2005－HI－116（7）
2005／11／17

子供向けビジュアル言語 Viscuit とそのインタフェース

原田康徳
NTT CS研

子供でも簡単にプログラムを楽しむことができる Viscuit(ビスケット)を紹介する．Viscuit は，柔らかい書き換えという原理に基づいて図形の配置を変更することで，アニメーションを生成する．プログラミングと実行のモードがない，エラーを出さずにとにかく実行する，など，それを使用する人にとって敷居ができるだけ低くなるように，設計をした．本稿では，これまでの使用経験を交えながら，特に子供向けとして注意して設計した部分を中心に，述べる．

A Visual Programming Language Viscuit and its Interface for Kids

Yasunori Harada
NTT Communication Science Laboratolies

We introduce Viscuit, a visual programming language for kids. Fuzzy rewriting that executes programs softly is its main innovation. It is appropriate for kids because the programming mode and execution mode are unified such that no error is possible for any program or input. Typical programs produce animations by repetitive execution of rewrite rules. This article presents details of Viscuit's design considerations for kids and reports on several experiences.

1．はじめに

子供にプログラミングを修得させることは，非常に難しい課題である．これまでにも，いくつものソフトウェアが開発され，それなりな成果を上げてきているが，まだ，成功しているとはいえない．プログラミングそのものが難しいだけでなく，それ以外の部分，たとえばキーボードが打てるとか，マウスの右クリックやダブルクリックが必要，といったインタフェース的な障害がある．

また，根本的な問題としてプログラミングのどの部分を教えるべきか，といった点でも大きく意見が分かれている．概念の部分だけの理解を重視する立場もあるし，最終的には Java といった一般的な開発に使われている言語へ繋げられるような理解を重視する立場もある[3]．その違いによって，対象とする子供の年齢層やカリキュラムも異なるので，設計時の想定と異なった使い方をすると効果が低い．

Viscuit（ビスケット）[1,2]は，子供や情報処理の専門家を目指さない大人に対して，プログラミングの楽しさを伝えるために開発された．コンピュータはプログラムを取り替えれば様々な道具に変身するし，自分でプログラムを作れば思い通りに動かすことができ

る．これがコンピュータの最大の特徴である．しかし，いま世の中に普及しているコンピュータはそのような使われ方をしていない．単なる情報の入出力の端末であり，他人が作ったソフトウェアをインストールして使うだけのものである．我々は，すべてのユーザが本格的なプログラムを作れるようになる必要はないけれど，この最大の魅力を少しでも知って欲しい，と願いビスケットを開発している．

また，子供の表現力の向上という点も一つの目的である．絵を描いたりするのと同じように，プログラミングも一つの表現手段である．絵は最低限筆を持つ能力で表現できるが，音楽のためには楽器を演奏する能力が必要である．誰でも演奏できる楽器を発明するのと同じように，誰でもプログラミングできる言語を発明する必要がある．優れた楽器は修得が容易で表現力も豊かであるのと同様にプログラミング言語もそのような性質が求められる．

2．ビスケット

ビスケットは子供や情報処理の専門家を目指さない大人に対して，プログラミングの楽しさを伝えるために開発された．ビスケットは絵を並べることでプログラムが表現され，絵を並べたとおりに動作させることができる．プログラムの動作は絵の動きによって確認することができる．

2．2．ビスケットの設計目標

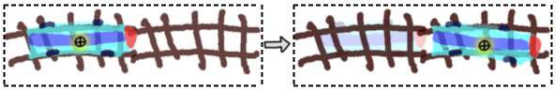
ビスケットの設計目標を以下に述べる．

1．最初の1分で結果がでること
単純な例でよいので，とにかくコンピュータの別の面白さを感じてもらう．

2．どんなプログラムとデータの入力に対してもエラーを出さずに，とにかく動作すること
エラーを出して，それを直すという作業は，コンピュータに対する悪いイメージを与えてしまう．エラーの原因は些細なことが多く，その繰り返しはコンピュータ嫌いを育てることにつながる．エラーを自動的に直すというアプローチがあるが，我々は，そもそもエラーが存在しないシンタックスというのを追求する．

