02

基般

小学生に分かるコンピュータサイエンスとしてのプログラミング教育
―ビスケットを用いて―

原田康徳（デジタルポケット）

プログラミングでなければ学べないこと

プログラミングはコンピュータを使うすべての人が学ぶべきである．しかし，この主張に説得力を持たせるためには，プログラミングによって何を学ぶことができるのかを説明できなければならない．たとえば，プログラミングを学ぶことで，創造性であるとか問題解決力であるといったさまざまな能力が身に付く，という意見がある．これは間違ってはいないけれど，これではすべての人が学ぶべきという主張を裏付けるまでにはいかない．それらの能力が身に付くのはプログラミングを学ぶことによる副次的な効果であり，せいぜい，効率良く身に付くとか，同時に身に付くといった程度である．プログラミングでなければ教えられないことではない．

プログラミングでなければ教えられないことは唯一「コンピュータとは何か」ということである．しかし，ただ単純にコンピュータとは何かという問いでは，コンピュータ上のアプリを数多く使いこなすことでなんとなく得られる「コンピュータはこんなもの」という感覚的なものも含まれてしまう．それらと明確に区別するために，より厳密に「プログラムを書くことによってしか得られないコンピュータ観」としよう．それが定義できたとして，そのコンピュータ観は義務教育の貴重な時間を奪い合ってまでみなが知る必要のあるものかどうか，そこが問われる．

「プログラムを書くことによってしか得られないコンピュータ観」とは具体的にどういったことであろうか．コンピュータの専門家たちは，自分たちがあまりにも専門に慣れすぎて，それを知らなかったときのことを思い出すことが難しい．逆にそれらを知らない人たちは，自分たちが何を知らないのかを知らない．したがってこれらを明確にズバッというのはなかなか難しい．

ビスケットで教えているコンピュータ観

ここでは，筆者らが小学生に対して行っている授業やワークショップなどの活動を例にして，我々が考えている「プログラムを書くことによってしか得られないコンピュータ観」とはどういうものか説明する．この試みは始まったばかりであり，非常に未熟でこれから多くの人たちの手によって膨らませていくべきものであることは重々承知している．しかし我々の活動を通して，その内容も少しずつ増えてきているので，本稿で紹介する．以下で示すコンテンツはいずれも，実際にワークショップや授業でビスケットを用いて実施してきたものである．なお，ビスケットについては本稿では丁寧に説明する余地はないので，文献 1）などを参考にされたい☆1．

1） プログラムで動くものとは

次のコンテンツはビスケット未経験の状態から 5 分くらいで実施できるものである(画面の様子を**図 -1**に示す)．

> 棒人間を描いてください（図 -1 左）．棒人間をステージにたくさん入れてください．
> メガネを持ってきて，メガネの右と左に棒人間を入れると，ステージの棒人間が一斉に動き出します（図 -1 中）．
> メガネの中の棒人間の位置をずらすと，ステージの棒人間の動きが一斉に変わります．ずらし方で，動く方向と速度を制御できます．
> メガネの中から棒人間が出てしまうと，メガネが壊れ，ステージでの棒人間は一斉に動きを止めます（図 -1 右）．
> では今度は，皆さんが考えて，棒人間を水平方向に横に動くようにしてみてください．

☆1 最小限の説明をすると，「メガネ（2 つの○が並んだもの）の左側とマッチする絵を，右側の絵＋配置に書き換える」ことがビスケットの基本動作であり，すべての動きはこれから作り出される．

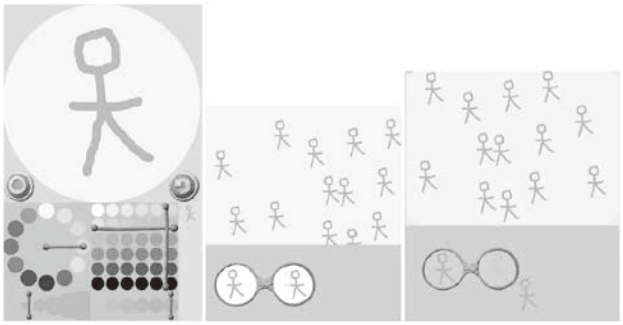

図 -1　多数の棒人間を動かす

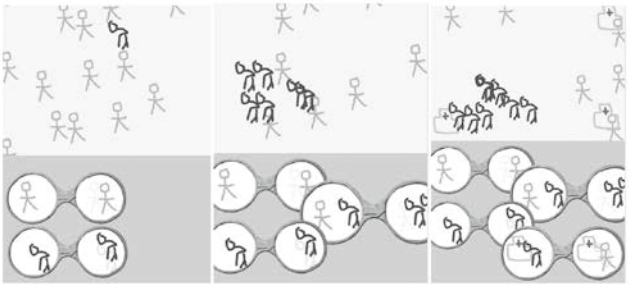

図 -2　病気の感染のシミュレーション

ここまでを，実際にタブレットを使って一人ひとりに体験をしてもらう．小学校低／中学年を対象にこれを実施すると，いつも子供たちは動きに夢中になり，歓声があがる．このような簡単な体験ではあるがこれだけから次のようなことが説明可能である．

