情報教育とプログラミング体験の必要性
— デジ教 研Open Meeting 07 in Tokyo —

久野　靖*

2012.3.25

1　はじめに…

□ Q. 「情報教育」と「教育の情報化」は同じ?　違う?

- (A) 同じものである
- (B)「情報教育」⊂「教育の情報化」
- (C)「情報教育」⊃「教育の情報化」
- (D) 違うものである

□ 文部科学省的には、(B)「情報教育」⊂「教育の情報化」

□ 個人的には反対!　そんなの言葉の使い方がおかしい!
→ 断然 (D)

- 教育の情報化 → さまざまな教育に情報 (技術・機器) を活用
- 情報教育 → 情報 (技術・機器・情報そのもの) について学んでもらう

□ この講演では上記の定義に基づきたいと思いますがいいでしょうか…

- よろしくなければ「狭い意味での教育の情報化」という言い方にしますが…

□ Q. 「情報教育」の目的は何であるべきでしょうか?

- (A) 会社に入ったら使うであろう Word、Excel が操作できること
- (B) コンピュータやネット等の情報技術の原理・しくみを理解すること
- (C) 情報社会で危険な目に逢わない行動のしかたを身につけること
- (D) データを分析したりそれに基づく問題解決ができるようになること
- (E) ソフトウェアが作れたり開発者のキモチが分かるようになること

□ (B)〜(E) のどれも大切だと思うけど、(A) だけは勘弁してほしい!

- 高校教科「情報」の現状 → 新設時に講習で移行した非専門家教員多数
- そのうちの多くの先生は「ソフトの操作方法しか教えていない/られない」
- ソフトの操作を学んでもそこに何の原理も理屈も見出せない
- 詳細なソフトの操作をやっても生徒は退屈で嫌いになるだけ

□ (D) の手段として文書を作ったり計算するのにツールを使うので十分

□ 文部科学省的「情報教育の目標」は「情報社会を生きる力」であり、それはさらに次の 3 目標から成る (3 つとも必須)

- 「情報活用の実践力」→情報を扱え問題解決ができる (前記 D)
- 「情報の科学的理解」→原理が分かり自己評価ができる (前記 B)
- 「情報社会に参画する態度」→情報社会を知り適切に行動できる (前記 C)

□ 「実践力」は「ソフトの操作方法」と曲解されがちだがそれは間違い

□ Q. 日本のソフトウェア技術のレベルはどれくらいだと思いますか?

- (A) 米国・欧州と並んで一流レベル
- (B) 欧米より下だが中印韓あたりよりは上
- (C) 中印韓と同じくらい
- (D) 中印韓より下で三流

□ これは (D) でしょう。

□ 2006 年近辺に「日本の情報技術ってひどい」という事件多数

- 構造計算書偽造事件、フィギュアスケート誤採点事件、ライブドアショック東証取引停止、1 円 61 万株事件
- 情報処理学会「2005 年後半から 2006 年初頭にかけての事件と情報教育の関連に関するコメント」

http://www.ipsj.or.jp/12kyoiku/statement2006.html

*筑波大学ビジネスサイエンス系

□ つい最近もやってます「特許庁ソフト開発中止・55 億円無駄に」

□ 「日本はソフトの暗黒大陸」(Ruby くらいしかない)

□ なぜか?

- だいたい、日本の情報技術者は「IT ドカタ」「7K」と蔑まれてます
- 優秀な人がそんなのになりたいと思うわけがない?
- 他国では情報技術者は専門家として尊敬されるのに…

□ なぜか?

- 発注元は SI 企業に「よきに計らえ、安いほどいい」と丸投げします
- SI 企業は利益を抜いて下請けに丸投げします
- 3〜4 重の下請けの一番下で「頭数だけ揃えた」素人に開発させます
- 当然、できるものはボロボロ…(例: 特許庁)
- たまにできる人がいると皆その人におぶさって潰します

□ 結局いちばんの根本原因は…「ソフト開発など金さえ払えばできる、ドカタがやることだ」「俺はソフトなど分からなくてもいいのだ」という日本社会の風潮にあるのでは

- 海外では本当に重要なソフトは自社開発するし、アウトソース開発するにしてもきちんと設計管理する窓口部門の専門家が自社にいる
- 自社に専門家がいなかったらまともな発注も検収もできないのに、そういうことが分からないまま済ませている
- トップも発注担当も「コンピュータがどういうもので、何ができて何ができないのか」分からないまま「安いほどいいじゃん」で発注→まともな開発ができるわけがない

□ じゃあどうするか?

