小学生のプログラミングの学習における言語環境についての一考察

佐藤 正範 *1*2
Email: masanori-sato@ec.hokudai.ac.jp

*1: 札幌市立豊平小学校 教諭（～H25 年度）
*2: 北海道大学 教育学院 教育方法学研究グループ（H26 年度～）

◎Key Words 小学校 プログラミング 情報教育

1. はじめに

1.1 本稿について

2012 年の新学習指導要領に基づき，中学校技術家庭科において「プログラミングによる計測・制御」が必修となった(1)。また，イギリスのナショナルカリキュラム(2)の中では，Key Stage 1 （5 歳～7 歳）からコンピューティングについて学ぶことが定められ，いよいよ 2014 年 9 月から実施されることになる。

日本でも，小学生向けのプログラミングの学習についての関心が急速に高まり，各地で様々な形態でプログラミングの学習を利用した教育的活動が行われるようになった。

本稿は，札幌の小学生を中心に，プログラミングの学習をする活動を続けてきた中で，認めることのできた成果と課題についてまとめたものである。その中でも特に，プログラミング言語についてスポットを当て，言語環境が違っても，学びにおいて共通なレスポンスとして返ってくる部分と，環境の違いが学び進める上で差異となる部分をまとめていく。

1.2 プログラミングの環境の変化

機械を思い通り動かすために，最初に登場したものは，織り機のパンチカードと言われている。コンピューターが誕生し，操作も，プログラミングを書くのも文字の入力 CUI(Character User Interface), で行われた。その後，理解しやすい単語で組むことが可能な高級言語「FORTRAN」等が作られていく。1990 年代に入り，グラフィックの機能が向上していくと，MacOS や Windows などが開発され，コンピューターが一般家庭にも普及しパソコンと呼ばれ，専門家でなくとも誰でも操作ができる GUI(Graphical User Interface)ベースでのコントロールが一般的となる。コンピューターの操作だけでなく，プログラミングもマウスを使ってパネルなどを操作するだけで組み立てることができる「Scratch」などの言語環境が誕生してきた。

※本稿では，便宜上，文字の入力を基本にプログラミングをしていく言語環境を CUI ベースのプログラミング，パネルなどを操作してプログラミングをしてく言語環境を GUI ベースのプログラミングと定義する。

前節でも述べた通り，中学校の技術家庭科で必修となったプログラミングであるが，先進各国での開始年齢が低くなっている面からも，日本の初等教育でも導入される可能性があると考えるのは自然であろう。

2. 小学生へのプログラミングの学習で導入した言語環境の紹介 –実施授業より-

2.1 RoboLab – LEGO MINDSTORMS

2004 年～2009 年まで小学校 3 年生～中学 3 年生までを対象に実施。LEGO 社のロボット教材 MINDSTORMS に付属する RoboLab という言語環境を導入。パネルを配置して結ぶ GUI ベース。

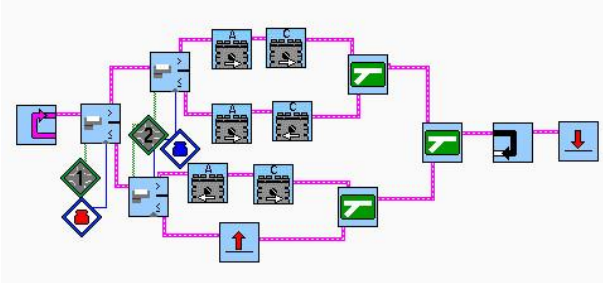

図 1 「RoboLab」 のプログラミング画面

2.2 BASIC （MBASIC）

2006 年～2009 年まで, RoboLab の学習を進める中で, 他の言語環境にも触れさせる狙いから，短期的に導入。変数，条件分岐等を文字入力で実行する経験を重ねさせた。CUI ベース。

図 2 「MBASIC」 のプログラミング画面

2.3 C++ （ROBOT C）

2008年～2009年まで RoboLab と Basic の経験を重ねた小学校高学年の子どもたちに導入。RoboLab ではできない細かな処理が可能な言語環境。CUIベース。

