プログラミング教育のための Web 教材における改善点とその改良

坂東 将光 *

Improvements of a Programming Teaching Material

Masamitsu BANDO *

Recently, programming education attracts much attention from educators and students. In order for students to learn programming efficiently, appropriate programming teaching materials are necessary. In this paper, I introduce some improvements to a programming teaching material (JSPad) for programming education.

keyword software, programming, teaching materials

1. 緒言

近年、プログラミング教育は教育関係者を始めとした様々な人々から注目を集めている。2020 年のプログラミング教育必修化により、今後プログラミングはより身近なものとなり、プログラミングのスキルが就職や進学において重要となることが予想される。しかしながら、現状のプログラミング教材が今後のプログラミング教育にとって十分なだけ存在するとは言い難い。

初学者向けのプログラミング教材については、MIT メディアラボのグループが提供している Scratch [1]、株式会社 AHIRU が提供している PROCK [2]、合同会社デジタルポケットが提供している Viscuit [3] や、文部科学省が開発したプログラミン [4] など、比較的充実している。Viscuit は全くプログラミングというものを知らない子供を主なターゲットにしており、PC を使って創造できるということを学ぶことができるため、とりわけ小さな子供には有効な教材であると思われる (表 1 を参照)。PROCK、Scratch、そしてプログラミンも比較的小さな子供をターゲットにしており、単純な命令に対応するブロックを組み合わせることで様々な絵的表現を実現できる。これらは Viscuit よりは難解であるが、それでも子供が十分に学べるだけの容易さとなっている。

本論文では、これらの教材の次のステップの教材として開発された JSPad における、最近の改良点と残る改善点について紹介する。

表 1: 様々な初学者向けプログラミング教材とその主な対象年齢。

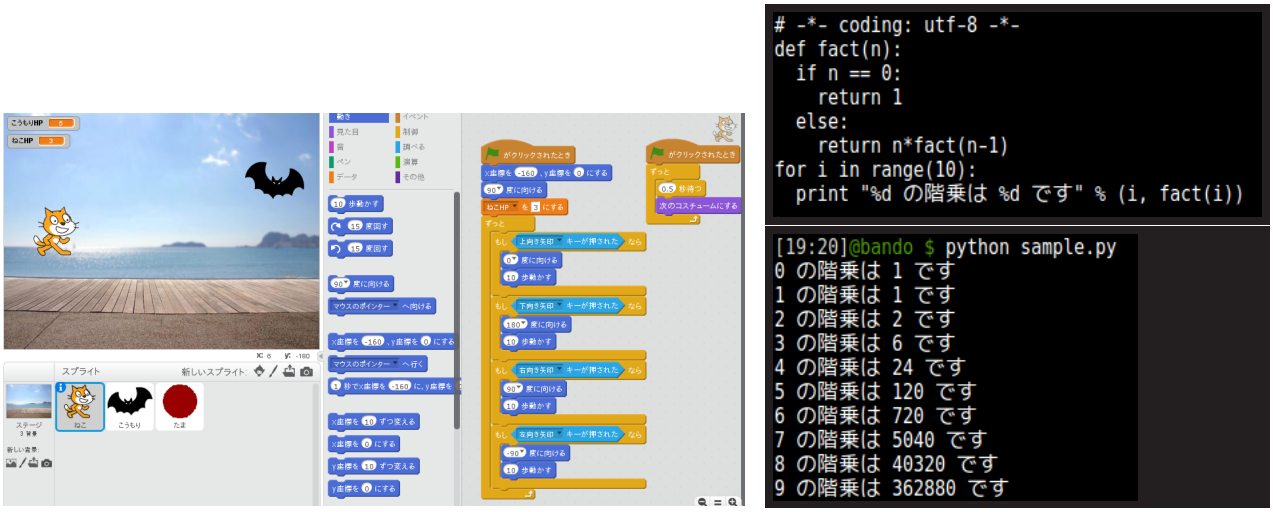
プログラミング教材	主な対象年齢
Viscuit	5 歳から小学校低学年 (推定)
プログラミン	小学生 (推定)
PROCK	小学生から高校生
Scratch	8 歳から 16 歳

2. JSPad とは

JSPad とは、インターネット環境と Web ブラウザさえあれば利用できるプログラミング学習環境である [5]。ここでは、JSPad の必要性について述べる。

前述のように、様々な導入向けのプログラミング教材があることでプログラミングに興味をもつ子供は増えるだろうが、では「“本格的に” プログラミングを学ぼう」となったとき、適切な難易度のプログラミング教材はこれまであまり存在しなかった。大抵、こういった場合には実際に開発現場で使われる言語、例えば JavaScript や Java や Python、を学ぶことになるだろうが、JavaScript は (Node.js などの特殊な場合を除いて) 基本的に HTML の理解が必要であるし、Java はオブジェクト指向を学ぶ必要がある。Processing というグラフィカルな表現がしやすく丁度良い難易度の言語もあるが、PC へのインストールが必要である [6]。Python はそこまで難解ではないものの、グラフィカルな表現は Scratch と比べると相当難解である。かといって、図 1 (a) のように Scratch などで “見える” プログ

*近畿大学工業高等専門学校
総合システム工学科　制御情報コース



(a) Scratch　　(b) Python

図 1: Scratch と Python の違い。Python は簡単ではあるが、Scratch などの次のステップとしては難解である。

ラミングをしてプログラミングに興味を持った子供達に、次のステップとして図 1 (b) のような算数の計算結果の標準出力をさせても失望しか生まないことは想像に難くない。プログラミングに興味を持ってもらいやすい環境が整いつつある現在、興味を持った子供に対して、いかにスムーズに学習という名の階段を上ってもらうかが重要である。