- 大人は今更変えられないでしょう…
- だったら、子どもの時に学んでもらうしかない! (← 今ココ)

□ そういうわけで「(E) ソフトウェアが作れたり開発者のキモチが分かるようになること」を追加させて頂きました

- 以上、長〜い前置きでした…

2 情報教育の目的達成のために

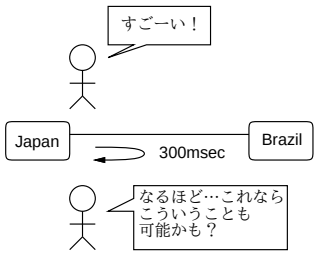
□ 再度: 目的は…

- (B) コンピュータやネット等の情報技術の原理・しくみを理解すること
- (C) 情報社会で危険な目に逢わない行動のしかたを身につけること
- (D) データを分析したりそれに基づく問題解決ができるようになること
- (E) ソフトウェアが作れたり開発者のキモチが分かるようになること

□ 命題: (D) や (C) のためには (B) ひいては (E) が必要 (ないし早道)

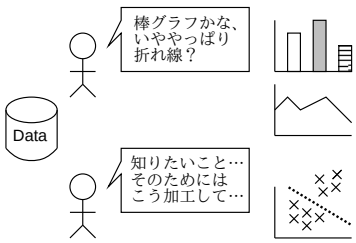
□ (C) 情報社会で危険な目に逢わない行動のしかたを身につけること

- なぜ危険な目に逢うのか→「世の中の現実 (悪い人、悪い会社)」と情報技術の「すぐにアクセス」が組み合わさっている
- 情報技術の「速く、遠くまで、たくさん」の度合いは人間の皮膚感覚を大きく超えている→結果を想像できない
- 「超えている」のを「自分の感覚にする」には、お客様感覚で「見物」してても駄目→自分が「作る人」になれば全然違う



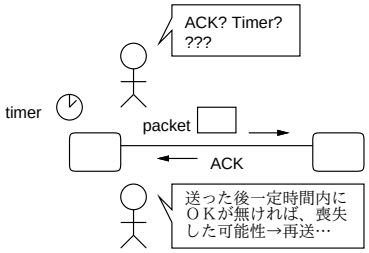
□ (D) データを分析したりそれに基づく問題解決ができるようになること

- できているグラフや数表を読むのは「統計を習った」人で OK
- 問題はそこじゃなく「どうやってデータを読める形にするか」
- それが無いと「Excel でめくら打ちグラフ」…無意味
- 「こういうデータがある」→「これをこういう風に組み合わせたり選別して、こういう変換をすれば、これが見て取れるはず」
- 「手続き的思考」そのもの!



□ (B) コンピュータやネット等の情報技術の原理・しくみを理解すること

- いくら「お話」として知識を詰め込んでも「どんな風に動いているか」とか「なぜそうなっているのか」は納得できないと思う
- 逆に「自分で作って動かすとしたら」という立場で考えるとさまざまなものの仕組みは非常に納得が行きやすい
- だから原理・しくみの理解のためには「作る人の立場になってみる」ことが早道



□ (E) ソフトウェアが作れたり開発者のキモチが分かるようになること

- これは久野が勝手に追加したものなので (笑)
- でも前述のように、わが国の将来のためにとても重要だと思います

□ …で、「プログラムを作ってみる」ことがこれらすべての目的に到達する早道でもあると思います

- 実は「(A) 会社に入ったら使うであろう Word、Excel が操作が操作できること」にも効果あり

3 プログラミングをやるとよい理由

□ 情報教育の目的達成をサポートしてくれる

□ わが国の将来のために重要 (情報技術/技術者を適切に評価)

□ それだけじゃない! 子どもたち本人のために…

□ プログラミングは「考える」活動につながる

- 現在、多くの教科が「知識つめこみ」「パターンマッチ」に陥っているとされる…(数学ですら)
- その結果、子どもたちが「考える」ことをしなくなっているとも…
- プログラミングの本質は「作りたいもの」をどう実現するか考えること→「考える経験」を担保
- 考えた結果が合っているかどうかも「思った通りに動くかどうか」ではっきり分かる