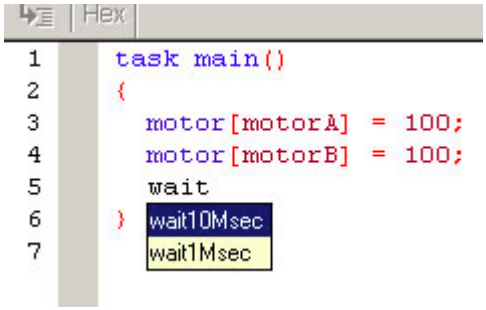

図３「ROBOTC」のプログラミング画面

2.4 C++（sketch – Arduino-）

2011年～2013年まで Arduino で LEDなどを制御する Arduino を体験するために導入。プログラミング未経験の小学校高学年対象。CUIベース。

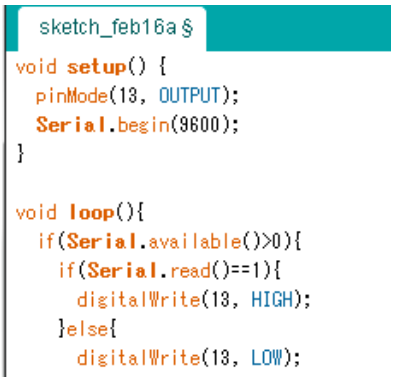

図４「sketch」のプログラミング画面

2.5 Java (Processing)

2011 年～ 文字入力がダイレクトに返ってくる描画に特化した環境として導入。CUIベース。

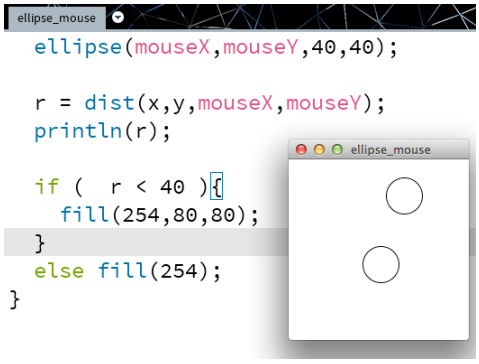

図５「Processing」のプログラミング画面

3. 小学生のプログラミングの学習における壁

小学生を対象にしたプログラミングの学習では特に以下の２つの部分がネックとなることが多い。

１． 英語入力
２． プログラミングの諸概念の獲得

3.1 プログラムの学習における英語入力について

英語入力は，ローマ字入力を学習していない低学年の子どもたち(学習者)にとって非常に大きな壁である。

プログラミングの環境そのものが英単語を基本に作られているため，英語圏の子どもたちがプログラミングの学習をする時は，コードの意味を理解するプロセスを踏まないで，学び進めることができる。

■英語圏のコードの理解のプロセス

読む → 生活経験と合わせて使い方を学ぶ

■英語がわからない場合のコードの理解のプロセス

大文字小文字を覚える → 読む → 読み方を知る →

単語の意味を知る → 使い方を学ぶ（生活経験なし）

図６ コード理解のプロセス（英語圏・非英語圏）

英語入力については，手立てを講じることで，英語がわからくともプログラミングの学習は進めることができる。その手立てを挙げると，１つは，アルファベットの大文字小文字の対応表を用意し，使用するコマンドを学習内容に応じて最小限に絞ることである。大文字小文字を使い分ける必要がある言語環境の場合には，キーボードの印字と，サンプルプログラミングの文字が大文字小文字で別であることがネックである。対応表を見ながら学びを進めていくと，すぐに対応表がなくとも R = r　G = g などを理解してしまう子どもたちには驚かされた。

別の方法としては，英語を使用しない言語環境を導入することである。２章で紹介した「RoboLab」や「Scratch」などは既にコードの入力の必要のない GUI ベースの言語環境であり，必要なコードがまとまったパネルを操作することで，誰でも簡単にプログラミングを楽しむことができる良さがある。

図７　RoboLab を使用した画像認識の学習

実際に，RovoLab を使ったプログラミングの学習では，複数センサーの制御や，ビデオカメラを使用した画像認識処理を習得し，矢印の方向や，数字を認識してロボットを制御することができるようになった。難しい制御を理解していく上でも，パネル等を使用した言語環境は有効であると言える。

