Processing Web IDE を用いた プログラミング基礎教育の試み

三浦 元喜[1,a)]

概要：これまで C 言語を中心に行っていた大学 2 年次の学生に対するプログラミング基礎科目について，Processing.js を併用したカリキュラムを構築し，教育を試みた．Processing Web IDE を用いると，Web ブラウザのみでソースコードの編集と実行を行う環境を提供できるため，学習者は授業時間以外にプログラムを気軽に修正できる．またスマートフォン上で動作するブラウザ環境でも作成したプログラムを即座に実行できるため，学習者のプログラミングに対するモチベーションを高める効果が期待できる．課題提出システムのログを解析したところ，Processing.js を用いない場合と比べて，時間外アクセスの増加が確認された．

1. はじめに

プログラミング初学者に対する教育においては，動機づけ [1] や，自己効力感や満足度を高める [2] ことの重要性が指摘されている．これらを達成して学習意欲および学習効果を高めたり維持したりするために，ゲーム環境への没入感を利用した学習 [3]，ゲームエンジンを利用した学習 [4] や，戦略的シミュレーションゲームを題材にした講義 [5]，ゲーム性を取り入れた課題を設定する試み [6] などが行われている．このように，学習意欲を喚起するための様々な工夫の重要性が増加している．しかし一方で，現実的な環境やリソースの制約下で多数の学習者を同時に指導する必要があったり，C 言語等の言語に特化した形で従来型のプログラミング学習スキルを身に付けさせてほしいといった要請も根強い．

そこで本研究では，現実的なリソースの制約下のなかで，従来型の C 言語等のプログラミングを踏襲しつつ，ゲーム性やインタラクティブ性により動機づけや満足度，プログラミングに対する興味を高めるための方策として，Processing Web IDE を用いたプログラミング基礎教育の試みについて述べる．本研究ではおもに非情報系学科の学生に対するリテラシーとしてのプログラミング教育を対象としている．

[1] 九州工業大学 基礎科学研究系
Faculty of Basic Science, Kyushu Institute of Technology
[a)] miuramo@mns.kyutech.ac.jp

2. Processing

Processing [7], [8] は Benjamin J. Fry と Casey Reas によって開発されたインタラクティブアートやビジュアルデザインを簡単に記述するためのプログラミング言語である．アニメーションやインタラクションを簡潔な表記で記述できるため，プログラミング初学者教育 [9] やメディア造形教育 [10] をはじめ，多くの授業で用いられている．また外部のセンサを接続したフィジカルコンピューティング教育 [11] にも利用されている．

Processing.js [12] は，John Resig らによって開発された，Processing の記法で記述したスケッチと呼ばれるプログラムを Javascript に変換し，HTML5 の Canvas 要素上で動作させることができる Javascript ライブラリである．最近ではほとんどの Web ブラウザが HTML5 Canvas に対応しているため，Web ブラウザさえあれば Flash や Java のプラグインなしで，視覚的なプログラムを動作させることが可能である．

Processing.js のおかげで，従来 Processing のスケッチを記述・動作するために必要であった IDE (統合開発環境; Integrated Development Environment) を設置しなくても Web ブラウザのみでプログラミングできる環境 (いわゆる Web IDE) を簡単に構築し，利用者に提供できるようになった．Sketchpad.cc [13] や OpenProcessing [14] 等の Web IDE は，Processing のスケッチに対応しており，他者が作成したスケッチを参照したり，他者をフォローしたり，他者のスケッチを改良して編集，保存したりする機能

を備えている．

我々はこれらの Web IDE を参考に，多人数クラスへのプログラミング教育に対応した Web IDE を開発し，これまで C 言語を中心に行っていた大学 2 年次の学生に対するプログラミング基礎科目について，Processing.js を併用した講義を試みた．

3. 多人数クラスに対応した Web IDE

筆者が担当している講義は 3 クラスあり，受講者は合計で 206 名 (1 クラス 56〜77 名) である．当初これらの受講者に上記の Sketchpad.cc [13] や OpenProcessing [14] 等を利用させることも考えたが，多人数のアカウント作成や提出物の管理に支障がでることが予想された．また学生がプログラムを入力するエディタの見た目や機能，操作感については，通常使用しているエディタに近いことが望ましいと考えた．

