Squeak を利用したプログラミング教育環境の構築と実践

松澤　芳昭,　杉浦　学,　大岩　元
慶應義塾大学　政策・メディア研究科
macchan@crew.sfc.keio.ac.jp

概要

一般情報教育におけるプログラミング教育に求められるのは，論理的思考能力，問題解決能力，問題の分析・抽象化・モデル化能力の育成である．しかし，実際のプログラミング教育現場では，1)文法エラーに悩まされる，2)枝葉末節の知識を覚えなければならない，ことから，学習者が「創造的作業」に集中できないため，教育効果が上がらない．これを改善できる環境として近年注目を集めているのが，Alan Kay の提唱する Squeak[1]である．しかし，たとえ Squeak を利用しても，従来型の教育方法では，結局文法教育が操作教育に摩り替わっただけの結果となってしまう可能性が高い．Squeak の環境を活かしたプログラミング教育カリキュラムが必要である．本稿では，カリキュラムの提案と実践の結果について述べる．

1. はじめに

初等，高等教育を問わず，一般情報教育におけるプログラミング教育に求められるのは，論理的思考能力，問題解決能力，問題の分析・抽象化・モデル化能力の育成である．これを目標に，コンピュータサイエンスを専門としない生徒も対象として，さまざまな教育現場でプログラミング入門教育が行われている．これが目指すものは，人々がコンピュータを自らの思考を助ける道具として真に使いこなすことができる社会である．

しかし，実際のプログラミング教育現場において指導者を悩ませている問題は，どうしてもプログラミング言語の文法が中心になってしまうことである．プログラミング教育では，創造活動の表現の手段としてプログラミング言語を用いる．しかし，プログラミング入門者にとって，プログラミング言語の文法を習得することは最初の大きな壁となる．学習者は大量に表示される文法エラーを前にして，挫折してしまうことが多い．そのような環境では，学習者は文法エラーを正しく直すことが目的となってしまうので，本来の目的が伝わらない．

もう一つの問題は，実用言語をはじめとするプログラミング言語を入門教育の言語として用いた場合，実用性に由来する枝葉末節の知識を教えることが中心となってしまうことである．結果として，受講者はプログラミングの知識を覚えることが目的となってしまうので，これも本来の目的が伝わらない．プログラミング教育では，掛け算の筆算の方法を教えて，正しく行うことができるようにするような，人間をコンピュータ化するような教育を目的としていない．

プログラミング教育では，物事を深く理解・分析して，プログラムとして組み立てる「創造的作業」が中心になることが理想である．しかし，現在のプログラミング教育では，与えられた知識を自分で組み立てて，応用できた者だけしかプログラミング能力を獲得することができない．それゆえ，プログラミングの入門教育においては，文法などの知識項目は最小限にして，学習者が「創造的作業」に集中できる環境で行うのが望ましく，これが実現されればプログラミング教育の効果が劇的に改善する可能性がある．

これを実現できる環境として近年注目を集めているのが，Alan Kay の提唱する Squeak[1]である．Squeak に搭載された Squeak-Toys と呼ばれるビジュアルプログラミング環境(以下 Squeak プログラミングと表記した場合，Squeak-Toys を使ったプログラミングのことを指す)を用いることにより，文法エラーを気にする必要がなくなる(Squeak では，文法エラーになるような部品構成は，最初からできないようになっているため)．さらには，Squeak では提示されたプログラミング要素から必要な要素を選ぶことができるため，覚えなけれ

ばならないことを最小限にすることができる．日本でも Squeak を用いた教育が行われ始めており[2]，成果を挙げている．

しかし，たとえ Squeak を利用しても，その基本操作だけを教えて，応用は学習者次第という従来型の教育方法では，結局文法教育が操作教育に摩り替わっただけの結果となってしまう可能性が高い．これからのプログラミング教育では，Squeak のような環境を活かし，「創造的作業」のプロセスを教育するカリキュラムが必要である．本稿では，カリキュラムの提案と実践の結果について述べる．

