サイエンス&プログラミング教室の試みと実践

柳田 拓人 [1)], 中村 景子 [1)], 林 晋平 [2)]

1) 株式会社スペースタイム
2) 東京工業大学 情報理工学院

yanagida@stxst.com

Trial and Practice of the Science and Programming School

Takuto Yanagida [1)], Keiko Nakamura [1)], Shinpei Hayashi [2)]

1) Space-Time Inc.
2) School of Computing, Tokyo Institute of Technology

概要

2014 年秋から、株式会社スペースタイムではサイエンス教室とプログラミング教室とを合わせた小中学生向け教室を、「ラッコラ」と名付けて運営している。ラッコラの特徴は、同一のテーマをサイエンス編とプログラミング編とで相互補完的に扱う点、プログラミング編において、プログラミング言語として JavaScript を採用し、独自の開発環境を使用する点にある。本稿では、ラッコラの概略とこれまでの歩みについて説明する。

1 はじめに

近年、小中学生の習い事として、サイエンス（理科）教室やプログラミング教室が注目を集めている。ここでのサイエンス教室は、ある科学に関するテーマに沿って実験やアクティビティを行うことで、そのテーマやより科学一般について、知識や技能、考え方を学ぶ場を指す。一方、プログラミング教室は、文字通りプログラミングの概念やプログラミング言語を学び、何らかのプログラムを成果物として制作する場を指す。

株式会社スペースタイムでは、2014 年の秋から、小学生・中学生を対象としたサイエンス&プログラミング教室「ラッコラ（Laccolla）」を運営している[1]（図 1）。ラッコラはサイエンス&プログラミング教室と銘打っているとおり、サイエンス教室とプログラミング教室を組み合わせたものであり、次の 2 点の特徴を持つ。

図 1. ラッコラの様子

- 各回の前半をサイエンス編、後半をプログラミング編と分け、同じテーマをそれぞれで関連付けて扱うカリキュラム。
- 使用するプログラミング言語として JavaScript を採用した上で、初心者向けの開発環境（エディター）を独自に開発。

以下では、著者らが考えるプログラミング教室の課題をまとめたあと、ラッコラのカリキュラムと構成等について述べる。また、これまで教室を運営してきた結果を踏まえた考察を行う。

2 プログラミング教育の課題

プログラミング教室では、何のためにプログラミングを習得し、どんなプログラムを制作するのかという動機付けが重要となる。

ここで、プログラミングの学習、そしてプログラミングそれ自体が持つ役割として、次の二つを挙げる。一つ目は、表現するための「メディア」としての役割、二つ目は考える能力という「リテラシー」としての役割である。

プログラミングは、自分自身が知識や概念をどのように理解しているかを「表現」するためのツールとして用いることができる。同時に、プログラミングによって、プログラミングをする上で必

要な「手順＝アルゴリズムを考える力」を向上させることもできる。そこで、動機付けにおいても、二つの役割を反映させるべきであると考える。

また、どのようなプログラミング言語を用いるのかも、検討課題となる。

著者らは、プログラミング教育における学習の連続性と、実際との連関を意識させることを重視している。本来、プログラミングの学習に終わりはなく、学習者にその意思があるならば、発展させられるべきであろう。また、プログラミングは実学であり、その学習の先には実用的なソフトウェア開発が見えているはずである。それは学習者のモチベーションに繋がる。なお、ここでは、単に職業としてのプログラマーを増やすことを意図しているのではない点に留意されたい。

従って、使用するプログラミング言語の選定にあたり、以下の要件を満たす必要がある。

- 実際のソフトウェア制作のために行われるプログラミングで用いられていること。
- 現在広く使われているプログラミング言語が共通に持つ言語機能を備えていること。

プログラミング教育の現場ではビジュアル・プログラミング言語が広く用いられている。ビジュアル・プログラミング言語ではブロックのような部品を画面上に並べることによってプログラミングを行う。典型的なものに Scratch [2]がある。これらの言語を用いることにより、ソース・コード（プログラミング言語で書かれた、プログラムを表すテキスト）の記述を避けたプログラミング教育が実現できるため、学習面でのメリットが大きい。しかし、これらの言語の多くは教育に特化されており、現実のソフトウェア開発では利用されておらず、著者らの要件を満たさなかった。