1. 1つのメガネが何度も使われ，複数の棒人間が一斉に動く．
2. メガネを修正すると，その変更が全体に一瞬で反映され，一斉に動きが変わる．
3. メガネが壊れると，全体が一斉に動きを停止する．

これは，プログラムによって作られたものと，これまでの物理的なものとの違いを非常に明確に見せている．たとえば自動車のリコールというのは壊れるかもしれないけれど，必ず壊れるというものではない．それに対してプログラムの不具合は起きるかもしれないではなく，必ず起きる．コンピュータの何かの不具合でシステムが一斉に停止するという事件は頻繁に起きている．不具合の修正もプログラムの場合は簡単にすべてのコンピュータに送ることができるが，物理的なものの場合は1つ1つ直していかなければならない．最近では製品が出荷された後から，プログラムを更新して機能を向上させるといったこともよく行われている．

我々は大人向けの講座の場合にこのような裏側の説明はするが，小学生向けの体験では，このような説明はせず感覚としてコンピュータによって作られるものとはどういうものなのかを知ってもらうようにしている．子供たちがこれをどれくらい理解しているのかを知ることは難しいが，これに続くほかの作品作りでも，複数の絵が自分が気に入った動きを得られるまでメガネを調整しているので，すんなりと受け入れているといった印象である．

1. に続いて，次の説明を提示する（**図 -2**）．

> もう1人元気のない棒人間を描いてください．その人を1人だけステージに入れ，上に動くようにします（図 -2 左）．
> 最初に描いた棒人間は健康な人で，後から描いた元気のない棒人間は風邪を引いています．健康な人と風邪を引いた人がぶつかると，風邪がうつる，というメガネを作ってください（少し間を置いて自分で考える時間をとる）．
> 答えは，このようにします（図 -2 中　その結果，1人だけだった風邪がどんどん増えて行く）．
> 次に，病院を建ててください．（少し間）病院を建てるということは病院の絵を描いて，病院をステージに入れ，病院に風邪の人がぶつかると治るというメガネを作る，ということです（図 -2 右　病院になかなかぶつからず，治りが遅い．病院を増やすと少しバランスがとれる）．

この体験で4つのことが説明できる．「プログラムの複雑さはどこからくるのか」「ものと情報の違い」「仕様からプログラムを作るということ」それと「シミュレーションとは」である．

2）複雑さはどこからくるのか

メガネ1つ1つは単純なことしか指令していない．ここでの4つのメガネは，健康な人が横に動く．風邪を引いた人が縦に動く．健康な人と風邪を引いた人がぶつかると，2人とも風邪を引いた人になる．病院に風邪を引いた人が入ると，病院から健康な人が出てくる．というものである．それぞれは1つのメガネで表せる簡単な命令でしかない．しかし，風邪がどんどん広がったり，病院で治ってもすぐにまたかかってしまったり全体の振舞いはきわめて複雑である．

コンピュータには最初から複雑な命令が入っているのではない．単純な命令しか理解できないけれど，その単純な命令を組み合わせることで複雑な動きが作れるのである．メガネがたった 4 つでもこれくらい複雑な動きになるが，メガネをどんどん増やしていくことで，世の中にある便利なすごい動きをするプログラムに近づいていく．

3）ものと情報の違い

感染は指数関数的に広がる．これに対し，病院では 1 回に 1 人ずつを治すので，線形関数的であり治療はまったく追いつかない．ものを他人に渡すと自分からはなくなる．ものは移動するだけ．それに対して，情報は他人に教えても自分からはその情報はなくならない．たとえば，他人に美味しいラーメン屋さんのことを教えても，自分はそれを忘れたりしない．風邪の感染も同じで相手に風邪をうつしても自分は治らず風邪を引いたままである．風邪の感染も実際には情報の伝達を模倣しているものと考えることができる．相手に教えても自分からはなくならない場合，このように指数関数的に一気に広がっていく．広がり方がすごい．これは情報が持っている基本的な性質である．情報には良い情報と悪い情報があるとすると，良い情報がすごい勢いで拡散するのは良いことだけど，悪い情報がすごい勢いで拡散するのは悪いことである．情報によって作られた社会，インターネットには基本的にこの原理を持っているのだから，それをよく踏まえた上で使いこなせるようになるべきである．