3．修得の途中に障壁がなく，ある程度複雑なレベルまで滑らかに学習できる
ユーザの興味や目標，集中度の持続具合などで到達点は変わってくるだろうが，知識やヒントを少しずつ与えることで，先に進めることを考える．

以上を達成するために，この目標をもう少し具体化する．



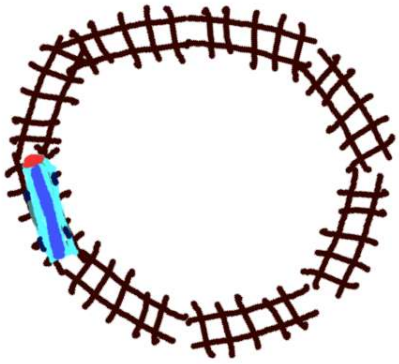


図 *1* 線路と電車

図 1 はビスケットを設計する以前にひとつの目標として考えていたプログラムとその実行例である．どのように実行するかはともかく，このように絵を並べてどんなどうさをして欲しいかを最初に考えたのである．プログラムは上の2つの矩形内に並んだ図形で構成されている．ここでは線路が 2 本つながっていて，

その上に電車があったとき，電車は次の線路に進む，という意味のプログラムとして解釈する．このプログラムに対して，線路が何本か並べて，その上に電車を置いて，電車をクリックすると，電車が線路の上を走り出す．

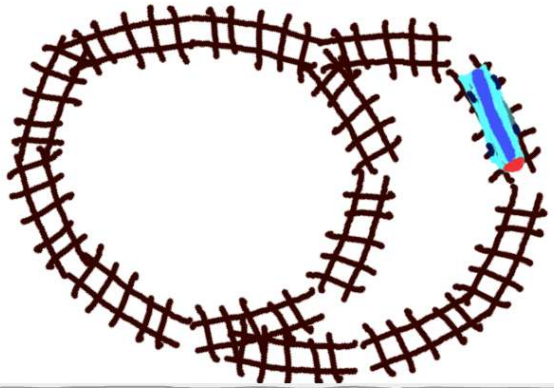


図 *2* 線路の分岐

線路の分岐があるときは，線路がまっすぐに並んでいる方を選択する（図2）．線路を取り外したり置いたりすることで，好きな方向に電車を走らせることができる．

このように，プログラムは非常に単純であるが，実行の対象に依存して，プログラムをどのように解釈するかは変化する．上の例では，最初プログラムは電車が線路の上を進むという意味で解釈していた．しかし，線路の分岐では，まっすぐ進むほうを選択するという意味も，最初のプログラムから読み取る．ちなみに図1のプログラムを線路が曲がった状態で配置すると，電車は分岐において曲がりを優先して選択する．

2．3．やわらかい書き換え

さて，目標とするプログラムの表現と実行の仕様は大体決まったが，次にそれをどのようにして実装するか，という問題を考える．3．のある程度複雑なところまで滑らかに学習できる，という目標のためには，その場限りの仕組みの組み合わせではなく，スケーラブルな性質をもった計算法が必要である．

そこで，「やわらかい書き換え」を提案する．

「やわらかい書き換え」は，「実行例による書き換え」と言い換えてもよいだろう．コンピュータに希望の動きの例を与えて，その例を参考にしてコンピュータが動作する．例を沢山用意すれば，それだけ複雑な状況でも動作させることができる．コンピュータが知らない状況のときでも，限られた例を拡大解釈してとにかく動作する．

この仕組みは2つのステップに分解できる．まず，今の状況がどの例に一番近いかを探し，次にその例で与えられた動きを今の状況にあわせて修正することである．この計算の詳細は省略し，その性質の述べるだけにとどめておく．

やわらかい書き換えは，与えられた例と厳密に一致する状況のときには，与えられた例と同じ動作をする．したがって，厳密に一致する範囲では従来のプログラミング言語と同じ性質を持っており，たとえば，チューリングマシンを作ることができる．しかし，今の状況と一致する例がない場合に，もっとも近い例を拡大解釈して実行する，ここが拡張された部分である．

この性質からもわかるように，例が多い場合には状況に一致する例を見つけられる可能性が高くなるため，一般の言語に近い振る舞いとなる．それに対して，例が少ない場合には，これまでのコンピュータらしくない振る舞いとなる．