以上の理由から，筆者が開発してきた課題提出 Web システムを改良し，独自の Web IDE を構築した．**図 1** に，構築した Web IDE の画面を示す．画面左には，スケッチを実行するための領域を配置し，その右側に，スケッチを編集するためのエディタを表示している．エディタの上部には，入力したスケッチを実行するボタン，スケッチの実行を停止するボタン，スケッチをサーバに保存・提出するボタンなどを配置している．エディタは行番号表示，カーソルがある行および選択中の行ハイライト，改行時の自動インデント，予約語や数値部分の色付けを行うハイライト機能を備えている．また，細かな記号の違いを視認しやすくするため，文字サイズの変更機能を追加している．その他の機能として，ソースコード全体のカット，コピー，ペーストを行いやすくするための全選択ボタンや，ソースコードに全角スペースなどの記号を含めてしまった場合のチェックを行う機能 (**図 2** 参照) がある．CTRL+a による全選択や，SHIFT+カーソルキー上下による行選択，CTRL+x, CTRL+c, CTRL+v によるカット，コピー，ペーストおよび CTRL+z によるアンドゥといったショートカットキーも利用可能であり，講義で利用している Linux OS 上のエディタ (kwrite, gedit 等) とほぼ同等の操作感覚で，ストレスを感じることなくプログラムを作成・修正できる．本 Web IDE は，Yutaka Kachi 氏が提供しているファイル[*1]を参考に，Javascript で記述された高機能エディタ CodeMirror[*2]を組み込み，さらにフォントサイズ変更，全選択，全角記号チェック機能等を追加することで構築した．

またスマートフォン用の実行ページへのリンクを用意した．スマートフォン用の実行ページは，スケッチを実行するキャンバス画面のみを表示する．またスマートフォン用の実行ページは，システムにログインしていなくても表示・実行できる．そのため学習者はこのページの URL をメールで送信したり，Twitter や Facebook で共有することで，自分が作成したプログラムを簡単に公開することができる．

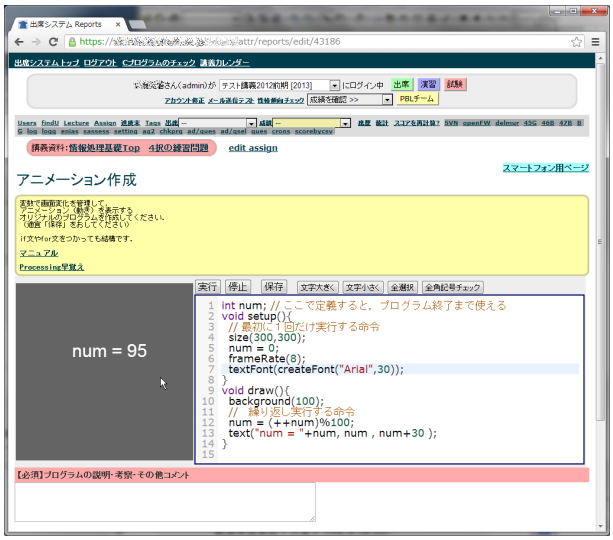

図 1 課題提出システムに組み込んで構築した Processing Web IDE

3.1 利点

Processing.js に対応した一般的な Web IDE の利点と，我々が構築した独自の Web IDE の利点に分けて述べる．

Processing.js に対応した一般的な Web IDE の利点

- Web ブラウザがあれば，どこでもプログラミングができる．
- Web ブラウザ上でプログラムを実行できる．
- プログラムがスマートフォン上でも動作するため，学生のモチベーション向上が期待できる．
- ハイライト，インデント機能付きエディタにより，編集が容易．
- Processing は C 言語の文法に似ているため，C 言語の講義と併用した場合に混乱が少ない．printf() 命令で擬似コンソール出力も使える．(scanf 命令のような標準入力には対応していないが，Javascript の prompt() 関数を利用してポップアップ画面から文字列を入力させることはできる．)
- プログラムを管理しやすい．Java 環境で動作する従来の Processing IDE はスケッチ 1 つ 1 つをフォルダに分けて管理する必要があった．
- サーバ側でコンパイル・実行しないため，サーバ側には負荷がかからない．

我々が構築した独自の Web IDE の利点

- 操作が簡潔．ログイン後に課題一覧から課題を選ぶだけで，編集画面が表示され，前回保存時のソースコードの編集や実行が可能になる．
- 教員のサンプルプログラムを初期コードとして提示で