4）仕様からプログラムを作るということ

「病院を建ててください」という一言で，すぐに何をすべきか分かる人が 1 割くらいいる．この一言は 3 つのステップ「病院の絵を描く」「病院をステージに置く」「病院で風邪が治るというメガネを作る」に分解されるが，一言で分かった人というのは，この 3 ステップへの分解ができる人である．仕様からプログラムを作るということは，人間が分かる文章からコンピュータが理解できるステップへの変換を意味する．プログラミングというのはこの繰り返しである．

5）シミュレーションとは

健康な人が移動する速さ，風邪を引いた人が移動する速さ，病院の数などを変えると，感染の広がり方が変化する．このようにいろいろと変えて実験することがシミュレーションである．子供や大人に対して，シミュレーションはコンピュータの重要な応用の 1 つであり，ほかにどのような例があるか（天気予報や大きな建物を建てるときといった）を説明している．そして（特に大人に対して）プログラミングが自由に書けることの強力な利点として，シミュレーションが自由自在にできるようになること，を説明している．

6）脳の拡張としてのコンピュータ

シミュレーションの有用性を実感するには，結果がすぐ直感的には分からない題材が適している．風邪の感染の例の後に，次の課題が続く（**図 -3**）．

> 風邪の例では，健康な人が一方的に風邪にやられてしまいました．今度は絵を 3 つにして，それぞれがやられる 3 すくみの関係を作ってもらいます．1 つの例がじゃんけんです．たとえばグーとチョキがぶつかるとグーの勝ちなのでグーとグーになる，というものです．これを作るには，まず 3 つの絵を描いて，それぞれを動かして（3 つのメガネ　図 -3 左から 1 番目），2 つがぶつかると勝った方の絵になる（3 つのメガネ　図 -3 左から 2 番目）というのを作ります．メガネは全部で 6 つです．ステージには同じ数のグー，チョキ，パーを入れて走らせるとどうなるでしょう（図 -3 左から 3 番目）．

15 分くらいでほとんどの子供は完成させられる．これを体験したほとんどの子供は（実は大人も），何度やってもどれか 1 つの手だけになってほかが全滅してしまう（図 3 左から 4 番目）ことに首をかしげる．予想では，メガネがバランスしているのでいつまでも 3 つの部品が動き続けると思っていたからである．プログラムが間違っていないか何度も見直すが，簡単なプログラムなので間違えていないことがすぐに分かる．プログラムが間違っていないのに，自分の予想と違う結果が得られた．そこからもう一度，この 3 すくみのメガネを考えてみると，グーが頑張ってチョキを滅ぼしてしまうと，パーにとっての天敵であるチョキがいなくなるので，最終的にグーはパーにやられてしまう．つ

図 -3　じゃんけんのシミュレーション

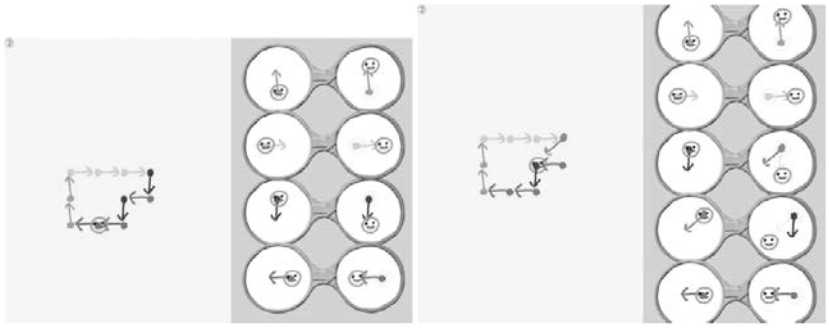

図 -4　ピタゴラスイッチ的なプログラム

まりこの 3 すくみでは最初に頑張りすぎて相手を滅ぼしてしまうと，その手は負けることになっているのである．この結果を踏まえてメガネを注意深く検討すると，たしかに理解できる．しかし，人間はそこまで最初から深く考えずに，先入観から単純にバランス良く動き続けると思ってしまう．

コンピュータが人間の脳の拡張であるといわれるのは，この点においてである．人間はあまり注意深くは考えない．そこをシミュレーションで精密に計算する．その結果は人間にとって予想外の結果かもしれない．しかし，その結果を踏まえて判断すれば，人間が注意深く考えないという欠点をコンピュータが補ってくれているということになる．