2 ラッコラの概要

ラッコラでは、サイエンスを学び、それを表現する手段としてプログラミングに取り組む。すなわち、サイエンス編とプログラミング編とで同一の対象を扱い、その学びを交互に行い深化させる。これによって、応用対象への適用（メディア）としてのプログラミングと、手法（リテラシー）としてのプログラミングを、同時に動機付けることが可能となる。さらに、サイエンスを単に知識として身につけるのではなく、サイエンスの本質であり、かつ、深い理解に必要となる「考える力」の習得に繋げることができる。

また、プログラミング言語として JavaScript を採用した。これは、インターネット上を含めた様々な分野で使用されている、実用的なテキスト・ベースのプログラミング言語である。このような実用言語では、あるプログラムを別のプログラムから「ライブラリ（プログラムの部品の集まり）」として再利用することが可能である。それによって、複数のプログラムを組み合わせて複雑な結果を得ることができる。

ラッコラでは、参加者は自らのプログラムを組みわせることによって、全体として複雑なプログラムを制作できる。また、カリキュラム内で使用する各種ライブラリも同じ言語で開発されているため、参加者はすべて、自らのプログラムと同様に、中を「覗いて見る」ことが可能である。さらに、参加者が使用するアプリケーション（2.4 節で説明）も、JavaScript で開発されている。これは、参加者にとって、自分たちが学ぶ言語が実用的なものであるという証拠になる。

2.1 カリキュラム構成

ラッコラにはカリキュラムとして、初級 3 コース、中級 2 コースがあり、各コースにおいてそれぞれ科学に関するテーマが一つ扱われる。1 コースは全 5 回で構成され、基本的に各回 150 分である。第 1 回から第 4 回までは前半 60 分がサイエンス編、後半 90 分がプログラミング編となる。第 5 回は後半が参加者の保護者を招いての発表会となるため、前半を発表のための準備時間としている。毎回 16 ページ程のワークシート（図 2）が配布され、参加者はそれに書き込みながら学習を進める。

なお、ラッコラの対象者は小学 3 年生から中学生までとしているが、サイエンス編の内容は大学学部レベルまでとし、小中学校の教育指導要領等に合わせることを意図していない。

2.2 各コースのテーマ

初級コースは、「星びとコース」、「葉っぱコース」、「雪そらコース」から、中級コースは「生き物戦略コース」、「感じる生命コース」からなる。各コースについて、以下にテーマの概略をまとめる。