□ プログラミングは「他人とのつながり」になる

- 作ったものを見てもらう→作品観賞によるつながり
- 作ったものを使ってもらう→他人の役に立つ嬉しさ
- 共同でソフトを企画する→アイデアを出す楽しさ
- 共同で制作→協力して大きなものを作る達成感

□ 一番大事なことは…「楽しい」

- (個人的には) こんなに楽しいものはまず他にないと思っています…
- 楽しいから熱心にやる、一生懸命考える、上達する
- だから「楽しさ」が増すように/損なわれないように進めるべき

□ 大学 1 年生対象の「情報科学」と「Ruby プログラミング」を扱う授業

- 情報科学適な内容よりも「プログラミング」の印象が強い
- 最後に感想→「楽しかった」というのがやはり一番大事だと思う

4 よくある誤解

□ 「プログラミングを授業に取り入れて欲しい」というと、非常な反発を受けることがある…

- だいたいは、プログラミングをやったことがない人や、やっていても特殊な (狭い) 範囲だけでやっている人の想像にもとづく誤解

□ プログラマになるわけでもないのに、プログラミングなどやる必要はないのでは?

- ←プログラマになるためにやるわけではなく、どのようなものか「体験」できる程度にやるのでよいと考えます
- ←職業人になるためではなく、情報教育の目標達成のため、日本の将来のため、子ども本人のためにやるわけです
- ←もちろん、その方面に興味・適性のある子どもが結果としてこの道に進んでくれることは歓迎ですしわが国のため必要なことです

□ そのような特殊なことに多くの時間を費すのは適切でないのでは?

- ←目標は「プログラミングを体験してもらう」ことであり、多くの時間を費す必要はないと考えています (数時間程度?)
- ←先に述べたように「手続き的な考え方」は広い範囲で有効であり、決して「特殊なこと」ではないと思います
- ←プログラミングは「考える」という非常に汎用的かつ重要なことが学べる題材でもあると考えます

□ プログラミングは非常に難しく、初等中等教育で学ばせるのは困難があるのでは

- ←難しい (多数の学習者が脱落) というのは、手段 (使用する言語など)、目標設定 (高度なプログラム作成など)、教え方 (書き方の規則を逐一説明など) の間違いによることが大部分と思います
- ←あくまでも「体験」を目標とし、「動かして動作を確認させ、その後分かる範囲で直してみる」という教え方をすることで大部分問題なく実施できるというのが我々の経験
- ←使用言語については、新しい (教育に適した) ものが広まってきていますので、ぜひ採用を (BASIC とか LOGO とか「30 年前の」言語はやっぱりやめた方がいいかも)

5 新しい言語・環境

□ 近年のコンピュータの性能向上・情報科学の進歩→新しい言語を作ることが簡単に

□ 児童・生徒に使ってもらうためのものも多数作られている

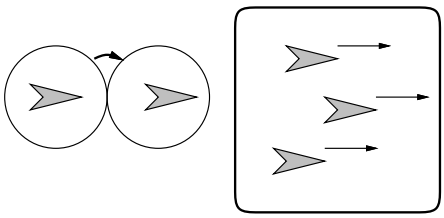
- ビスケット http://www.viscuit.com/
- プログラミン --- http://www.mext.go.jp/programin/
- アルゴロジック --- http://home.jeita.org/highschool/algo/
- Squeak eToys --- http://etoys.jp/squeak/squeak.html
- Scratch --- http://scratch.mit.edu/
- ドリトル --- http://dolittle.eplang.jp/