*1 http://www.catch.jp/program/processing.js/edit/
*2 http://codemirror.net/

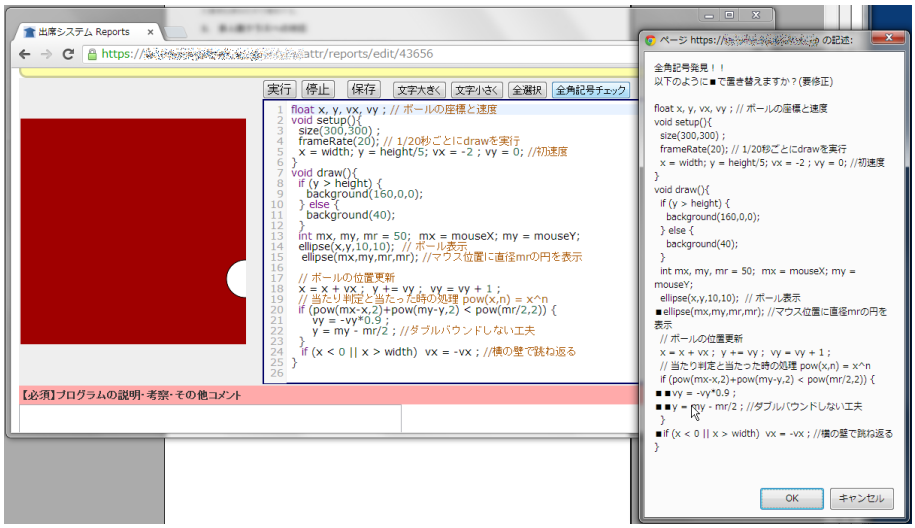

図 2　ソースコードの全角記号チェック

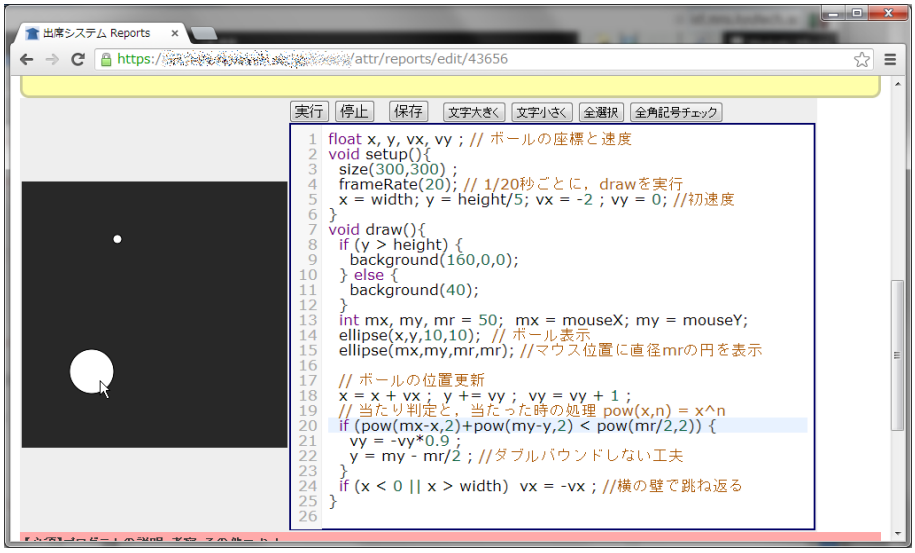

図 3　if 文の演習．ボールの跳ね返り条件を if 文で記述するサンプルを提示．

きるため，学習者は試行錯誤をくりかえしながら，プログラムを改良しやすい．

- 文字サイズを変更したり，全角記号チェック (図 2) により，プログラムが動かないときに自己解決しやすい．
- (教員側のメリット) 学生が提出したプログラムを一元管理しやすい．タグによって管理したり，フィードバックを行いやすい．
- (教員側のメリット) 1 つ 1 つのスケッチを確認する際の起動時間が短縮される．

とくに「操作が簡潔である」ことは，限られた授業時間を有効に活用するうえでは重要な要件である．従来の課題提出システムと統合したことで，C 言語の課題提出と Processing の課題提出をほぼ同等のインタフェースで実現することができ，学習者の混乱を避けることができた．

