Vol.2017-CE-138 No.2
2017/2/11

Web ブラウザを用いたプログラミング学習支援環境 Bit Arrow の設計と評価

長島 和平[†1]　長 慎也[†2]　間辺 広樹[†3]　兼宗 進[†4]　並木 美太郎[†1]

高等学校の次期指導要領では，プログラミングが必履修科目の単元として含まれることが決まり，適切なプログラミング教育のための環境やツールを用いることが重要になる．これまで教科書で扱われていたプログラミング学習では，テキストエディタでコードを記述し，ブラウザで実行を行う形態で JavaScript を用いることがあった．しかし，この形態ではファイルの扱いが煩雑であることや，エラーの発生やメッセージがわかりにくいこと，プログラムをローカルファイルで保存するため自宅での予習復習を行いにくいことのような欠点がある．そこで著者らはブラウザだけでプログラミング学習を行える環境 Bit Arrow を開発した．本環境では，ファイルをクラウド上に格納することでネットワークとブラウザがあればいつでもどこでも学習でき，エラーメッセージとその原因となった場所を表示する．また、初学者にはわかりにくいオブジェクト指向とイベント駆動が中心となる JavaScript を初学者向けに改善した．本報告では，利用者のログをもとに，初学者向けの本プログラミング環境の設計と評価について論じる．

Design and Evaluation for Bit Arrow : Web-based Programming Learning Support Environment

KAZUHEI NAGASHIMA[†1]　SHINYA CHO[†2]　HIROKI MANABE[†3]
SUSUMU KANEMUNE[†4]　MITAROU NAMIKI[†1]

1. はじめに

次期学習指導要領に関する答申のとりまとめ[1]には，情報科新科目のイメージとして，共通必履修科目「情報 I」の中にプログラミングが含まれており，高等学校においてプログラミングが必修になる．現在の情報科は，「社会と情報」と，「情報の科学」があるが，プログラミングを扱う「情報の科学」を履修している生徒は約 2 割にとどまっている．今後プログラミング必修に向けて，教員と生徒両方を支援する適切な学習環境を用いることが重要である．

現在教科書に掲載されているプログラムは，JavaScript や VBA など，コンピュータに標準搭載されている環境で扱えるものを使っていることが多い．しかし，これらの環境は教育用に作られていないことがあり，初学者が触れることには適さない．授業にあたり，教科書に掲載されたプログラムではなく生徒の興味を惹く題材を扱った実践報告もある[2][3]．これらの実践では，ゲームなどの作品制作で学習者の意欲を刺激するものが多いが，インストールなど事前準備を必要とする環境を用いていることが多い．準備が不要な環境として，Web ブラウザを用いることもある．Web ブラウザには，学習支援として課題の設定や自動採点を行う環境もある[4]が，実践は大学で課題を解くかたちの演習で用いられており作品制作には向かない．

本研究では，準備を必要としない環境を用い，エラーからの復帰を支援する環境の実現を目的とする．3 章で述べる機能で学習者の支援を行えているかどうかを，利用者のログから評価する．

本論文は，まず 2 章でプログラミング学習環境の課題について述べ，3 章で本環境の設計と機能について述べる．4 章で本環境が利用されたログからの評価を述べ，5 章で結論と今後の課題を述べる．

2. プログラミング学習環境の課題

2.1 JavaScript を用いたプログラミング教育

高等学校の科目「情報の科学」の教科書には JavaScript が使われることがある[5][6]．教科書で扱われるプログラムは，計算結果や，ソーティングアルゴリズムを用いた数値の並べ替え，暗号の解読の結果などを画面に表示させるような単純なものである．この教科書では，順次，分岐，反復といったプログラミングの基本的な概念を教えている．JavaScript は，高校生の多くが所持するスマートフォンなどで見られている Web アプリケーションの制作に使われており，生徒の身近で活用されているプログラムと関連付けやすい．JavaScript は，大学で行われるプログラミング学習