2．4．ビスケットの例

この節では，ビスケットを用いて子供にプログラミングを教えることを想定しながら，順に説明してゆこう．

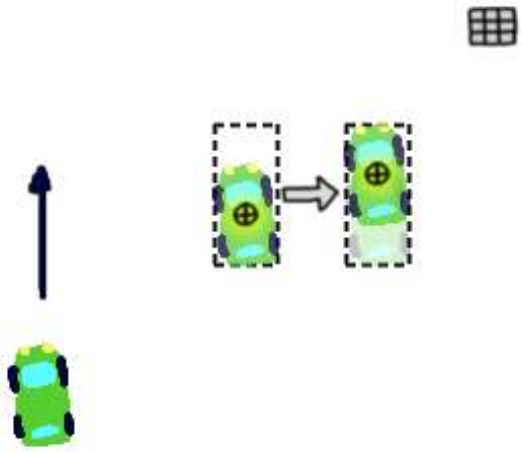


図 *3* 車は進む

最初にグリッドアイコンを画面上に置いて，マウスドラッグで絵が不用意に回転しないようにする．次に，ルールの矢印を置いてから，左右に車の絵を入れる．さらにもう1つ車を置いてそれをクリックすることで，車が進みだす．ルール内の車のずれを調整すると車が進む速度が変化する(図3)．

2つの信号機の絵が交互に変わるルールをつくる(図4)．変化するタイミングを調整するために，大きな時計をルールに入れている．

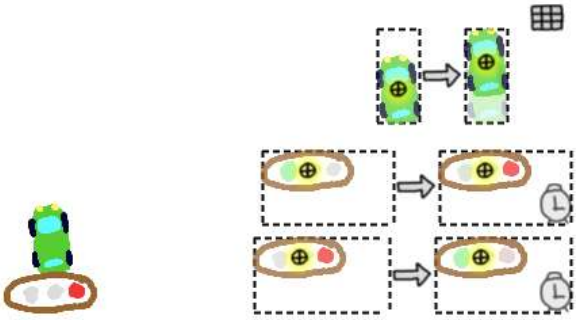


図 *4* 信号機はゆっくり変わる

信号機と車を適当に配置して，車が赤信号のときには動かない，という意味のルールをつくる．車がその通りに赤信号で停まることを確認する(図5)．

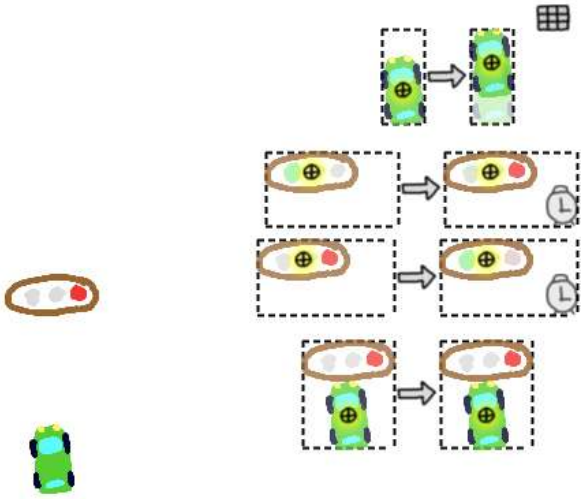


図 *5* 信号機で停まる

次に，信号機のかなり手前で停止している車を信号機ギリギリで停止するように修正する．すでに信号機の直前では動かないというルールがあるので，さらに信号機の少し前のときはゆっくり進むというルールを追加すればよい(図6)．

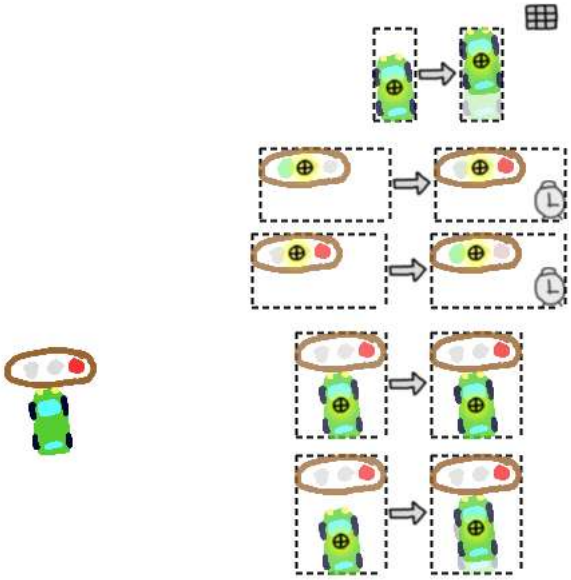


図 *6* 信号の前でゆっくり停まる

なぜ，1つのルールのときは，信号機の手前で停止していたかというと，やわらかい書き換えの拡大解釈のせいである．厳密に信号機の直前でなくても，ある程度手前ならば，信号機の手前と解釈しているからである．