4.　カリキュラム

本章では我々が Processing Web IDE を導入して実施した講義のカリキュラム内容について述べる．

元々は C 言語を想定した講義であるため，条件文やループの理解促進を中心に Processing を導入したカリキュラムを設定した．

printf() による出力，変数の概念と四則演算，scanf による変数への動的な代入，変数の型 (整数型と小数型) の違いと演算結果に与える影響までは C 言語で説明・演習を行った (ここまでで，90 分講義 2 回分)．その後，条件文やループの説明にはいるまえに，Processing を導入した．まず，size() でキャンバスサイズを指定したうえで，rect() や ellipse() 等の関数で矩形や円等の図形を任意の位置に描くことができることを示した．つぎに fill() や stroke() 等の関数で図形の色を数値によって指定できることを教え

図 4　配列の導入サンプル．マウスカーソル位置に円を表示．左右クリックの判定例を提示．

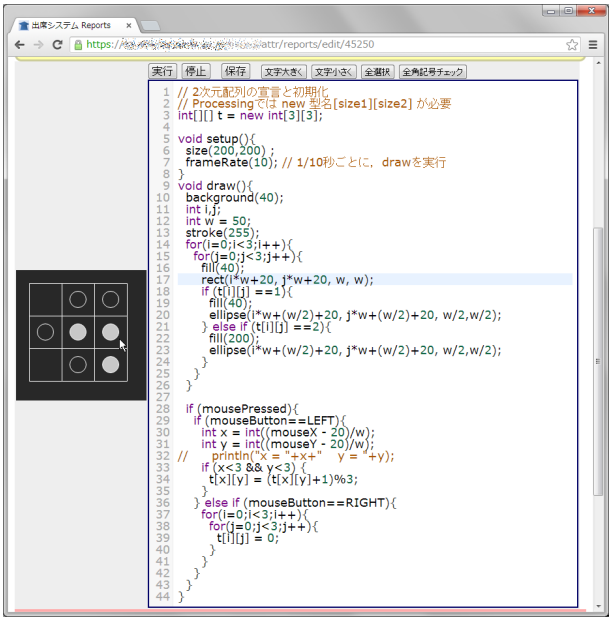

図 5　2 次元配列の導入サンプル．マス目をクリックすると●○空白の順に表示が切り換わるボード．

た．さらに数値のかわりに変数が使えることも示した．また，draw() 関数がシステムから定期的に呼ばれること，呼び出しの頻度は frameRate() 関数で指定できることを示した．剰余演算子の使い方なども例示したうえで，簡単なアニメーションを作成する課題を与えた．多くの学習者は，簡単な命令や変数の組み合わせで，動きや色の変化を出せることに驚きを感じていた．なお text() 関数による文字列の表示やフォントサイズの指定，println() による擬似コンソール出力については資料のみで詳説はしなかったが，半数程度の学習者は自分で使いかたを調べたり，他者に教えてもらったりしながら文字列を表示させていた．この時点では文字列の加工はできないが，表示位置や色を調整して工夫すれば徐々にメッセージが現れるなどの表現が可能であるため，C 言語による printf() 表示に比べると，表現の自由度が高いといえる．

第 4 週目は，条件文の例題として，ボールが跳ね返るアニメーションのプログラム (**図 3**) を示し，自由に改良させた．改良のポイントとして，跳ね返しのパッドを円形から長方形にする，パッドに当たった場所によってボールの跳ね返りの方向を変える，天井で跳ね返るようにする，等を提示した．学生はパッドを長方形表示にすることはできても，判定条件を適切に修正しないと，おかしな動作結果になることを体験によって学んでいた．なお Processing では条件式が boolean 型 (true/false) の値で返されるため，整数型で返される C 言語との違いについては若干説明する必要があった．

第 5 週目は，ループ (while) の説明を C 言語で行ったうえで，C 言語での演習と Processing の課題を課した．C 言語での演習の内容は，秀吉に仕えた曽呂利新左衛門 (そろりしんざえもん) が，1 日目に米 1 粒，2 日目に米 2 粒，3 日目に米 4 粒というように一日毎に前日の倍の米を，30 日間続けてもらったとき，全部で何粒の米を得たかを途中経過を表示しながら答えるというものであった．これまで Processing の draw() 関数でアニメーションの作成をしてきているため，繰り返しによって変数を変化させる，という考え方に対する抵抗感は減らすことができた．それでも，ループから抜ける条件やループ内の処理の順番によって求める結果にならないことがあるため，一部の学習者はループの考え方に苦戦していた．第 5 週目は，次回で配列を導入するうえでのサンプルとして，マウスの座標を配列に蓄積して軌跡を表示する**図 4** のプログラムを提示し，改良してもらうオープンな演習課題を与えた．配列の初期化宣言についても，C 言語と Processing では若干の違いがあるため，補足説明が必要であった．

