則の内部はスクリプトで記述している．

これらと比較すると，ビスケットは，回転，スケール，平行移動が自由に可能なアイコンを対象とした書き換え言語と位置づけられる．Kidsim ではアイコンは格子状に並んでいたため，書き換え規則に対して厳密なマッチングが可能であった．しかしビスケットではアイコンの配置は連続的であり，書き換え規則と厳密に一致させることが難しい．そこで，書き換え規則のマッチングに曖昧性を導入した．複数の書き換え規則のなかで，アイコンの配置関係の一致度が一番高い書き換え規則が選択され，それが適用される．書き換え規則と書き換え対象とが完全に一致しないため，それらのズレを書き換え結果にも反映するようにしている．この仕組みにより，単純な指令で複雑で深みのある動きが生成できるようになっている．この曖昧なマッチングと曖昧性を保存した書き換えを Fuzzy Rewriting と定義した．

3 芸術表現のメディアとしてのビスケット

ビスケットが開発された最初の年 (2003/10) に，ICC というメディアアートの美術館でビスケットの最初の展示が行われ，同時にビスケットのワークショップを初めて実施した．著者は芸術やワークショップに関する知識がまったくなく，周りに言われるままにやっただけである．その時のわからなさを解消したく，その後，美術大学の通信教育課程に入学した．そこで得た著者なりの考えを次のバージョンのビスケットに反映させた．

3.1 作られた作品が一番目立つべきである

それまでのビスケットでは，プロのイラストレータが描いたかわいいイラストをインタフェースやあらかじめ用意された部品などに多用していた．よいイラストをつかうと，高級感・完成度感が高まり，ソフトウェアの価値が高まった．しかし，この上で作品を作ってもその作品が埋もれてしまった．

そこで，次のバージョンではイラストなどはすべてやめ，全体の色使いやインタフェースのボタンなどを非常にシンプルで地味にした．世の中の，子ども向けアプリのほとんどが派手な色使いで，操作ごとに派手な効果音やアニメーションが提示されるのに対して，このビスケットの地味な路線はある意味時代に逆行しているかのようであった．

ところが，この新しい地味なビスケットの上で作品が作られると，急に画面が生き生きとし，自分が作った作品が一番目立つことがわかった．ツールが目立ってはいけない．

3.2 理想的な画材を目指す

当時，CREST[†1]などでコンピュータを芸術に利用する研究が盛んに行われていた．それらの中には，工学部的なセンスで，従来の芸術を変えず（ブラックボックスのまま）に，コンピュータがそれをどう支援するかという視点で研究されていたものがあった．たとえば，油絵の具とまったく同じ挙動をするペイントソフトなどである．そういった研究の意味はそれなりにあるとは思うが（UNDO が出来る油絵の具の需要），子どもの教育という視点では別の発想が必要ではないか．油絵の具や水彩絵の具など様々な画材は，物理的な特性から非常に多くの制約を持っている．そしてその制約は美術教育の視点とはまったく別のものである[†2]．せっかくコンピュータが使われるのだとしたら，理想的な画材とは何かを追求すべきであると考えた．ビスケットの描画ツールは本当の意味で理想を追求しているとは言いがたいが，アプリケーションのバランスを考えて実装している．ある大学の先生（幼児教育のご専門）からは「どうしてビスケットで描いた絵はきれいなんですか」と聞かれたことがあるが，以下にその理由となるソフトウェアデザイン上の工夫について述べる．

†1 科学技術進行機構 戦略的創造研究推進事業（CREST）

†2 友人のお子さんのエピソード．幼稚園で水彩絵の具で夢中になって描いていたら，絵が真っ黒になってしまった．先生はその暗い絵をみて，この子は心に闇を抱えている，と勘違いした．水彩絵の具は色が混ざると暗く濁るので，きれいな絵で完成させるには，丁度よいところで止めるという高度な判断が必要である．作り込み過ぎると悪くなるというのは画材としての欠点である