新しく追加したルールは少し手前ではゆっくり進むという情報を追加したことになるので，前のルールの信号機の手前がより厳密な意味として解釈され，信号機の手前でピタッと停止する．

ここで車が 2 台に増えたときのプログラムを考えてみる．ここまでのプログラムだけでは，2 台の車は赤信号の前で重なって停止する．それに対して次のルールを追加する．これによって，信号機の前に車が直列に並んで停止するようになる（図7）．

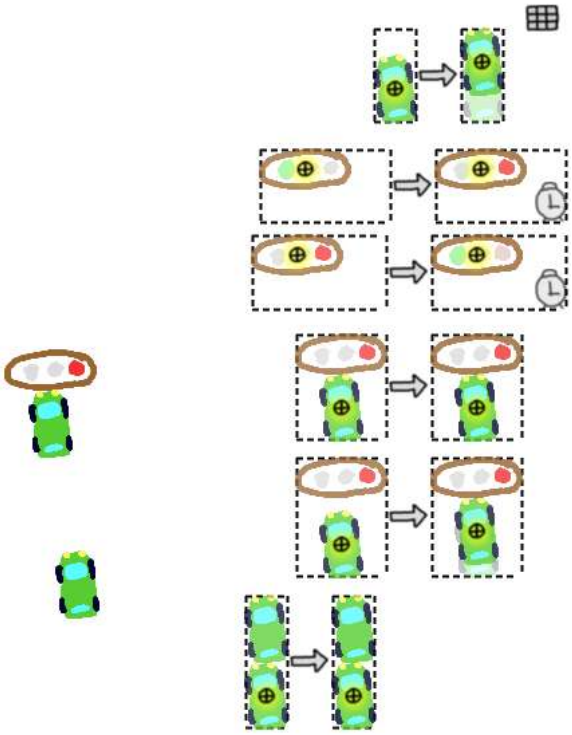


図 7 前に車があると停止

前の車との停止の距離を短くするには，同様に「車にぶつかりそうなときはゆっくり進む」というルールを追加する．図8のルールは，図7をより厳密にする働きだけでなく，2 台の車の車間距離を保つ働きもする．ぶつかりそうなときと，図3のなにもないとき，のどちらとして判定されるかで，車の速度が変る．

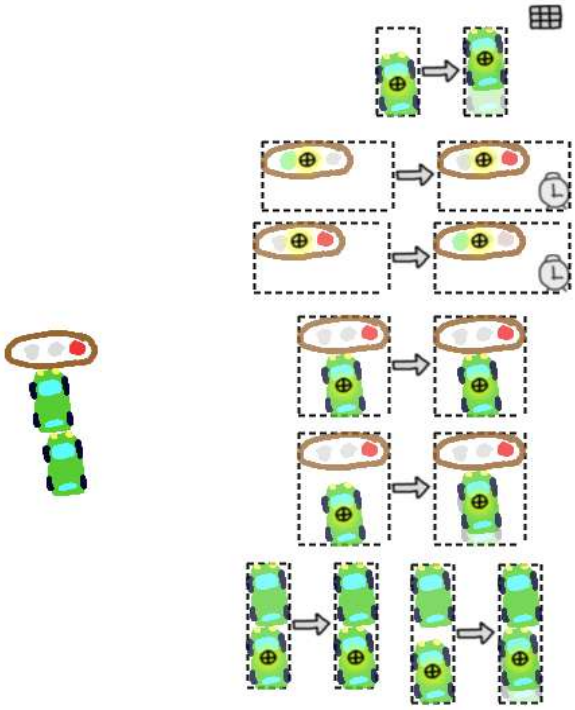


図 8 ぶつかりそうになったらゆっくり

これら一連の例は，一つずつルールを追加して行き，その都度，全体の動きが詳細化されている．つまり，上手に小さな連続した目標を設定できれば，小さなステップでかなり複雑な例までも，滑らかに学習できることがわかる．