2. Squeak を利用したプログラミング教育カリキュラム

2.1 カリキュラムの基本的考え方

我々は，実現手段としてのプログラミング言語の知識を除いた場合，プログラミング教育の中心は，

1) 自分の作りたいものを考えること(Imaging),
2) それを言葉や図を利用して表現，記述すること(Describing),
3) プログラムの部品を組み合わせて，作品を実現すること(Developing),
4) 第三者からの評価をうけて作品を改善し，第三者の評価に耐えられる作品にしたてること(Evaluating)

になると考えている(図 1).

この考え方に基づくと，プログラミング教育が，これまで行われてきたプログラミング言語の知識(図では Architecture と表現されている)中心ではなく，それをどのように活用するかを議論することが中心になる．言い換えれば，どのように(How)ではなく，何を(What)が中心テーマとなるということである．

そしてこのプロセスにおいての特に重要な点は，自分のアイディアを出す段階(Imaging)から，それを実現する段階(Developing)の間に，自分のアイディアを記述する段階(Describing)を設けていることである．我々は，この段階で自分のアイディアとその目的，さらにそれをブレークダウンした項目について，自然言語（日本語）で記述させ，それをコンピュータではなく，まず第 3 者に伝わるかどうか試させる．

この段階の教育が手薄であると，学習者は最終作品の作成プロジェクトにおいて，学習したプログラミング言語の知識を利用して，自分のアイディアを直接作品にしようとしてしまう．しかし，そのような学習者は，アイディアを自分の中で整理できていないため，始めにイメージしたものをうまく作ることができない場合が多い．

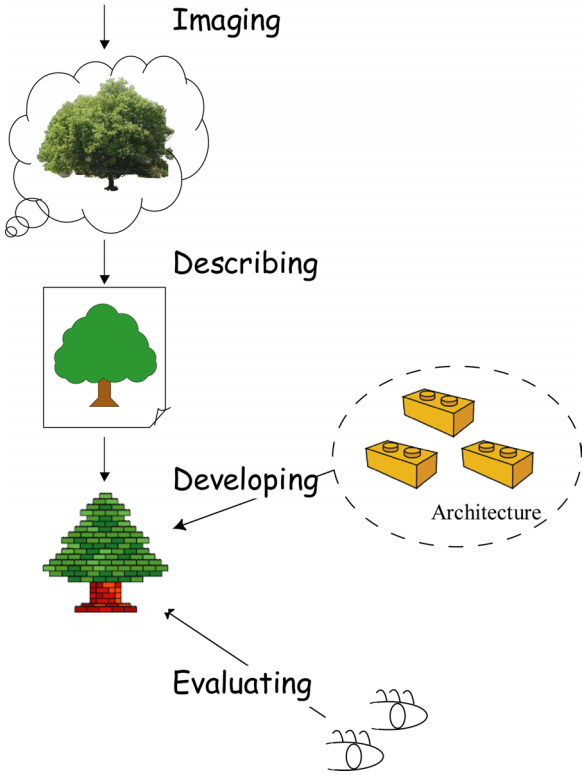

図 1　提案カリキュラムにおけるプログラミング教育の全体像

従来のプログラミング教育においても，同じように擬似言語を用いてプログラムを記述させる手法が用いられている．しかし，従来のプログラミング教育では，プログラミング言語の知識に時間をかけて教育しなければならなかったため，この領域をじっくり議論できなかった．さらには，擬似言語による表現と，プログラミング言語による表現の乖離が大きすぎるため，学習者がこの段階の学習をするための動機付けが弱かったと我々は考える．

しかし，Squeak を利用するとその文法の容易さと，オブジェクト指向の概念による自然言語との親和性の高さ，さらには日本語で構成できることも加わって，これらの問題は解決できると考えられる．