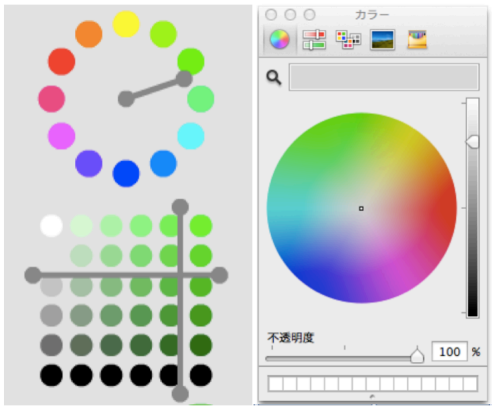


図 4　色選択ツール

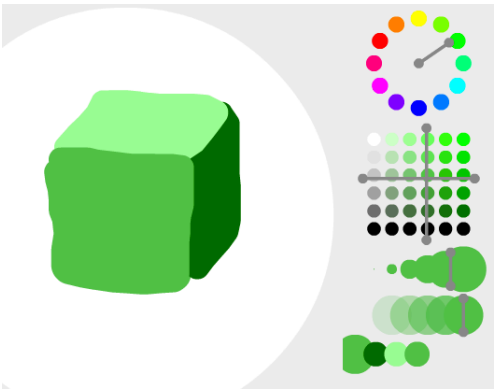


図 5　影のある絵

3.2.1　完全な色選択ツール

市販の子ども向け画材（24 色クレヨンなど）の色の選択が多いに不満である．多くの子ども向けお絵描きソフトがその点をまったく考慮せずにクレヨンの真似をした色選択になっている．子どもに身近でよく使われる色を選んでいるのだと思われるが，色の原理からすると，茶色や肌色†3といった色は，橙色を基準にして考えると，それを暗くしたのが茶色，白っぽくしたのが肌色という，応用的な色である．同じように青，赤，緑，黄色，紫といった基本的な色に対しても，茶色・肌色のような応用的な（暗くした，白っぽくした）色は存在する．100 色入りくらいの色数の多い画材なら，そのような色も扱われているし，それらの色にも茶色や肌色ほど有名ではないけれど名前はつけられている．しかし，少ない色数のときに，何の色を採用するのかは，実に難しい問題である．

ビスケットの色選択ツールは，すべての色が対等に扱われるようなインタフェースである（図 4 左）．まず，上の円で色相を選択する．その下に，明度と彩度を同時に選択できるコントローラがある．色彩学では明度と彩度の組をトーンと呼ぶ．絵を描くときや色彩的なデザインをするには，色相とトーンで考えることが基本となる．トーンがそろっているというのは，たとえば，明度を最大にし，彩度を落とした（白っぽくした）トーンは，ピンクや水色などであるが，それらの色はパステルという色のグループとなり優しい印象を，明度と彩度を最大にしたトーンは，ビビッドとなり，南国的な生き生きした印象を作ることができる．ビスケットでは絵を描く際に，色相は独立で，明度と彩度は同時に調整できるため，トーンがそろった絵を描きやすい（明度と彩度を意識的に触らなければ，自動的にトーンがそろう）．

また，逆に，色相を固定して，明度と彩度を操作すると，光や物質の反射の質感を表現することができる．たとえば，ある色相で，明度を中間ぐらいにして形を描く．その形の影の部分は，色相を固定のまま，明度を下げて描く．また明るく光が当たっている部分は明度を上げて，すこし白っぽく彩度をさげて描く．この描き方で，立体的に飛び出して見える形を簡単に描くことができる（図 5）．

ちなみに，明度と彩度を同時に，色相だけ独立に操作する，というインタフェースは自明ではない．色相と彩度を同時にし明度だけ独立，といったインタフェースを採用している市販のソフトも多い（たとえば MacOS に付属のペイントソフト　図 4 右）．

3.2.2　色選択の表示部分

色選択のコントローラの表示にグラデーションを使わないのも意識してデザインした部分である．多くのペイントソフトではこの部分がグラデーションで表示されている．このコントローラは滑らかに操作できるので，実際に選択できる色は連続値だが，画面の表示の上では代表的な色だけ表示させている．

色の実際の見え方は，その周囲の色によって簡単に

†3　肌色は，現在では「ペールオレンジ」「うすだいだい」と名前が変わっているが，無くなってはいない

騙される．錯視である．グラデーション表示では，色が連続しているため欲しい色の認識が難しい．ビスケットのコントローラでは白に対して明確に領域を切って色を提示することで，正確な色が感じられるようにしている．また，赤と橙の中間よりすこし赤に近い，といったアナログ的な表現が直接でき，混色の直感的な理解にもつなげられる．

ビスケットの色選択ツールは，最初から色彩に関して，明確な意志を持って描ける人にとっては，操作が最小になり使いやすいインタフェースであり，色彩に関する意識が低い人にとっては，ビスケットで絵を描いていると，知らず知らずに色空間の直感が身に付く，ということを想定して設計されているのである．