初級 星びとコース（地学・天文学）

普段とは異なる視点で宇宙をイメージし、星の自転と公転が生み出す現象に迫る。プログラミングで天体の見え方や動きを表現する。

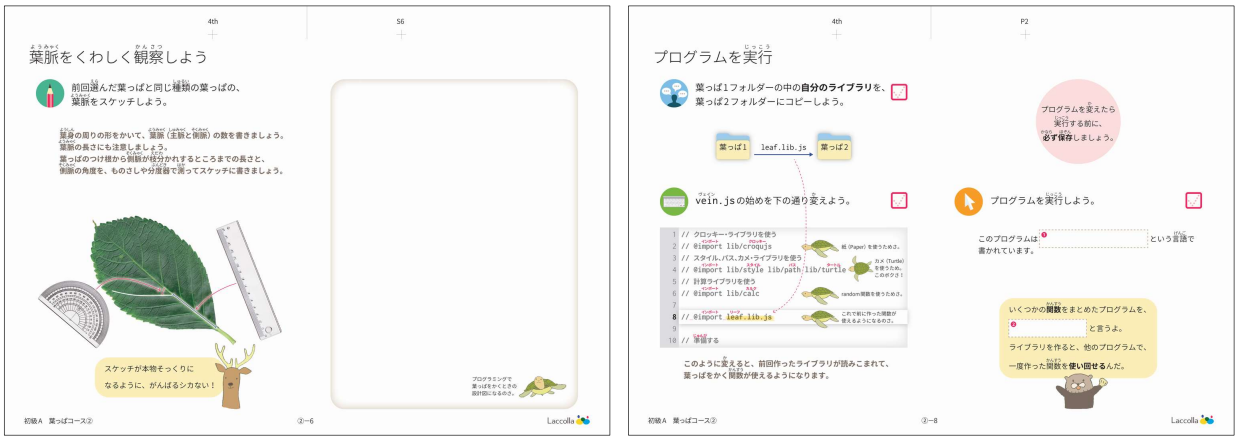

図 2. ワークシートの例（左: サイエンス編、右: プログラミング編）

初級 葉っぱコース（植物学）

木の葉に注目し、形や色、葉の付き方の特徴を見つけ、紅葉の仕組みを実験で探る。プログラミングでその特徴や仕組みを表現する。

初級 雪そらコース（気象学）

雪の形から、雪ができる過程、雪ができる場所などを、実験などを通して探る。その自然現象をプログラミングによって表現する。

中級 生き物戦略コース

生き物の生存のために共通する方法「捕食、成長、繁殖、防衛」を探り、プログラミングで作った生き物でそれらを確かめ表現する。

中級 感じる生命コース

生き物が環境からの刺激に反応する仕組みを、様々なレベルを通して捉え、プログラミングで自分独自の微生物として表現する。

初級コースでは、参加者は毎回異なるプログラムを初めに受け取り、サイエンス編で学んだことを表現するために、それを変更していく。この際、前回までに制作したプログラムを再利用することも要求される。全 5 回を通して自ら書いたプログラムを徐々に組み合わせていくことで、プログラミング未経験の参加者であっても、全体として複雑なプログラムを制作することになる。

一方、中級コースでは、参加者は全 5 回を通じて一つのプログラムを改良していく。本稿では詳細を省くが、中級 2 コースではいずれも参加者は仮想的な「生き物」を制作する。これをコンピューター上の仮想的な「自然環境」で動作させ、その様子を観察する。複数の参加者によって作られた生き物を同一の環境で動作させることによって、環境と生き物、あるいは生き物同士の相互作用を観察できる。置かれた環境でよりよく繁栄する、すなわち絶滅しない生き物となるように、参加者は自分のプログラムを改良する。

2.3 コースの内容（例: 葉っぱコース）

ここで初級葉っぱコースを例に、各回の内容を説明する（表 1）。プログラミング編で制作するプログラムの動作結果の例は図 3 にまとめた。

第 1 回 サイエンス編では、参加者は様々な木の葉を観察し、色や形、においなど、様々な特徴を見つけ、それに従って自分独自の分類を行う。さらに、図鑑等の分類の仕組みを学ぶ。

プログラミング編では、選んだ葉の形状に注目し、タートル・グラフィックス※1 によって画面上に葉（の形状）を描くプログラムを書く。プログラムの各行の順序に意味があることに気付かせ、逐次処理の概念を習得する。

第 2 回 サイエンス編では、葉を構成する葉脈や葉肉などの組織に注目して、その役割について学ぶとともに、葉脈標本を作成する実験を行う。

プログラミング編では、第 1 回で作ったプログラムを再利用し、葉脈の付いた葉を描く。繰り返し処理を学び、効率よく描く方法を習得する。

第 3 回 サイエンス編では紅葉した葉、していない葉から薄層クロマトグラフィーによって色素を分離する実験を行い、色素の変化を観察する。また、気温と紅葉の関係について学ぶ。