2.2 タスク分解を教育する

我々の提案するプログラミング教育カリキュラムの最終目的は，カリキュラムの終了時に学習者自らが設定したプロジェクトを実現することである．学習者が設定する課題はゲームやシミュレーションなどが主となるが，それらのプログラムはある程度の規模を持ったものになる．

ある程度の規模を持ったプログラムの場合，うまくタスク分割を行い，分割された部品を一つ一つ作り，うまく動作するかテストするという作業が必要である．しかし，初学者はたいてい，すべての部品を一気に作りあげようとする．そして悲惨な結果となる．

ある程度の規模を持った最終プロジェクトを伴うプログラミング教育では，例えそれが入門教育であっても，問題をタスク分割し，計画的に組み合わせることを教育するべきである．Squeak を利用することで，プログラムの知識に関する教育を行うコストを減少させ，このような能力の育成に注目した教育を行うことができる．

3. カリキュラムの実践

2 章で述べたカリキュラムの考え方を基に，カリキュラムを構築した．これらのカリキュラムはいくつかのプロジェクトから構成されている．本章では，これらのプロジェクトについて，高等学校の生徒と大学の文系学生に対して実施した結果を交えながら述べる．

3.1 流れ星プロジェクト

このプロジェクトは，オブジェクトの座標を利用して動きの制御を行うことをテーマにしたカリキュラムである．このプロジェクトの目的は，たくさんの流れ星を画面の端から端まで移動させるプログラムを完成させることである．

このプロジェクトを行う前に，完成したプロジェクトを学習者に提示し，実現のための手段を考えさせることから授業を開始する．Squeak には「～を進める」というタイルが用意されているが，このタイルを使用した場合，画面の端にオブジェクトが接触すると，つかえてそれ以上進まないという問題がある．これを解決するための方法を考えることから学習者の試行錯誤が始まる．

星を画面の外まで進めるためには，オブジェクトの座標値を増減させることで動きを制御する必要がある．教師側が与えるヒントは「星の座標をうまく利用すると良い」ということだけである．このヒントだけでも，学習者の多くは，星を任意の方向に進めるアニメーションを試行錯誤して作成することができる．

星を画面の端から端へワープさせるためには，画面の端に出たときに，星の座標を反対の画面外の座標を設定する必要がある．Squeak において，条件分岐を実現するためには，「テスト」のタイルを使用する．このタイルの使い方は単純で明快なので，細かく説明することなしに Squeak 上で条件分岐を実現することができる．

このワープのための実現方法を考える時間を学習者に与え，考えついた方法を日本語で説明させる．その後に，必要になるテストのタイルの使い方を説明し，実装を行わせることで，プロジェクトが完成するまでのプロセスを分解して順番を考えることを体験し，目的をコンピュータで実現するまでのプロセスを教育することができる．

3.1 自動運転自動車プロジェクト

このプロジェクトは，Squeak コミュニティでよく行われるプロジェクトである．このプロジェクトのテーマは，センサーによるフィードバックを利用して自らの動きを制御するプログラムを作ることである．

我々の教育手法では，まず車が道を検知するアルゴリズムを日本語で説明させる．具体的には，ボール紙に大きく印刷した車の模型と，黒板を利用し，自分の考えるアルゴリズムを他の学習者に説明することを行った．

この演習において，学習者は様々な実現方法を考案した．例えば，道を識別する方法として，色を識別する役目を果たすセンサーを利用する方法が考えられるが，車が一方方向にしか進まない場合，センサーを両端につけなくとも自動車の動きを制御してプロジェクトを成功させることができる．

こうした演習を行うことにより，自分でプログラムのアルゴリズムを考えることができる能力を育てることができる．さらには，生徒に実現手段に関する正解はないということを体験させることができる．