□ 学齢等の条件に応じ適したものを選ぶことが可能

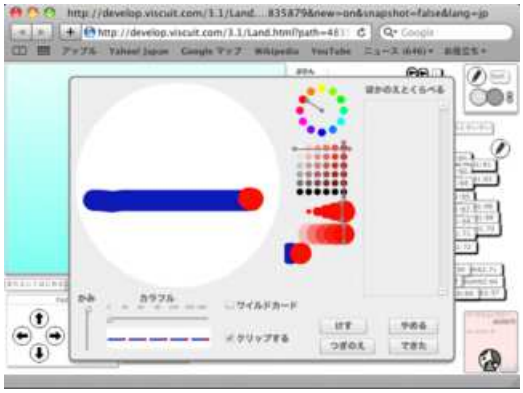
5.1 ビスケット

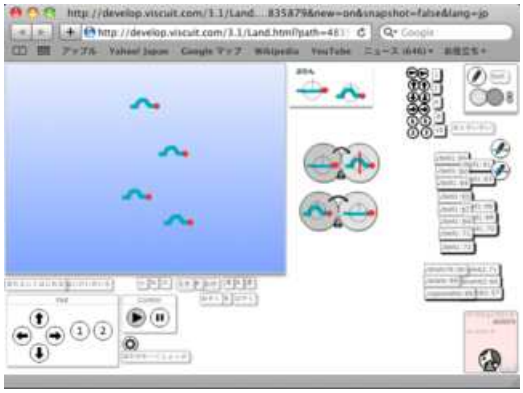
□ 原田康徳さん (NTT) が開発したプログラミング環境

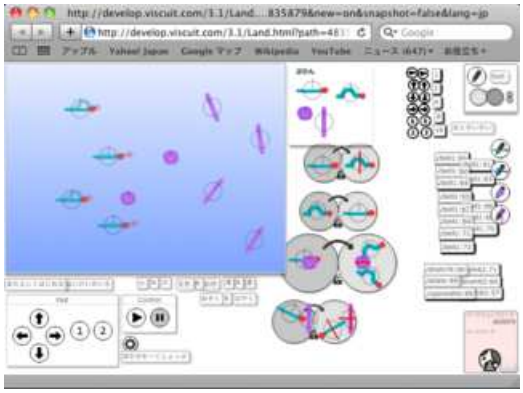
- 「絵を描く」＋「絵から絵の書き換え規則」ですべて制御



- 原理が非常に簡単、小学生で OK(ワークショップ多数開催)







□ 素朴な疑問: これがプログラミングと言えるのか? →

- → YES。プログラミングの本質は、「ある規則に従って自動実行」であり、ビスケットの「めがね」がやっていることはまさにそれ
- ビスケットで製作している子どもは、この簡単な仕組みをどのように活用して自分が思うような動きを実現するか一生懸命考える

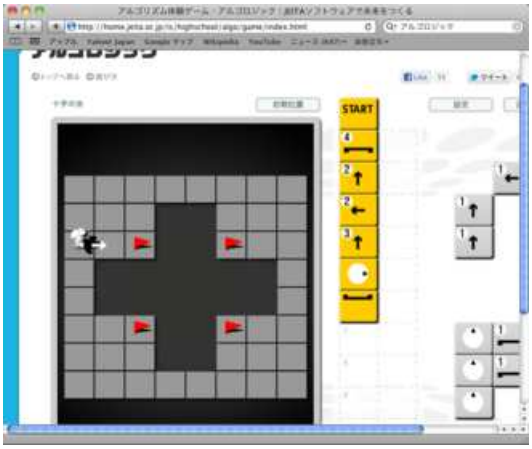
□ ビスケットの特徴

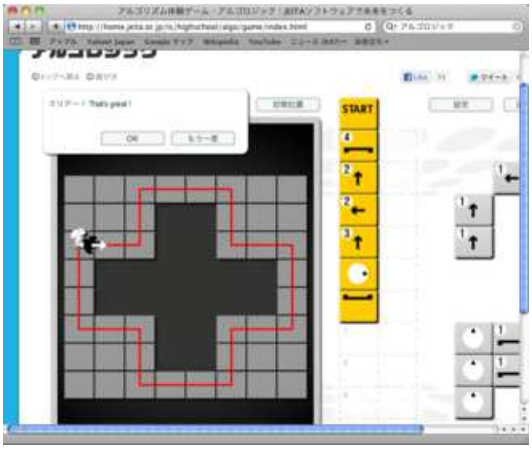
- ◎規則が簡単。低年齢から可能。作品共有の仕組み(ビスケットランド)
- △通常の言語とはだいぶ違っている

5.2 アルゴロジック

□ JEITA(電子技術産業協会)の大山さんが開発・公開した「パズルでプログラミングの概念に親しむ」環境

- 前進、横移動、方向転換、ループなどのブロックを組み合わせてロボットを動かす
- 少し慣れたら「旗取りゲーム」に進む→さまざまな通路と旗の問題が用意されている→最小のブロック数で全部の旗をクリアすると「That's Great!」がもらえる





□ アルゴロジックの特徴

- ◎できることが制約されているのでとまどいにくく、またその制約自体がパズルの要素になっているのでプラスに働く
- ◎「最小のブロックで旗をクリア」という分かりやすいお題だが頭を使わないと「That's Great!」がもらえない→必死に考える
- △できることは「ロボットを動かすこと」だけ