7）プログラミング言語とはどういうものか

ビスケットでピタゴラスイッチ的なプログラムを作る[☆2]．

たとえば，顔と上，下，左，右向きの矢印を用意し，顔が矢印に沿って移動するメガネを用意する（**図 -4** 左）．すると，矢印で作った経路に沿って顔が移動する．矢印を自由に並べて，好きな経路を作って遊ぶことができる．さらにメガネを追加して，矢印を通過するたびに矢印の向きが変わる，というものを作る．ここでは下矢印にぶつかると，矢印を斜めにして下に移動する．斜め矢印にぶつかると矢印を下にして，斜めに移動する（図 -4 右）．顔が矢印を通るたびに経路が変化するプログラムができた．

このプログラムは今までのものと異なり，2 つのレイヤの遊びが混在している．メガネで動きを作るという従来からの遊びと，矢印の並べ方で動きを変えて遊ぶプラレール的な側面である．矢印の並びはプログラムにとっての操作対象であると同時に，その並び自体で動きを規定しているプログラム的でもある．その場合メガネによる矢印の解釈はプログラミング言語を定義していると見なせる．

プログラミング言語というのは，それまで操作対象としてデータであったものが，並び替えることでプログラム的な意味を持たせることができて，記述のレイヤが 1 つ上がるということである．

小学校 1 年生くらいでも作業として同じものを作ることは可能である．彼らがどれくらい理解しているのかであるが，これまでとは違う種類の遊びと認識し，矢印の並び替えを自然と楽しんでいるようである（斜め矢印のアイディアは子供から出た）．もちろん彼らはプログラミング言語の存在自体に親しみはないだろう

☆2　このアイディアは東京工業大学の首藤一幸さんによっている．

けれども，自分で遊び道具を作って，その道具で遊ぶという階層の面白さと可能性については伝えられたと思う．

プログラミング言語を新しく作ることの魅力は，コンピュータそのものの可能性を大きく高め，人間が巨大なプログラムを間違いを少なく作れる理由にもつながっている．

コンピュータの理解

以上の体験による説明をもう一度並べてみよう．

1）プログラムは何度でも使われ，プログラムは簡単に修正でき，一斉に壊れる．
2）単純な命令の組合せで，複雑な動きができている．
3）情報はすごい勢いで拡散する．これは情報の原理．
4）プログラムを作るということは，人間が理解している意味を，コンピュータが分かる単純な意味へ分解する作業．
5）シミュレーションは簡単に実験できる場．
6）脳の拡張としてのコンピュータ．
7）プログラミング言語とはどういうものか．

これらのことは，どれもプログラミングができる人ならば，明確に言語化できているかどうかは別として，直感的に知っていることばかりである．一方で，コンピュータをただのユーザとして使っているだけではなかなか知ることのできないことでもある．たとえば 2）の組み合わせるから複雑になるということに対して，最近のアプリはどんどん高度化し，小さな操作でさえも細かいところにまで考えが行き届いた作りになっている．これを使えば使うほど，その裏側で作っている人の苦労は感じられなくなり，この理解から遠のくだろう．

プログラミングという体験をせずに，これらの内容を伝えることができるだろうか．筆者はきわめて難しいと考えている．というのは，いずれもプログラミング以外で体験することのない，きわめてオリジナルな体験だからである．たとえばプログラミングは料理のレシピに例えられることがあるが，料理の体験をしたからといって，上記のことが伝えられるだろうか．2）の単純な命令の組合せで複雑な動きができるという部分は，シンプルな調味料の組合せで複雑な味が作られるという言い換えはできるかもしれない．

これらは，貴重な義務教育の時間を削ってでも教える必要があるほど重要なことだろうか？　たとえば，3）の情報は原理的に拡散しやすいということを教えることは，ネット安全教室などで「XX は危険だから気を付けよう」といった言葉でインターネットを安全に使う方法を教えるよりも，ずっと効果的である．5）や 6）は，これからのほかの学習においても，シミュレーションを活用するべきであるという提案にもつながる．

残された課題

本稿ではビスケットによるプログラミングを用いて，子供たちに分かるかたちでコンピュータサイエンスを取り上げるやり方について述べた．これらがプログラミングという体験を通じなければ理解できないかどうか，これらの内容が義務教育の時間を奪い合ってまで知らなければならないほど重要なのか，これらの問いについては一切答えていない．我々，コンピュータの専門家ができることは，専門的な内容をできるだけ優しく教えられるようにすることであり，ひとまずはそれに徹する．

ここで述べてきた内容は，ビスケットを使う前提で構成されているが，ここで扱われていないけれど重要なことはまだたくさんある．たとえば TCP/IP やプロトコルの考え方など．これから多くの方々と協力しながら，進めていきたい．

参考文献

1）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 (2014).

（2016 年 1 月 4 日受付）

原田康徳 ■ 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．