3.3 並び替えプロジェクト

このプロジェクトでは，ゲームのハイスコアの並び替えを題材にして，本格的なアルゴリズムについて議論する．さらには，入れ物に入った任意

のオブジェクトという抽象的なプログラムに挑戦する．

このプロジェクトでも，まずは日本語で，どのようにしたら並び替えることができるか，意見を出し合う．「順番に」や「右側に寄せていく」といった，人間にしか伝わらない言葉で説明する生徒と，「隣の数値と」といったコンピュータに伝わる言葉で説明する生徒が意見交換する場面が観察された．

その後，第 3 者が記述した自然言語を基に，自分でその作業を行わせると，人間にしか伝わらない言葉での表現は実は解釈によって異なる結果が出てしまうことが分かる．さらには，そのような記述では当然ながら Squeak でのプログラムには変換できないことが分かる．

このプロジェクトによって，自然言語による記述においても，人間に伝わる表現と，コンピュータに伝わる言葉の区別ができるようになり，このプロジェクトの以後，学習者は，自然言語の記述方法について吟味するようになった．

3.4 ブロック崩しプロジェクト

このプロジェクトのテーマは，ゲームの作成プロセスを一通り体験することである．

このプロジェクトでは，計画の段階において，ブロック崩しではどのようなゲームか，そしてそれをどのように実現するか，日本語で記述させる．そして，挙げられた要素どのような順序で作っていけばよいか，そのタスク分解について議論する．

そうして分解された要素を一つずつ，アルゴリズムを受講者に考えさせながら作り上げていく．そのような実習により，分解を行い，一つずつ部品を作っていくことで，規模が大きなプログラムにも対応できることを体験させることができた．．

4. 考察

4.1 生徒の質問の質の変化に関して

従来のプログラミング教育では，学習者からの質問の大半は文法エラーの対処法に関するものであった．Squeak を使った教育においては，文法エラーの対処法に関する質問は激減し，その代わりにどのようにタイルを組み合わせるべきかについての質問が増加した．

このことは，Squeak を利用したプログラミング教育では，プログラム言語の知識を用いて，いかに自分の作りたいものを実現するかについての議論が可能になったことを示している．

4.2 利用できるタイルの粒度に関して

学習者は，プログラムを実現するときに，自然言語で記述されたアイディアを頼りに，Squeak で学習者自身がタイルを組み合わせ，アイディアの実現方法を試行錯誤する．その際に，提供されているタイルで実現できる機能の粒度が重要であることが考察された．

現在の Squeak では，機能の粒度が一定ではなく，一つのタイルで複雑な目的を実現するものまで用意されてしまっている．特に初学者にとっては，小さな目的を実現するための手段を考える経験を数多く行う経験が重要であるため，粒度の荒い（機能が複雑な）タイルを使用できてしまう環境はかえって教育を阻害することになる．

例えば，「オブジェクトを折り返す」といったタイルは，その実現手段が学習者に実装できる場合には便利であるが，初学者にとっては「折り返す」という手段の実現方法を考えてプログラムとして実現するという経験を奪ってしまう原因になる．こうした問題を解決するためには，タイルの複雑さを学習者のレベルによって変更できるような環境が理想的である．

5. おわりに

本稿では，Squeak を利用したプログラミング教育のカリキュラムについて，カリキュラム実践の結果とともに述べた．

結果，従来の環境を利用した場合に比較して，プログラミング教育の理想に近い教育がしやすくなるという実感が得られた．しかし，この実践は，我々にとって始めての試みであり，実際には，Squeak を利用しても，サンプルプログラムの丸写ししかできないような学習者は存在する．今後，2 章で述べたような理想に近づくべくカリキュラムを改良していく予定である．

6. 参考文献

[1] Squeak http://www.squeak.org/

[2] 軽野宏樹，木實新一，上林弥彦．ALAN-K プロジェクト：Squeak を活用した創造的な情報教育の試み．情報処理学会研究会報告(CE-69-1), pp.1-8, 2003