第 6 週目も，配列の説明を C 言語で行ったうえで，C 言語の演習と Processing の課題を課した．C 言語の演習は，

月日を 3〜4 桁の整数で入力されたとき，カレンダー上の日付として存在するかどうかを判定するものであった．模範解答としては，月日を表す整数を 100 で割ったときの商と余りをそれぞれ月，日とし，あらかじめ初期化しておいた各月の日数を入れた配列を使って if 文で範囲チェックするという手順を想定していた．また演習とは直接の関係はないが，配列を高度に利用するために不可欠となる for ループについて簡潔に説明した．Processing 課題は，二重 for ループと 2 次元配列を使う例として，ボードゲーム風の画面を作成し，マウス操作によってコマや碁石を置くプログラム (**図 5**) を提示したうえで，マス目の数を増やしたり，コマの表現を変えてみるといったオープンな課題であった．

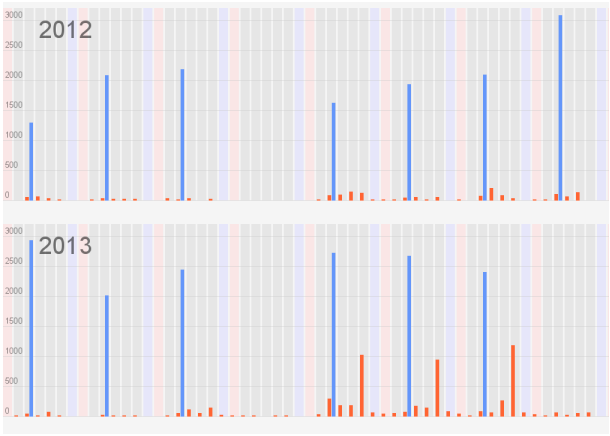

図 6 講義 A (火曜 1 限開講) のアクセス数の比較．青：授業時間内，赤：授業時間外．(注：2013 の第 8 週のみネットワークトラブルの影響で授業時間内ログが存在しない)

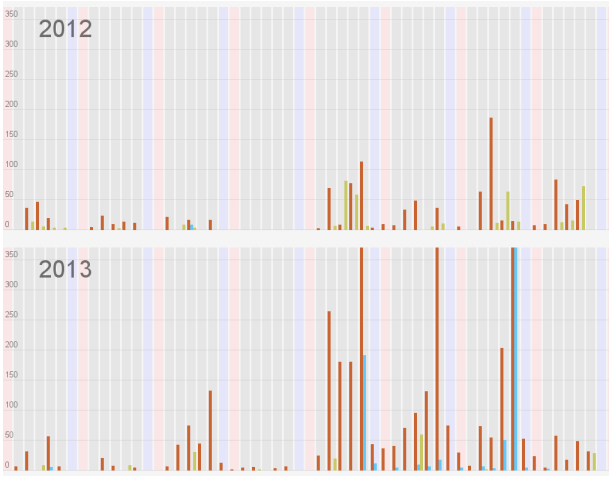

図 7 講義 A (火曜 1 限開講) の授業時間外アクセス数の比較．赤：学外，青：学内，黄：講義室

上記のように，C 言語での演習を行う前段階として，Processing を用いて概念や利用例を体験しイメージを持たせることを念頭に，条件分岐やループ，配列の導入を行なった．同様のアプローチで，関数や構造体 (クラス) についても事前導入による説明が可能であると考えている．ただし C 言語特有のポインタや関数呼び出しにおける明示的な値渡し/参照渡しについては，直接対応する表記が Processing 側にないため，工夫が必要である．

5. 評価

Processing に対応した Web IDE の導入による，授業内容に対する学生行動の変化をみるために，課題提出システムのアクセスログを解析した．Web IDE の導入により，Web ブラウザがあればいつでもどこでも手軽にプログラミングを再開できる環境が提供されることによって，学生の学習機会が増加していれば，課題提出システムへのアクセス数が増加したり，授業時間外や学外からのアクセス数が増えることが予想される．