3.3 初期値の制御

ペンの色，太さ，背景の色といった値の初期値に関して面白い課題がある．ここで初期値というのは起動後に何も操作していない時の値を言う†4．

3.3.1 機能の探索を促す

初期値のペンの太さをわざと太いものにしている．小さい子どもには彼らの描画の制御能力と比べるとちょうど良い太さであるが，緻密な絵を描きたい高学年や大人には太すぎる．最初に何も考えずに描いた線が太すぎると感じると「描いた線を消す機能」「線の太さを変更する機能」を探そうとする．最初から初期値がちょうど良い太さであれば，線の太さを変更する機能を探さずに，最初のままの太さでずっと描いているだけかもしれない．最初に小さな失敗をさせ，それを自力で乗り越える癖をつけさせる．それによって，様々な機能や概念に自然と興味をもってもらう．

3.3.2 複数人の作品のバランス

絵をこだわって描く人は，色や太さを変更する機能をさがして，すぐに使いこなすが．使う色を気にせず，最初に与えられた色で描いてしまう人も多くいる．そんな作品が大量に増え，それらを合わせてみたときに，全体としての色のバランスが悪くならないようにしたい．そこで，起動時のペンの色は，色相は 360 度で均一な乱数，明度と彩度は 0.5　1 の範囲の乱数で与えている．これによって，後述のビスケットランドのように，全員の作品をあわせたときに，どんな人もその全体の作品を良くする方向で参加できることになる．

4 ビスケットの教え方

ビスケットで子どもたちにプログラミングを教えることがどういうことなのかを考えるようになった．一般的な教育では，理解させたい目的を定めめ，それが達成されたかどうかで教育を評価する．ビスケットでプログラミング教えることは，プログラミングの技術を習得してもらうということだったのか．

ビスケットの設計当初から，伝統的なプログラミング教育のスタイルでビスケットを教えることに違和感があった．伝統的なスタイルとは if や for といった構文や変数，配列，サブルーチン，関数といったプログラミングの構成要素を順に教えて行くものである．そんな知識をいくら詰め込んだところでコンピュータの本質には迫れるとは思えない．早期プログラミング教育の反対派の中には，自分が学生時代に受けたこのような知識詰め込み型のプログラミング教育の影響を受けている人も少なくない．その誤解の解消も中々大変である．ビスケットを開発した動機は，そういったプログラミングスキルの習得ではないし，言語に必要な構成要素を教えるものでもない．

むしろ，コンピュータに対して様々なレベルでの接し方を許すようにしたい．たとえば，小さい子は自分の描いた絵が自分の命令によって動くという経験ができればよい．さらに方向や速度の制御，二コマの絵の変化ができ，絵と絵がぶつかった時のインタラクションを導入することで，プログラムの可能性が爆発的に広がることを知る．これらはどのレベルであっても，それはビスケットの目的である「コンピュータとは何か」いうことの理解が一つ深まったことになる．

このような非常に幅の広い目標を持つ教育のスタイルの一つとして「ワークショップ」が近年注目されている．ワークショップは様々なレベルの参加者に対し，参加者同士のコミュニケーションを促すことによって，一斉授業や個別授業では得られない幅広い効果を期待するものである．ビスケットの教え方も，参

†4 この項目は論文構成上この節に入れたが，実際には現場の観察により考察された機能である．

図 6　ビスケットランド（お菓子の国）

図 7　床でビスケットで遊んでいる様子

加者同士のコミュニケーションや教え合いを促すように進めている．

4.1　コミュニケーションを促す工夫

ビスケットの入門として非常に人気のあるワークショップにビスケットランドがある．グループごとにテーマを決めて，そのテーマにそった絵を描いて動かし，完成したら送信する．送られたアニメーションは即座にグループごとの画面に反映し世界が構築されてゆく．図 6 はお菓子の国をテーマに作られた作品である．それぞれの絵は一人ひとりの子どもたちが描いて動かしたものである．

グループ画面を介したコミュニケーションが生まれる．誰かが描いた作品の影響を受けて他の子がそれに続く作品を作ったり，真似をする．一つの作品を制作するのに速い子では 1 分くらい，ゆっくり考えても 5 分もかからないので，制作共有のフィードバックは，非常に素早く行われ，グループとしての作品の質が向上する．このコミュニケーションは言葉を介するだけでなく，無言のままでの関わりもある．

機材としてはタブレットを使うことが多いが，机の上ではなく床にタブレットを直接置いて使わせることもある．こうすることで，他人の画面を簡単に見ることができ，作品のフィードバックだけでなく，操作の教え合いなどがしやすくなるようになる．

ビスケットの教え方の手順も，コミュニケーションを促す工夫がされている．たとえば，重要な機能をわざと説明しない．子どもに聞かれても，すぐには教えないで，少し自分たちで探させる．自分で見つけた機能は他人に教えたくなるし，教えることで自分の理解にもつながる．幼稚園児が得意になって自分の親に使い方を教えているのは，よく見る光景である．