プログラミング編では、前回までのプログラムを再利用して、画面上の設定温度に合わせて紅葉する葉を制作する。温度に応じた色の変化を表現

※1 ペンを持った仮想的な「カメ」に、「前進する」、「右に曲がる」といった指示を行うことで絵を書く仕組み。座標系を意識せずに絵を描けるという利点がある。なお筆者らは、JavaScript でタートル・グラフィックスを実現するためのライブラリを開発した。

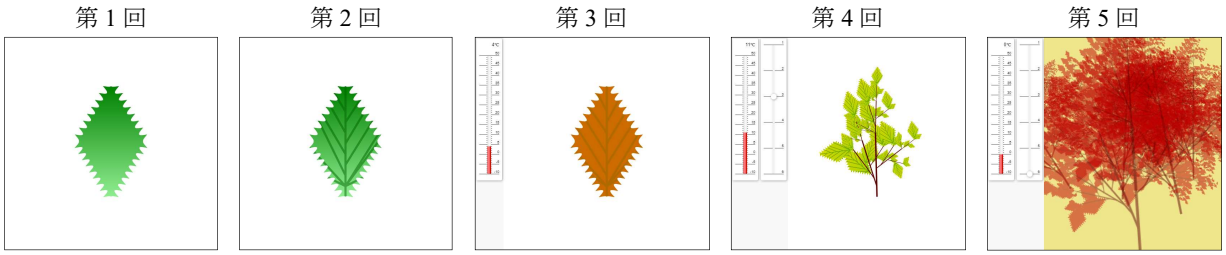

図 3. 葉っぱコースで制作するプログラムの動作結果例

するために、条件分岐の概念について習得する。

第 4 回　サイエンス編では木が形成される際に見られる二種類の成長のルールを、幹や枝に見立てた碁石を並べることで再現してみる。

プログラミング編では、再帰呼び出しを使って木の枝を描くプログラムを作り、枝の先にこれまで作ってきた葉を描く。これによって木を描く。

第 5 回　発表会として、それまでにサイエンス編で習ったことを参加者自らが出題するクイズ大会を行う他、すべてのプログラムを組み合わせて森を描くプログラムを作成し、発表する。

2.4 プログラミング環境の開発

プログラミングにはプログラムを入力・編集するためのアプリケーション、開発環境（エディター）が必要となる。また、JavaScript によって書かれたプログラムは、通常ウェブ・ブラウザ上にて実行され、その際には、HTML ファイルや CSS ファイルが合わせて必要となる。

既存のテキスト・エディターや統合開発環境の利用を検討したが、以下の問題点があった。

表 1. 葉っぱコースの構成（S はサイエンス編のテーマ、P はプログラミング編のテーマ）

第 1 回　葉っぱの分類学（仲間分け） S: 分類学を体験しよう。 P: 順番になぞって葉っぱの形をかこう。
第 2 回　葉っぱの形態学（形と役割） S: 葉っぱの特徴と役割を覚えよう。 P: 線をくり返して葉っぱに葉脈をかこう。
第 3 回　葉っぱの生理学（色の仕組み） S: 葉っぱが緑色の理由、紅葉する仕組みを知ろう。 P: 場合に合わせて葉っぱの色を変えよう。
第 4 回　木の発生学（形とルール） S: 木の形が作られるルールを考えよう。 P: 自分で自分を呼び出して木をかこう。
第 5 回　森の品評会（発表会） S: サイエンス・クイズ大会をしよう。 P: 今までのプログラムをまとめて森を作ろう。

- 学習時には不必要な機能を含むため、利用者が戸惑う恐れがあった。
- HTML ファイルなど別に用意する必要があり、プログラムを入力するだけですぐに実行してみるということが難しかった。
- エディタ画面の表示を変更して、参加者が使用するワークシートの紙面と、見た目を一致させることが難しかった。
- 編集時のエラー・チェック表示やプログラム実行時に発生するエラー・メッセージが英語で表示されることがあった。