図 6 に，講義 A(火曜 1 限開講) のアクセス数全体について，Processing および Web IDE を導入せず，C 言語のみの演習を行った 2012 年度と，Processing Web IDE を導入した 2013 年度を並べたグラフを示す．上段が 2012 年度，下段が 2013 年度であり，横軸は講義週を揃えた日にちを表している．2012 年度は 4 月 8 日の日曜日，2013 年度は 4 月 7 日の日曜日から表示している．薄い灰色で塗られた日は月曜日〜金曜日で，薄い青色が土曜日，薄いピンク色が日曜日である．縦軸はアクセス数を示している．授業時間内のアクセスを青色，授業時間外を赤色で示している．なお教員のアクセスは除いている．第 4 週は祝祭日のため，講義がなかった．図 6 をみると，第 5 週以降の授業時間外アクセスが増える傾向にあるが，2013 年度のほうが相対的に数が多いことが伺える．

図 7 に，講義 A(火曜 1 限開講) の授業時間外アクセス (図 6 において，赤色で表示した部分) を，アクセスした場所で区別したものを示す．黄色が講義室または演習室，青がそれら以外の学内，赤が学外を示している．このグラフから，2012 年度はほとんど学外アクセスがなかったが，2013 年度は学外アクセスが増えていることが読みとれる．

図 8, **図 9** および **図 10** , **図 11** に，講義 B と講義 C(それぞれ水曜 2 限，3 限開講) のアクセス数グラフを示す．読み方は図 6，図 7 と同様である．これらすべての講義について共通しているのは，水曜から金曜にかけてアクセスが増えていることであるが，これは各週の演習締切を基本的に金曜夜に設定していたことが原因である．特筆すべきは，講義 B(図 9) において，2013 年の連休で講義がなかった第 5 週のアクセス数が，2012 年のそれと比べて格段に増えていることである．これは演習締切を連休明けの 5 月 10 日に設定していたため，とくにオープンな演習課題についてじっくり取り組んだ学生がいたことを示しているといえる．講義 C(図 10 , 図 11) についても，2012 年の講義がなかった第 4 週は授業時間外・学外アクセスがほとんどないのに対し，2013 年の講義がなかった第 5 週については，授業時間外・学外での活動が増えていることが読みとれる．