†1 東京農工大学
Tokyo University of Agriculture and Technology
†2 明星大学
Meisei University
†3 神奈川県立柏陽高等学校
Hakuyo High School
†4 大阪電気通信大学
Osaka Erectro-Communication University

でよく用いられる言語にも近く，大学での学習への接続の面でも優れている．

教科書には JavaScript を用いたプログラミングを行う環境として，テキストエディタとブラウザを使うよう説明がある．テキストエディタとブラウザは，PC に標準で装備されていることから，教室の PC にアプリケーションのインストールなどの準備をせずにプログラミングを行うことができる．しかし，標準で使用可能なテキストエディタはテキストファイル作成・表示のために必要最小限の機能しか備わっておらず，テキストエディタとブラウザを用いた学習には次のような問題がある．

(1) エラーへの対応が難しい

プログラミングに不慣れな初学者は，タイプミスが原因となる文法エラーが発生させることが多い．しかし，テキストエディタとブラウザを演習に用いた場合，エラーが発生したことは画面に表示されないため，学習者はエラーが発生したかどうかわからない．ブラウザでファイルを開いたとき，想定通りの動きをしなかった場合は，エラーの有無を確認するために，開発者ツールやコンソールを表示させなければならない．また，エラーメッセージは，初学者が読んですぐに原因が特定できるとは限らない．エラーが発生している場所を探すのに手間取ると，エラーからの復帰に時間がかかる．エラーの修正はプログラミング学習の本質ではない．エラーの修正に多くの時間を費やすことなく，プログラミングの基本的な概念の学習に時間を長く使えるようにすることが重要である．エラーの原因などが分からなかった場合，教科書などに掲載されたサンプルプログラムを作成中であれば，サンプルプログラムと一文字ずつ見比べることでも修正可能ではあるが，正解が掲載されない課題のプログラムなどを作成している際にはエラーの原因を突き止めることが難しくなる．

プログラムでは，対応する括弧を書き忘れることや，違う種類の括弧を組み合わせてしまうことなどもエラーの原因になる．分岐や反復を学習することで括弧を書く機会も増えるが，テキストエディタでは記号の間違いや書き忘れを防ぐことが難しい．

また，プログラムには文法エラーだけでなく論理エラーがある．分岐や反復を書くとき，ブロックの範囲が間違っていると文法エラーがなくても思い通りに動作しない．インデントによってその範囲をわかりやすくすることで，想定通りに動かなかったときに原因が特定しやすくなる．しかし，テキストエディタではインデントが自動でつかないため，学習者自身で付ける必要がある．このとき，インデントを付け間違えたりすると，想定通りの動きをしなかったときにかえって混乱を生む原因にもなってしまう．

プログラミングの演習における学習者のつまずきの原因の中でも特に影響が大きいのがこれらエラーによるもので，大きく分けると文法エラーが原因でプログラムが実行できないこと，実行はできるが論理エラーがあり想定通りの動作をしないことの 2 種類がある．このうち文法エラーは，単純な入力ミスなどが原因となることが多く，プログラミングの本質の学習とは異なるが，論理エラーはプログラムの動きを正確に把握しなければ直すことができないことから，プログラムの構造の学習につながる．文法エラーの割合を減らすことで，プログラミングの基本構造の学習に取り組むことができる．

(2) プログラムの保存や実行が煩雑

テキストエディタを用いて JavaScript プログラミングを行うと，実行はブラウザで行う必要がある．そのために，ファイルをテキストエディタとブラウザで開く必要がある．通常，HTML ファイルはブラウザで開かれるため，テキストエディタでファイルを編集するためには，ファイルを右クリックし，プログラムから開く，を選択してテキストエディタを選ばなければならない．また，編集と実行を別のアプリケーションで行わなければならないため，ウィンドウを頻繁に遷移する必要がある．