そこで著者らは、参加者のプログラミングを支援する開発環境 Croqujs（クロッキー）を開発した（図 4）。これは、Windows と Mac で動作し、プログラムを入力・編集し、ボタンを押すだけで、内部的に生成された HTML を通じて、別のウィンドウに組み込まれたウェブ・ブラウザでプログラムを実行できる。また、Croqujs はプログラムの編集・実行機能の他に、プログラムを「ライブラリ」として保存する機能と編集中のプログラムにライブラリを導入する機能を提供する。

なお、Croqujs の開発にあたっては、Electron [3] というウェブ関連技術を用いてデスクトップ・アプリケーションを開発するためのフレームワークを採用した。つまり、Croqujs 自身が JavaScript（および HTML、CSS）によって実装されている。

3 考察

サイエンスとプログラミングを組み合わせたことによる効果を定量的に求めるのは困難である。

著者らはラッコラの実施前に、試験的にサイエンスの要素を取り入れず、代わりに「絵を描く」こと自体をテーマとしたコースを実施してみた。しかし、参加者の興味や関心は、プログラミングという行為自体にとどまっていたように思われた。

一方ラッコラでは、サイエンスの方が好き、あるいはプログラミングの方が好きと、参加者の興

図 4. Croqujs の画面。左のエディター・ウィンドウには編集中のプログラムが表示され、この例では葉の描画を行うプログラムが表現されている。右上の実行ボタンをクリックすると右の実行ウィンドウにプログラムの実行結果、ここでは葉が描画される。参加者は，プログラムの編集と実行を繰り返すことにより、プログラミングを実践する。

味の対象が分かれる場合でも、カリキュラム上、相互に関連があることによって、モチベーションが持続する様子が見られた。

JavaScript を使用し、参者者が直接編集するプログラム以外も同じ言語で制作したことの結果として、中級コースにおいて、参加者が著者らによって制作されたライブラリを見たり、あるいは編集してみたりする場面が見られた。これは、想定していたとおりである。

しかしながら、テキスト・ベースの言語を使用することにはデメリットもある。

一つ目は、キーボード入力の困難さである。ラッコラは対象者を小学 3 年生以上としており、小学校で英語教育がなされるようになった現在においても対象者の多くはアルファベットの読み書きが容易ではなく、また、そもそも入力デバイスとしてのキーボードに不慣れである。

しかしながら、多くの参加者は、1 コース全 5 回を修了するまでの間に、アルファベットのキーボード入力をある程度習得する。したがって、プログラミング教育において、キーボードの使用はそれほど大きな障壁とはならなかった。

デメリットの二つ目は、プログラムの打ち間違いによるエラーの発生とその修正の手間がかかる点である。これについては、エラー・メッセージを日本語で表示するなどの対応をとっているが、参加者が自力でエラーを修正できない場面が多々見られるため、不十分であると考えている。

4 おわりに

本稿では、プログラミング教室に必要な要件をまとめた後、サイエンス&プログラミング教室「ラッコラ」の構成とカリキュラムの概要について述べた。また、独自の開発環境について説明した。

今後の課題は、Croqujs の改良によるテキスト・ベースのプログラミング言語を使うことのデメリットの解消である。具体的には、エラー情報を収集し解析することによる、典型的なエラーを自動的に修正する機能、あるいは、より初心者にとって分かりやすいエラー表示等を考えている。また、新たなカリキュラムの開発も検討している。

※ラッコラでは参加者から、広報・研究目的に限定し、写真の撮影と利用の承諾を得ている。

参考文献

[1] ラッコラ - 小学生・中学生向けサイエンス&プログラミング教室
http://laccolla.com/

[2] Scratch - Imagine, Program, Shar
https://scratch.mit.edu/

[3] Electron
https://electronjs.org/