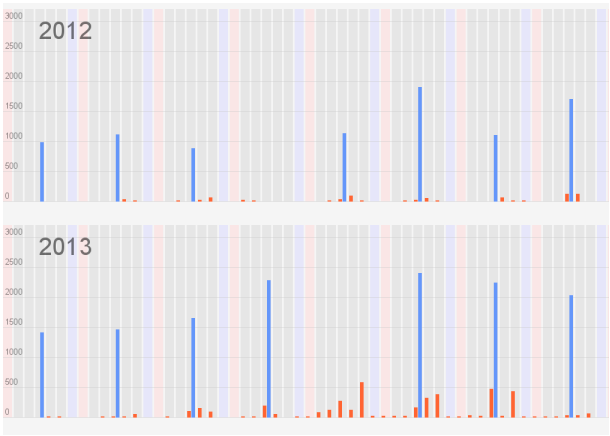

図 8 講義 B (水曜 2 限開講) のアクセス数の比較．青：授業時間内，赤：授業時間外

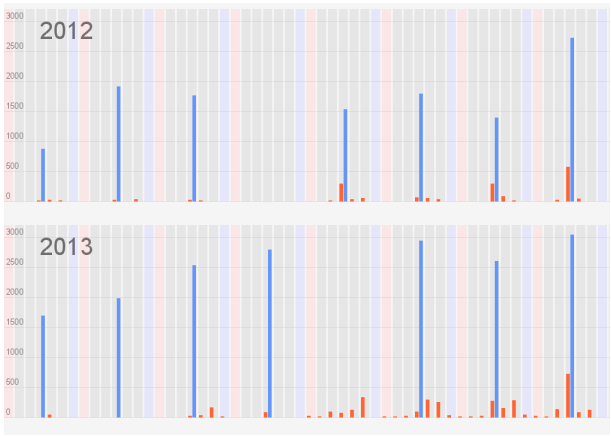

図 10 講義 C (水曜 3 限開講) のアクセス数の比較．青：授業時間内，赤：授業時間外

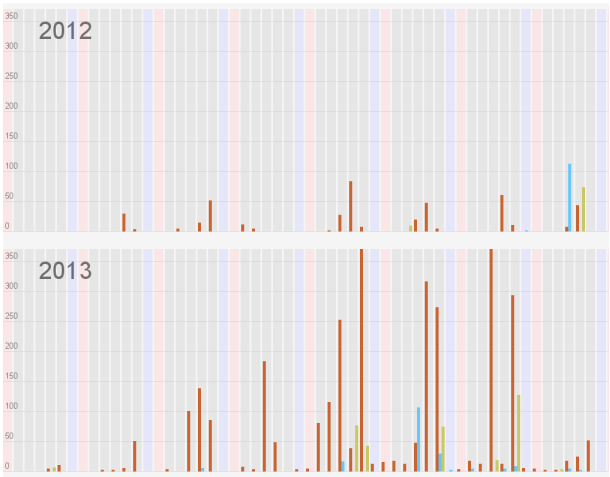

図 9 講義 B (水曜 2 限開講) の授業時間外アクセス数の比較．赤：学外，青：学内，黄：講義室

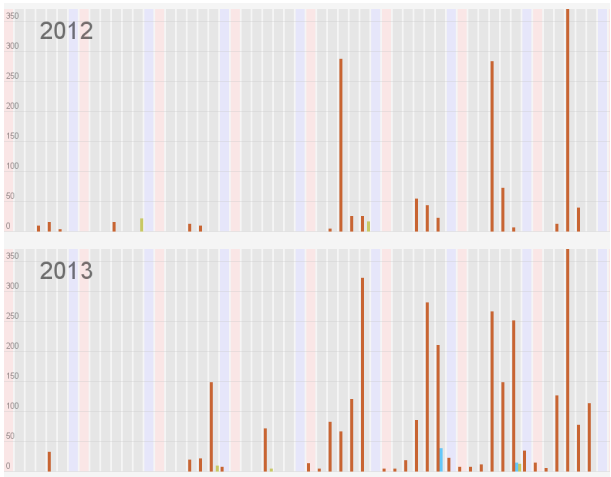

図 11 講義 C (水曜 3 限開講) の授業時間外アクセス数の比較．赤：学外，青：学内，黄：講義室

これらの結果から，Processing.js を手軽に試せる環境とオープンな課題を設定することにより，学生のプログラミング意欲を向上させる効果があることを確認することができた．オープンな課題は必ずしも全員の学生が意欲的に取り組むとは限らないが，意欲がある学生は精力的に取り組み，どんどん追求できる利点がある．教員も学生が提出した作品を 1 つ 1 つ観賞するのはとても刺激的であり，意外性のある力作を楽しんだり，また授業中に学生に紹介して，創作意欲を高められるというメリットもあった．一部の作品を付録 (**図 A·1**，**図 A·2**，**図 A·3**) に示す．

6. おわりに

我々は多人数クラスへのプログラミング教育に対応した Web IDE を開発し，これまで C 言語を中心に行っていた大学 2 年次の学生に対するプログラミング基礎科目について，Processing.js を併用した講義を試みた．アクセスログから，C 言語のみの演習と比べ，授業時間外のアクセス数の増加が確認できた．

Processing を用いると，編集結果を気軽に確認できるため，モチベーション向上の効果が期待できる．しかし，動作の詳細については気づかなかったり妥協してしまうなど，確認が困難な一面もある．そのため基本的な理解度の確認については，C 言語の演習で行い，ループや条件分岐の概念については Processing で導入するというアプローチを採用した．

Processing.js 以外にも，近年では Blockly[15] や Web 版 Scratch[16] など，Web ブラウザをプラットフォームとして動作するプログラミング環境が次々と登場している．表記における C 言語との親和性の高さから，今回は Processing.js を用いたが，上記のブロック配置型のプログラミング環境も手軽さという点では同様であるため，今後活用方法を検討していきたいと考えている．

参考文献

[1] Jenkins, T.: The motivation of students of programming, *SIGCSE Bull.*, Vol. 33, No. 3, pp. 53–56 (online), DOI: 10.1145/507758.377472 (2001).

[2] Martins, S., Mendes, A. and Figueiredo, A.: A strategy to improve student's motivation levels in programming courses, *Frontiers in Education Conference*