そこでこれまでの研究では、Web 上でプログラミング学習が行える環境である JSPad を開発した。これは JavaScript をベースにした言語の学習が Web ブラウザ上で行えるものであり、Processing のようにグラフィカルな処理が容易で、図 2 のように書いたコードの結果が同じページですぐに確認できる。Web ページなので、インターネット環境と Web ブラウザさえあれば利用することができ、スマートフォンでも利用できる。JSPad は近畿大学工業高等専門学校の 2 年次の授業である情報処理 II にて、2018 年度からプログラミングの基礎を学ぶために導入されている。

3. JSPad の改良と残る改善点

JSPad を導入しておよそ 1 年経ったが、これまでの授業において生じた主な問題は以下の 3 点であった。

(a) シンタックスハイライトが無くコードが見難い。

(b) 自動インデント機能がないので、コードを書き難い。

(c) 無限ループに陥った際に復帰する手段がない。

(d) 処理結果が纏めて出力される。

(a) および (b) については、コードを書く部分が textarea で作られていることが原因である。そこで、

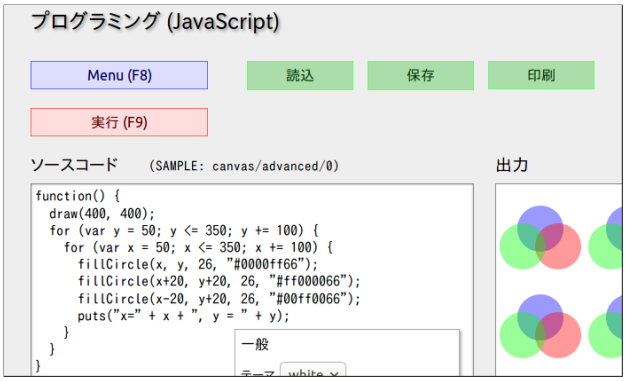


図 2: これまでの JSPad の画面。左にコードを書く部分があり、実行ボタンを押すと右側に結果が表示される。

ブラウザ上で使える JavaScript 製のテキストエディタである Ace [7] を textarea の代わりに JSPad に埋め込んだ。Ace には実に様々な機能があるが、JSPad では図 3 のようにリアルタイムシンタックスハイライト、自動インデント、行番号表示、そしてコードの折り畳み (フォールディング) 機能を有効にしている。Ace にはシンタックスチェックの機能もあるが、JSPad はオリジナルの関数も多く、通常の JavaScript とは書き方が多少異なるため無効にしている。

一方、(c) と (d) については JSPad の根本的な部分の問題である。JSPad では、ユーザが入力した文字列 (コード) を eval してその結果を出力しているが、このコードの停止性を完全に判定することはできない。特定の場合に限れば無限ループを検知できるが、これは今後の課題である。(d) については、ユーザの入力と出力を交互に繰り返すプログラムを実行した際、出力が最後にまとめて行われる現象である。Ruby などの言語

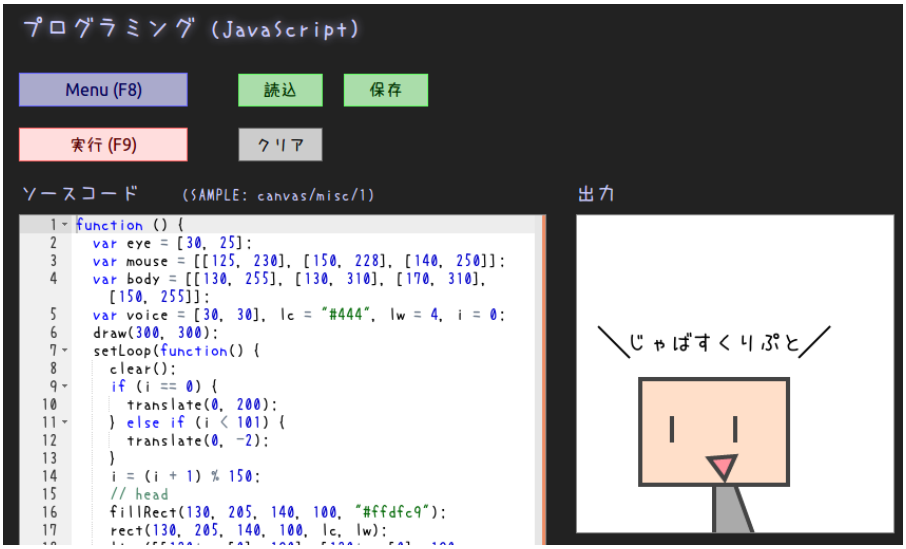


図 3: 最新の JSPad の画面。コード入力部分には行番号が付いているほか、シンタックスハイライトが有効になっていることがわかる。

でよくあるバッファリングと同じであると思われるが、現状解決方法がよくわかっていない。

4. 結言

本論文では、スマートフォンで利用できるプログラミング教材である JSPad における最近の改良点と残る改善点について紹介した。上で紹介した以外にも、ユーザビリティに関する細かな修正を多く行っており、処理が重くなったため別のアドレス [8] で公開している。JSPad は、プログラミング言語初学者に向けた、ウェブブラウザ上で JavaScript を学ぶ事ができる環境である。JSPad によって、プログラミングの面白さを実感してくれる学生が少しでも増える事を期待したい。

参考文献

[1] https://scratch.mit.edu/
[2] https://pm.ahiru.co.jp/hp/index.jsp
[3] http://www.viscuit.com/
[4] www.mext.go.jp/programin/
[5] https://masabando.github.io/JSPad/
[6] https://processing.org/
[7] https://ace.c9.io/
[8] https://masabando.github.io/JSPad/Ace/