しかし，英語入力という壁を定義した時に，GUI ベ

ースの言語環境の導入が壁を乗り越える方法かと言えば，そうではなく，壁の横を抜けるだけのことである。さらに，GUI ベースの言語環境を習得した子どもたちが，発展的に CUI ベースの言語環境を習得しようとする場合に，GUI に慣れたことで，文字を入力するわずらわしさと対峙することになる。GUI→CUI のギャップは，非常に大きなものになっていることになる。適切な手立てを講じ，環境を整えることで，CUI ベースのプログラミング環境もスムーズに学習を進めていくことができる。

3.2 プログラミングの諸概念の獲得について

プログラミングを学ぶという事は，コードの意味や，プログラムの書式や，原理的な逐次処理や条件分岐などの概念など，様々な概念について獲得していくプロセスを踏む事である。どの概念を，どのように獲得させるかを定めることが重要になる。

線を描く「LINE」というコードを例に挙げると，2 点 （X1, Y1）と別の点（X2, Y2）を結ぶ線を描くための書式は LINE（ X1, Y1, ,X2, Y2） となる。

LINE（X1, Y1←1 点目の座標 X2, Y2←2 点目の座標）

このコードの書式や座標の考え方を獲得させるために，座標のパラメーターだけを操作できる環境を用意し，小学校 3 年生から中学校 1 年生の 5 名に概念を発見できるかどうかの実験をした。

その結果，思い通りの線を描くことができたのは，中学 1 年生 1 名のみであった。特徴的だったのは，繰り返し描画を重ねて小学校 6年生が自分で考えた LINE のコードの意味で，

LINE（X1, Y1 ， X2, Y2 ）←線を書く
　　長さ，角度　長さ，角度

となったことである。Y 座標が大きくなる（下がる）ことを「角度が変わる」と考えたと推測できる。

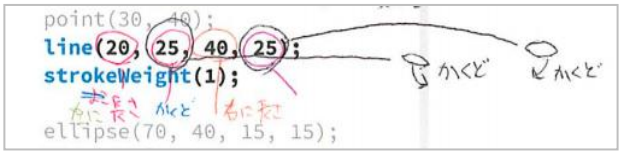

図８ 極座標と捉えた小学校 6 年生のワークシート*

また，座標の概念についても，左上が起点（0， 0）であることなど，基本的な部分は子ども達が自分で気付くことは難しかった。小学校 6 年生の子が角度と長さという極座標で考えたのは，三角形の作図で，「分度器で角度を決める。→定規で長さを決める。→描く。」という作図の経験があったからと予想できる。

そして，中学生が理解できた理由としては，比例反比例の学習で座標についての概念を習得しているという小学生との違いが大きい。

これらの事例は，学ぶ子それぞれがもつ生活経験や既習内容に応じた，プログラミングの学習の学習内を適切に定めていくことが大切だという事を示していると言える。

4. プログラミングの学習内容を定める

4.1 学習の方法の分類

プログラミングの概念の獲得のための学習の方法は，その概念の特性に応じて，大きく分けて以下の 3 つのパターンに分類することができる。

	概念の特性	指導側	→	子どもの変容
A	教わるべき内容	教える	→	習熟する
B	気付くべき内容	考えさせる	→	気付く
C	浸らせるべき内容	練習させる	→	定着する

表 1 プログラミングの学習における学習内容の分類

教えなければわからない概念なのか，気付かせることができる概念なのか，浸らせるべき学習内容なのかを判断することで，子ども達に適切な課題提示と充実した活動を促すことができると考える。例えば，前章で示した座標については，自分で気付くことが難しい概念であるので，A 教わるべき内容となる。また，座標ことが習熟されている状況であれば，LINE については，B 気付くべき内容となるかもしれない。この辺りの系統性については今後さらに検討していきたい。

4.2 コンピューターを使わない活動

プログラミングの学習でプログラマーを養成したいわけではないのは，どの指導する側の立場の人も同じであろう。中学校学習指導要領[1]では，以下の内容についての指導を明記している。

ア コンピューターを利用した計測・制御の基本的な仕組みを知ること。
イ 情報処理の手順を考え，簡単なプログラムが作成できること。

イ に関してはコンピューターを使ったプログラミングの学習となると考えられるが，ア に関しては，プログラミングをするだけでなく，コンピューターを使用しない活動で学び進める指導内容が予想される。様々なワークショップの形態で，子どもたちの多角的な視点，探求的な視点をもたせることが必要になってくる。