3. ビスケットの使用例

造形教室の生徒（小学校5年5名と3年生1名）に対して行った2回の授業（各2時間ずつ）について報告する．

1回目はビスケットの使い方，遊び方などを，2節の例などを元に，比較的自由に使わせた．そこで一通り教えた使い方に関して，1週間後の2回目ではテスト形式で授業を行った．テストはルールそのものを隠し，動作から各人がルールを推測して，それを自分のコンピュータ上で試してみる．造形教室の生徒である特徴を活かして，後半の問題は，自分で描いた絵を使って答えさせるようにした．

以下その問題と解答のコツ，狙いを列挙しよう．

1. 男の子をクリックすると手を上げる．もう一度クリックすると手を下ろす（クリックで絵が変わるということを理解してもらう）．
2. 女の子をクリックすると手を上げて，すぐに手を下ろす（クリックマークで連続して動く．多少絵の位置がずれてもここでは気にしない）．
3. 赤い車真っ直ぐ進む．青い車はもっと速くまっすぐ進む（ルールの絵のズレが速度に影響する．できるだけ真っ直ぐすすませるが，多少回転してもよい）．
4. しゃくとりむし．虫が伸びた絵と縮んだ絵を描いて，これを交互に切り替えて進む（切り替わる際に，絵の動く部分と動かない部分を意識させる．伸びるときは，尻尾が動かない．縮むときは，頭が動かない．虫の向きを真っ直ぐにしないと動きながら曲がる）．
5. パックマン．口を開けている絵，閉じている絵，えさの絵を描く．口を開けたままパックマンは進む．えさが前にあるとそれを食べる（「食べる」は口を閉じるルールと口を開けるルール．この例は様々な発展がある．口をパクパクさせながら進む．えさの増やし方（このままだと減る一方）．口を開くときに向きを変える（4つのえさをどう配置したら順番に食べるかを競うゲーム（ビリヤードのようなもの））．
6. 魚が海草をよけて泳ぐ．魚の絵と海草の絵を描く．魚が進む．魚の下で海草がぶつかると魚は上によける．魚の上で海草にぶつかると魚は下によける（3つのルールが必要．普段の動きと，下によける，上によける．それぞれの魚のズレ（泳ぐ速度）が同じくらいでないと，滑らかに泳がない．ルール右辺で，魚にクリックマークを付けるのを忘れると，肴の動きが止まる．海草はぶつかったときに動かないようにするべきだけれど，ぶつかるとよけるようにされるのも面白い）．
7. 電車と線路．電車と線路の絵を描く．線路の左端に電車があるとき，線路の右端に電車が移動する．線路をまるく並べると，それにそって電車が走る(絵に沿ってものを動かす方法．線路には左端，右端という方向があるので並べるときにその順序を守らないとそこで停まる．ルールの左右で，電車にクリックマークがついていること．ルールの左右で線路が動かないように並べること）．

1.2.3.に関しては子供達は集中力を切らさない時間で正解することができた．4.は虫が伸びた絵と縮んだ絵の長さの差が充分大きくないとまったく進まないということでつまずいている子供がいた．頭と尻尾の動きは，普段からしゃくとりむしをちゃんと観察していないと難しいのかもしれない．

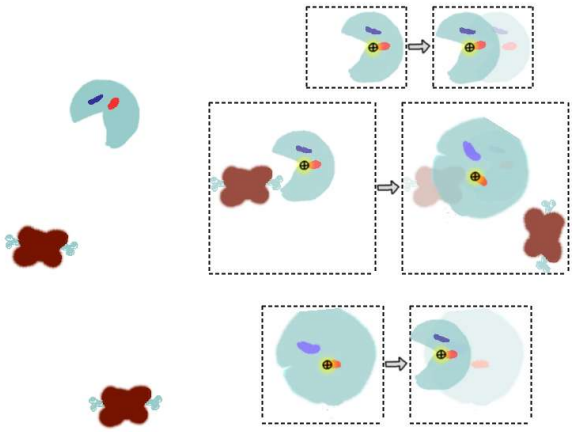


図 9 えさを食べたらすぐに生成する

5.は時間が許せば子供達はいくらでも遊び方を発明する非常にいい例であった．図9はその一つのバリエーションで，えさを食べ

た後に，すぐにえさを生成する．最初に与えた問題では，画面上でえさがすぐになくなってしまうため，どうやったらえさを増やせるかということを考えるきっかけになる．その一番簡単な解が食べたらすぐに生成するということなのであった．