(3) 初学者が JavaScript を使用する上での問題

JavaScript は HTML と連携し，HTML で静的な画面の見た目を設定し，それを JavaScript で動的に操作することが多い．これは Web アプリケーションの作成に用いられているため，結果に動きを持たせたり画像を用いたりすることで，コンソール上に文字が表示されるだけの実習に比べて視覚的に学習者の興味を惹きやすい．

しかし，そのために必要な DOM 要素の操作や教科書で使われる `document.write` 命令などには，オブジェクト指向の考え方が含まれている．オブジェクト指向の説明はプログラムが掲載された教科書にはない．大学におけるプログラミング教育においても，入門教育では C 言語が用いられることが多く，オブジェクト指向の概念はその後 Java などで終盤に教えられることが多い．このことから，教科書でも教えられていないオブジェクト指向の概念は，極力避けてプログラムを記述させるべきである．

JavaScript はボタンのクリックや一定の時間間隔ごとの実行などによるイベント駆動で動く．教科書では，順次・分岐・反復の概念を教えているが，イベント駆動については触れられない．また，イベント駆動プログラムを記述すると，プログラムの流れが把握しにくくなる．学習者の意欲を高めるために動きを付けたプログラムを作成しようとすると，イベント駆動を使用しなければならず，イベント駆動の使用により反復が使えないことがある．

教科書に記載されているような，通常の JavaScript プログラムは，HTML ファイルの script タグの中に記述する．しかし，この状態は HTML と JavaScript が同一ファイル内に混在する形となる．HTML 内に JavaScript が書かれて両者が混在することは，初学者の理解の妨げになる．

2.2 プログラミング学習の進捗の把握

プログラミング演習時に，学習者の進捗状況の把握をすることで演習を支援する研究もあり，教員を支援するシステムなどが開発されている[7]．このシステムでは，クラス全体の課題の作業進度，エラー，作業の遅れ，プログラムなどの学習状況を抽出し，これらのデータから，教員が必要な指導を行うことを支援している．しかし，この環境は教員側へのサポートに力を入れており，学習者が自ら課題を完成させることの支援ではない．

3. Bit Arrow の設計

本研究では，2 章で述べた課題を解決するためのプログラミング学習環境 Bit Arrow を提案する．著者らは昨年度 Web ブラウザで学習可能なプログラミング環境 Bit Arrow(旧 JSLesson)を開発し，高等学校における授業で用いた[8]．この実践では，ゲームを制作するチュートリアル教材を用意し，それに沿ってプログラミングを行う形での演習授業を行った．エラーが起きた場所の表示など，学習者がつまずきに対するサポートを行い多くの生徒がゲームを制作することができた．本環境でエラーの発生やエラーからの復帰を手助けすることにより，教科書に掲載されている順次・反復・分岐といったプログラミングの構成要素の学習支援を目指す．利用者が起こしたエラーの割合やエラーに悩んでいた時間を元に，プログラミング学習支援の効果をログから評価する．