小学生にプログラミングの学習をする場合には，アの内容が非常に重要になってくる。アの活動を充実させることで，知識の身体化にも繋がり，小学生も楽しんで学び進めることができると考える。

図９　絵を描く活動で逐次処理を身体化する様子*

4.3　学習時間と学習内容について

学習をする時間は限られており，公立小学校の授業で行おうとすると 45 分しか時間は与えられない。また，コンピューターを使用するという設備上の問題から，起動をさせるのであれば５分程度は削られてしまう。限られた時間の中であっても，獲得した知識や概念を子ども達自身が自分の手で確認する時間は確保していきたい。自力で入力したり，エラーを直したりするような時間が大切である。わかったのに，試すことができないという事は，体育で説明ばかりで十分な運動の時間が確保されず汗をかかなかったのと同じである。

プログラミングのコードや書式などは，仕様書で定められており，指導者側は仕様書や参考図書などを参考に授業を組むことが多く，学習で使用するコードを多く与えがちである。限られた時間で十分に自分の時間を確保するためにも，学習内容の精選と，学習の流れを綿密に計画しなければならない。

5.　今後の研究について

5.1　CUI ベースの言語環境の再認識

３章で述べた通り，CUI ベースの言語環境は，英語入力についての手立てを講じることで，十分にプログラミングの学習で活用することができる。また，GUI→CUI ギャップも発生しないので，将来的に子ども達が自分で発展的な内容に興味をもち，学び進めていくという場合に，CUI ベースの言語環境は実用的であると言える。「Scratch」をはじめとした GUI ベースの授業についての研究は日本全国や世界各国で行われているが，GUI→CUI のギャップについての研究はまだ少ない。GUI と CUI 相互のスモールステップを設けることで，ギャップを小さくしていく事が可能であるが，CUI ベースの研究を進めることが結果的にギャップを埋めるステップにもなると考えている。

5.2　Processing を活用した実験授業

現在，札幌市内で実験的に CUI ベースの言語環境 Processing を使ったプログラミングの実験授業を継続している。Processing は，導入が簡単であり，入力したコードがダイレクトに結果として返ってくる Java の描画部分を強化し抽出したような言語環境である。メディアアートの制作や，大学でのプログラミングの授業等で導入が進む言語環境であり，視覚的に「絵」として自分の活動が返ってくるので，子ども達もすぐに使いこなすことができている。現在の実験授業では，小学生の子どもにとっての適切なプログラミングの学習の内容について探っている段階である。学習内容については，４章で述べた通り「学習内容の分類」を当てはめながら，どのように子どもたちが知識を積み上げていくかを検証し，知識を積み重ねていくプロセスの系統性について研究を続けていく予定である。

図 10　Processing を使った実験授業の様子*

6.　おわりに

電化製品やコンピューターやタブレット PC などは，高性能化の代償として，ブラックボックス化してしまった。中身が見えなくなった簡単便利なモノや世の中の仕組みは，利用する側の思考と判断を単純化し，行動するまでのプロセスを簡略化すると同時に，提供する側には，サービス提供のわかりやすさ，操作のしやすさをアピールできたり，都合の悪い部分があれば見せないように隠すことができたりするなど，相互にとって都合がよい部分があると言える。

しかし，ブラックボックス化は，「どういう仕組みなのだろう？」という思考力を育むことには繋がらない。また，提供する側の説明責任をあいまいにすることができるという側面があると言える。世の中で言われている科学離れや，２０代のエンジニアが不足している問題などは世の中のブラックボックス化が影響していると言わざるを得ない。

「プログラミングの学習」が注目を集める背景には，ブラックボックス化されたモノや社会に対する，「多角的，探求的な視点の育成に繋がる学びとなりえる」という期待が込められているのではなかろうか。今後は，それぞれの活動の場で実験的に行われている授業実践を通して，プログラミングの学習に秘められている可能性を探り，その意味や価値を協力していきながら再構築していくことが求められるのではなかろうか。

参考文献 等

(1)　文部科学省 中学校学習指導要領 技術・家庭 (2008).
(2)　National curriculum in England:computing programmes of study (2013)

*　北海道大学教育学院で研究継続中 協力：スペースタイム