ここまでの問題の正解率が思った以上に高かったため，6.7.はこの当日に急遽準備した例である．そのため，理解の差が明確に出た．しかし，7.はこのとき初めて見せた動きであったが，一部の生徒は自分で解答を作り出すことができてしまった．

動きを見せたとき，その動きの各ステップを分析できると，その動きを生成するルールを作り出すのは，子供でも不可能ではない，ことを示している．どのように教えてゆけば，多くの子供にその能力を身につけさせることができるか興味深い．

4．考察

ビスケットを使って複雑なアニメーションを習得してゆく過程は，子供たちにとって何を理解したことになるのだろうか．思いつくままに述べてみる．

現実のコンピュータの仕組み，たとえばメモリーとCPUといった知識は得ることができないが，コンピュータが持っている本質的な性質は直感的に理解することができるであろう．たとえば，プログラムを作成する場合はその動作を想像しながら行う．そして，実際の動作を確認してプログラムの修正を行う．逆に一度作ってしまうと，何度でもそれを動かすことができる．これがコンピュータのプログラムがもつ本質的な難しさと便利さである．また，条件判定や繰り返しといった基本構造も直感的に理解することができる．

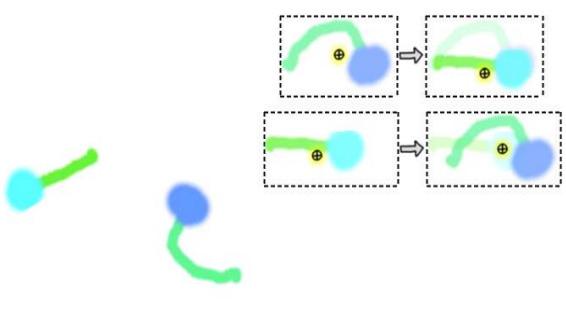


図 *10* うまく進まないしゃくとりむし

図10はうまく行かなかったしゃくとりむしの例である．最初に用意した虫の絵は，伸びたときと縮んだときで頭と尻尾の変化が少ない．そのため，アニメーションはなかなか前に進まない．本当は伸びた虫の絵はもっと長くかかなければならないのであった．

この例がうまく作れるためには，普段からしゃくとりむしの動きを注意深く観察していなければならないのである．

このように，ビスケットは，ものの動き方が簡単に再現できるツールなので，動き方の学習に最適である．これは従来のパラパラアニメのツールでは実現できなかった性質でもある．

注意深く設計したにも関わらず，やはり操作上の問題のトラブルは避けられない．課題のレベルに適したインタフェースを考える必要がある．たとえば，2節の一連の自動車の例ではグリッドアイコンを使用して図形の回転をさせないようにしている．

他にも，1つの絵が2つに増える，というルールを作るってしまうと，画面に絵がどんどん増えてしまう．これは一つの教育的効果（細菌の増殖，ネズミ講の理解）を狙ったものであるが，その現象が突然生じるのは進行上好ましくない場合もある．そのため，ルールに入れられる絵の個数の制約も一部取り入れられている．これにより，うっかりとしたミスを減らしている．

5. まとめ

Viscuit（ビスケット）を子供や情報処理の専門家を目指さない大人に対して使ってもらうために，設計思想から，具体的な使用例を述べた．ビスケットはコンピュータのある部分を非常に簡単にしたけれど，それ以外の部分（たとえば，ファイル操作，Web への公開など）に関しては，まだ従来のコンピュータのままである．この機能までも簡単に扱えるようにできて，やっと全体が完成すると考えている．

参考文献

[1] 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.

[2] Y. Harada, R.Potter: Fuzzy Rewriting – Soft Program Semantics for Children -, HCC 2003, IEEE.

[3] Squeak: http://squeakland.jp/