□ ちなみに…

- 基本版はループだけだが、「アルゴロジック2」には枝分かれもある(条件は「前が壁かどうか」のみ)
- 自由に絵を描くことだけを目的とした「お絵描きアルゴロジック」もある

5.3 Scratch

□ MITでSqueak eToysの後継として開発されている教育用プログラミング環境

- タイルを組み合わせて「文の連続」「反復」「分岐」を構成→ビスケットよりも通常のプログラミングに近い
- タイル方式の特徴→構文エラーが起きない(正しい場所にしかはめられない)
- 日本でもワークショップなど多数実施されている(小学生もあり)

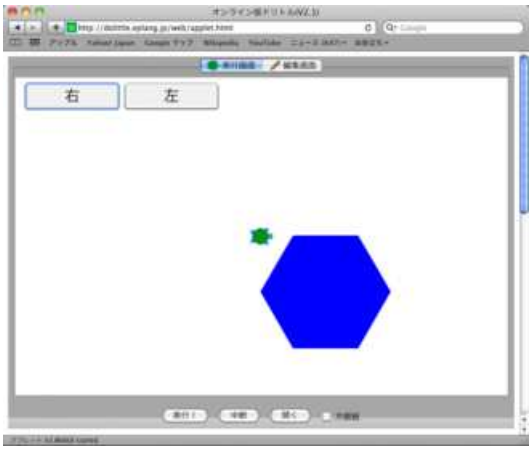
□ Scratchの特徴

- プログラムの主要制御構造が学べる
- 入出力やメッセージ(イベント)などの機能も遣える
- コミュニティ機能(作品公開)も充実
- ◎制御構造がきちんとしている→アルゴリズムを意識しやすい
- ◎構文エラーが出ない
- ◎音楽、ロボットカー制御などの発展も用意
- △タイル方式の弱点→かさばる、文字による言語とのギャップ

5.4 ドリトル

□ 兼宗・久野が開発した教育用プログラミング言語/環境

- 文字による記述を前提としている→中学校くらいからのつもり
- 日本語文字による記述、予約語 (if とか) なし
- タートルグラフィクス→ LOGO の利点をとりいれる
- オブジェクト指向に力点 (これからはオブジェクト指向が大切)



□ ドリトルの特徴

- △構文エラーを取る努力は必要 (どのみち必要になる、意味エラーはどんな言語でも同じ)
- ◎文字による記述→長いプログラムでも普通に作れる
- ◎日本語文字→小学生からでも対応可能、読みやすい
- ◎オブジェクト指向→さまざまな「部品」を提供 (GUI 部品など)、自分でも部品作成
- ◎通常のオブジェクト指向言語への移行性 (構文は違っている)
- ◎音楽、ロボットカー制御などの発展も用意

5.5 教育用言語を用いた経験

□ ドリトルの授業/模擬授業はこれまで多く見学/実施

- ビスケットや Scratch のワークショップはちょっと見学程度

□ ドリトルの授業経験で分かったこと

- 「自分がこういう風に書いたら、こういう動作になる」ということは比較的低年齢でも理解可能
- 次の段階→「このようにしたい」から「それにはこういう記述をすればよい」が思い付くようになる
- そこまで進むと「離陸」→プログラムをバリバリ書くようになる
- その段階まで「自分での理解」を着実に (ゆっくり) 進めてあげるようにする (題材をよくばらない)

□ 注意すべき (だと久野が思う) こと

- 子どもは「ソフトをやみくもに操作してみて何か起こす」ことに慣れているが、プログラミングでは通用しない
- 構文エラーの出ない言語 (タイルスクリプトを含む) では、やみくもにやって動かすことが通用してしまう→「何か動いたからいいだろう」で終わってしまい結局考えることができていない/学べていない
- この点でアルゴロジックは「旗取りパズル」なのでやみくもが通用しないところがよい
- ドリトルでは構文とか使用可能な命令やオブジェクトなどを丁寧に説明し、身につけてもらわないと「できない→興味失う」になってしまう→最初のうちはエラー探しを手伝うことが必須

6 まとめ

□ わが国の将来のために、「情報教育」は必須

- 情報技術を理解した人が社会を構成する必要
- コンピュータのプロになる人の入口が必要

□ そのコアにプログラミング「体験」をぜひ入れて欲しい

- 「楽しい」「他人とつながり」「考える力」の点でもプラス
- 現在ではさまざまな言語・環境があり多様なケースに対応可能→ぜひ先生がたも実践を!!