4.2　ワークショップを支援する機能

ここで，ワークショップを支援するシステムの機能について説明する．ここでは，前節で説明した厳密な定義のワークショップに限定せず，学校で行われるビスケットの授業や，ショー的なイベントも含んだ形式についての，システム的な支援の説明である．

ワークショップでは，最後の発表会で各自が作った作品を全員でが見える大きな画面に表示させるということがよく行われる．初期のビスケットではそういった利用がいっさい考慮されていなかったため，作品の表示は長い VGA ケーブルを抜き差しして対応していた．また，終了後にお土産として各自が作った作品を渡すことがあるが，初期のビスケットでは作品はローカルな PC に保存されているため，保存された作品を USB メモリーで吸い出して，それぞれを CD-ROM に焼いて配ることで対応していた．当然ながらそれらの作業には熟練したスタッフが必要である．ちなみに，これは 11 年前の話であるが，現時点でもワークショップを考慮していない一般的なソフトウェアでのワークショップはこのやり方が主流である．

その後，PC を無線 LAN で接続し，OS のファイル共有機能を利用して，ファイルを一カ所に集める作

業は自動化できた．

ビスケットの開発言語を C++から Flash に変更し，システム全体を Web ベースのグループウェアとした．ローカル又はインターネット上に設置した Web サーバにプログラムや初期データを置き，クライアントはブラウザを通じてそこにアクセスし，作品は HTTP の POST によって Web サーバ上に書き出される．これによって，現場でギリギリまでビスケットの改良をし（その以前は改良の度に再インストール），初期データも簡単に編集できる．

前述のビスケットランドのグループの共有画面も Web のクライアントとして実装されている．データの流れとしては，リアルタイムチャットを Web で実装したことと同じ構成である．

その結果，ワークショップ，イベント，展示などで非常に柔軟に展開でき，たとえば，今終わったばかりのワークショップで作られた作品が，あらかじめ作っておいた Web ページの内側に張り込まれ，ワークショップでの制作が終わると同時に世界に向けて作品が公開される．といったことを簡単な HTML レベルのプログラミングで実現した．

ビスケットランドでは，進行中に各グループが制作するテーマを決める，といったことも良く行われている．これもサーバ上のファイルを操作し，ブラウザを再読み込みすればよいだけなので簡単に実現できる．一般に，アナログのワークショップに比べてデジタルのワークショップは柔軟性において不利であるが，それをかなり克服できている．

子どもは色々な機能を勝手に試し，それで勝手に混乱する．それは全体の進行の妨げになるので，不必要な機能は表示されないようにしている．参加者のできることが増えるにしたがって，新しい機能が現れるようになっている．こういった UI が現れたり消えたりする仕掛けも，現場で修正したい．一般的な GUI プログラミングでは XML で画面やボタンなどの配置を定義するが，ビスケットではその GUI の配置の定義自身も作品の中に含まれている．wiki で文書の中に特殊なタグを挿入するとそこに対応するツールが登場するようなインタフェースに良く似ていて，テキストを入力する隠し操作によってその場で GUI のツールを出現させたり配置を変更したりできる．その配置をテンプレートとしてサーバ上に保存し，それを参照して起動するクライアントは新しい GUI に変更される．たとえば，参加者全員の理解がある一定のレベルに到達したとき，作品のテンプレートに絵を拡大縮小するといった新しい機能の UI を追加すると，参加者が次に新しい作品を作るときに，その機能が追加される．

4.3 改良のサイクル

ビスケットは 100%著者一人で開発している．システムやインタフェースの改良のヒントはすべてワークショップの現場の観察から得ている．多くのユーザが無意識に取ってしまう行動をみて，改良する．改良はソフトウェアだけではなく，アイコンのデザイン，配置，ワークショップの進行，機材に貼る保護テープなど様々である．たとえば初心者には不要な機能は，最初の段階では表示されないようにし，進行に従って機能が増えて行くといったことなどである．これらは，すべてスタッフの手間を最小にし，スタッフの能力のばらつきに影響されないためである．

たとえば，3 日間連続で開催されるイベントでは，初日に出たトラブル（これらは現場でその都度スタッフが対応する）は翌日までに修正され，二日目は作品の質を高めるための指導法を工夫する余裕ができ，三日目にはシステム，進行ともに完成する．指導法の工夫のためには，システム側からのサポートも不可欠である．良い作品，落ちこぼれが出ない進行のための演出なども，システム側からのサポートが必要である．