(FIE), 2010 IEEE, pp. F4F–1–F4F–7 (online), DOI: 10.1109/FIE.2010.5673366 (2010).
[3] Team, T. C.: CodeSpells. https://sites.google.com/a/eng.ucsd.edu/codespells/.
[4] Barnes, T., Powell, E., Chaffin, A. and Lipford, H.: Game2Learn: improving the motivation of CS1 students, *Proceedings of the 3rd international conference on Game development in computer science education*, GDCSE '08, New York, NY, USA, ACM, pp. 1–5 (online), DOI: 10.1145/1463673.1463674 (2008).
[5] Jiau, H. C., Chen, J. C. and Ssu, K.-F.: Enhancing Self-Motivation in Learning Programming Using Game-Based Simulation and Metrics, *IEEE Trans. on Educ.*, Vol. 52, No. 4, pp. 555–562 (online), DOI: 10.1109/TE.2008.2010983 (2009).
[6] Feldgen, M. and Clua, O.: Games as a motivation for freshman students learn programming, *Frontiers in Education, 2004. FIE 2004. 34th Annual*, pp. S1H/11–S1H/16 Vol. 3 (online), DOI: 10.1109/FIE.2004.1408712 (2004).
[7] : Processing. http://processing.org/.
[8] Reas, C. and Fry, B.: *Getting Started With Processing*, O'Reilly Media (2010).
[9] 菊池　誠：プログラミング，何をどう教えているか：Processing によるプログラミング教育，情報処理，Vol. 52, No. 2, pp. 213–215 (2011).
[10] 有賀妙子，森　公一：フィジカル・インタラクションを使ったメディア造形基礎教育におけるプログラミング学習の実践，情報処理学会論文誌，Vol. 52, No. 12, pp. 3096–3105 (2011).
[11] 大見嘉弘，滑川敬章，永井保夫：情報系高校におけるセンサを利用したプログラミング教育の実践，情報処理学会研究報告，Vol. 2012-CE-114, No. 5, pp. 1–7 (2012).
[12] Processing.js Team: Processing.js. http://processingjs.org/.
[13] Bader-Natal, A.: Sketchpad. http://sketchpad.cc/.
[14] OpenProcessing: OpenProcessing. http://processingjs.org/.
[15] Project, B.: blockly — A visual programming editor. https://code.google.com/p/blockly/.
[16] Project, T. S.: Scratch — Imagine, Program, Share. http://scratch.mit.edu/.

付　　録

A.1　学生が作成したプログラムの例

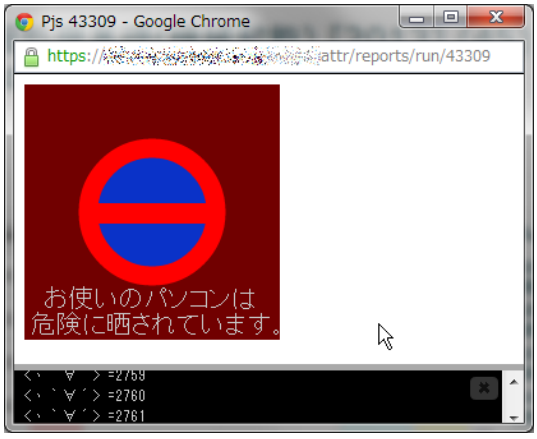

図 A·1　アニメーション作成課題 (背景がフラッシュする)

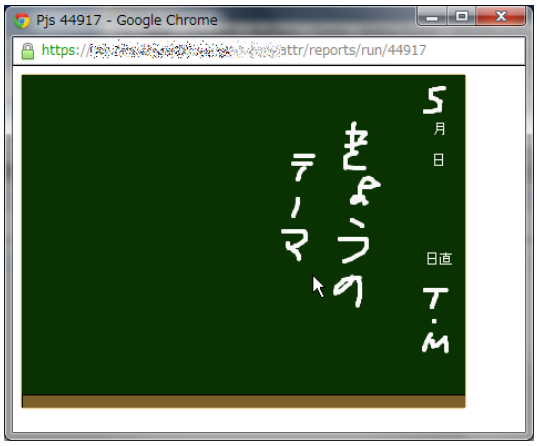

図 A·2　お絵かきツール課題 (黒板)

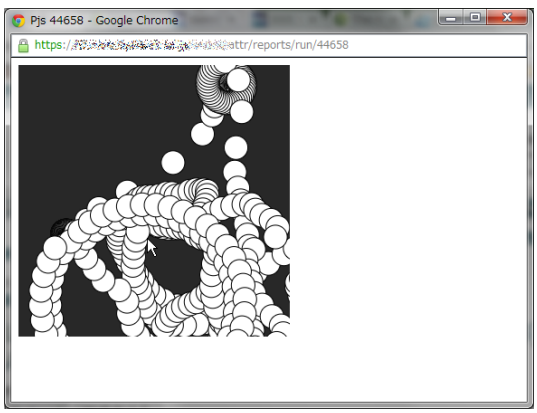

図 A·3　お絵かきツール課題 (キャンバスが螺旋状に周る)