こういった改良の結果，現在では開始から 20 分で，平行移動，2 コマのアニメーション，回転移動を使った作品を一人で複数個制作して，全員で鑑賞するところまで可能にしている．

4.4 指導者養成

ビスケットを改良して行く過程で一つの目標にしていたことは，ビスケットの指導に高度なスキルを求めずとも，いくつかの例題さえあれば自分で学ぶことができるようにアプリを使いやすくする，ということである．このような目標は，ビスケットのユーザビリティを一定の高さに保つのに貢献したと思われる．

しかし，実際のワークショップやイベントなど，実施時間の制約がある場合，全員が楽しんで制作できるためには，ある程度トレーニングを積んだ指導者が必要となる．トレーニングといっても適切なフィードバックを得ながら，現場で指導する回数を多くする，ということ以外にはない．

改良はたとえば次のような流れで行われている．

1. 初回に子どもから A に関する質問が多く出る．
2. その次の回で，A に付いて解説を追加する．
3. A の質問は減るかわりに B のトラブルが出る．
4. さらに次の回では，A と B を適切なタイミングで説明する．
5. トラブルや質問は減るが，作られた作品がどれも似たようなものになってしまう．
6. 説明しすぎないように，言葉を減らし，例を工夫する．

ビスケットの指導では，失敗をすることは基本的にはない．説明が上手でなければ，その後の制作時間で同じ質問を大量にうけ，その対応に他のスタッフが時間が取られるだけである．そのため，ワークショップの指導法の訓練として，ビスケットはよい題材になっている†5

小学校の放課後教室での事例は [12] に詳しい．ここではいくつかのアナログ的なツールを組み合わせて，特に大人からの強い指導はせずに，児童たちの自主性を活かして，教え合いを促す環境づくりをしている．

5 おわりに

ビスケットが開発され 11 年になる．その間にイベントやワークショップに参加してくれた子どもたちは 1 万人近くになる．自分の作ったソフトウェアが多くの人たちに使われ，その様子を目の前で見られることは非常に嬉しいし，やりがいがある．もともとは，純粋な研究目的で始まったプロジェクトであるが，そういったユーザの反応という魅力にとらわれて，ここまで続けてしまった．

ビスケットは Web サイト http://www.viscuit.com から無償で使用することができる．今後は機能を安定化させ，iOS や Android のタブレットの対応などで，より一層の普及を目指して行きたい．

†5 青山学院大学で開講している，ワークショップデザイナー育成プログラムにおいて，ビスケットを用いたワークショップが見学教材として使われている

参 考 文 献

[1] B.Bell, and C.Lewis : ChemTrains: A Languages for Creating Behaving Pictures, VL ’93.

[2] A. Cypher, and D.C. Smith : KidSim: End User Programming of Simulations, CHI ’95.

[3] G.W. Furnas : New Graphical Reasoning Models for Understanding Graphics Interfaces, CHI ’95.

[4] Y.Harada, K.Miyamoto, R.Onai: VISPATCH: Graphical rule-based language controlled by user event, VL ’97.

[5] Repenning, A. and J.Ambach, Tactile Programming : A Unified Manipulation Paradigm Supporting Comprehension, Composition and Sharing, VL ’96.

[6] Yamamoto, K.: 3D-Visulan: A 3D Programming Language for 3D Applications. Pacific Workshop on Distributed Multimedia Systems (DMS96), pp.199-206, 1996.

[7] Stagecast Creator : `http://www.stagecast.com/` .

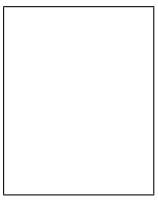
[8] Yasunori Harada, Richard Potter : *Fuzzy Rewriting – Soft Program Semantics for Children –*, HCC 2003, IEEE (2003).

[9] 原田康徳, 加藤美由紀, Richard Potter: Viscuit:柔軟な動作をするビジュアル言語, WISS 2003 (2003).

[10] 原田康徳，加藤美由紀: Viscuit:柔らかい書き換えによるエンドユーザ向けアニメーション記述言語, 第 45 回プログラミングシンポジウム, 情報処理学会 (2004).

[11] 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).

[12] 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 (Aug. 2014).



原 田 康 徳

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式会社 NTT 基礎研究所．2000 年 NTT コミュニケーション科学基礎研究所．1998 年-2001 年 JST さきがけ研究員．2004 年-2006 年，2010 年-2013 年 IPA 未踏ソフトウェアプロジェクトマネージャ兼務．博士 (工学)．ワークショップデザイナー．