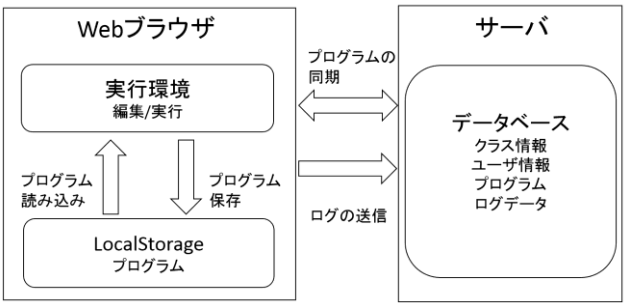

図 1 システムの構成図

図 1 に，本環境のシステムの構成を示す．本環境では，プログラミングはすべて Web ブラウザ上で行う．記述したプログラムはブラウザのローカルストレージに保存される．このプログラムは実行などのタイミングでクラウドと同期される．学習者が実行を行うと，実行結果か，エラーがあればエラーメッセージやエラーの場所などが表示される．この情報で，学習者のエラー修正を支援する． Web ブラウザ上の学習環境では，ブラウザで書かれたコードを，サーバに転送しコンパイルや実行を行うことが多い．しかし，その実装では実行ごとに通信が行われるため，ネットワークへの負担が大きい．本環境では，コンパイルや実行をクライアント側で行うことで，サーバとの通信を減らす．記述されるプログラムについても，HTML と JavaScript を別のファイルに分けて書けるようにしたほか，前年度の実践でも用いたオブジェクト指向やイベント駆動を意識せずに DOM 要素を操作するための短い命令などを用意した．

プログラミング学習支援環境に実装した機能を次に説明する．

3.1 Web ブラウザの編集・実行環境

Web ブラウザはコンピュータに標準で装備されているため，特別な準備をすることなく授業に導入することができる．授業以外においても，自宅での学習や課題の制作などを容易に行うことができるようになる．

(1) 一画面での編集と実行

従来の環境では，テキストエディタでプログラムを記述し，ブラウザで実行結果を見る，というウィンドウの移動が必要であった．Web ブラウザでコーディングと実行を行うことで,エディタと実行画面を遷移する必要がなくなる．同時に，テキストエディタとブラウザの両方でファイルを開く必要がなくなるため，プログラムから開くようなわかりにくいファイル操作も不要になる．図 2 に，プログラムの編集画面を示す.左には,ファイルリストが並んでおり，ファイルを作成すると HTML と JavaScript のファイルがそれぞれ一つずつ作成される．ファイルを開いた状態で実行を押すと，画面上に実行画面ダイアログを表示し，すぐに実行結果を確認することができる．

(2) クライアント側での実行

プログラムをサーバへ送信してのコンパイルや実行を行わず，クライアント側で実行を行うことで，サーバとの通信も削減できる．授業中に実行が集中することなどでネットワークのトラブルが発生したとしても実行は行うことができる．

図 2 Bit Arrow のコード編集画面

3.2 つまずきへの対策

(1) 文法エラー発生時の対策

文法エラーが起きた時，通常ブラウザでエラーを確認するためには開発者ツールなどを用いなければならない．しかし，開発者ツールやコンソールを開くという動作は，学習者が慣れ親しんだ行動ではない．また，表示されるエラーメッセージは，図 3 に示すような内容である．従来の環

境では，このメッセージを読んでエラーに対処する必要があった．つまり学習者は，実行が上手くいかなかったとき，次のようなステップを踏む必要があった．

① ブラウザのツールなどのメニューから開発者ツールやコンソールを開く
② エラーメッセージを解読し，間違いの原因やそれが起きた場所を特定する
③ テキストエディタで該当箇所を確認し修正する

初学者にとって，開発者ツールを開くことや，図 3 のようなエラーメッセージからその原因や場所を特定することは困難である．

Bit Arrow ではエラーが発生した際には図 4 のようなエラーダイアログを画面上に表示することで学習者へ通知を行う．このダイアログは，文法エラーがあるファイルを実行したときすぐに表示される．このことで，エラーの有無や原因の確認のために開発者ツールやコンソールを開かせる手間を省くことができる．また，ダイアログによる通知の内容は，原因を示すエラーメッセージを表示させるだけではなく，エラーが発生した場所周辺にマークを同時に表示させた．これを学習者に提示することで，エラーが起こった原因やその場所の特定を支援することができる．これにより，従来の環境よりも簡単にエラーを確認することができ，エラーからの素早い復帰を支援する．

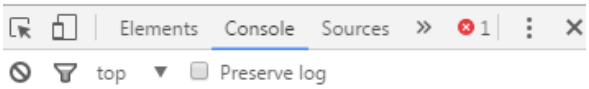

図 3 ブラウザ(Google Chrome)のエラーメッセージ

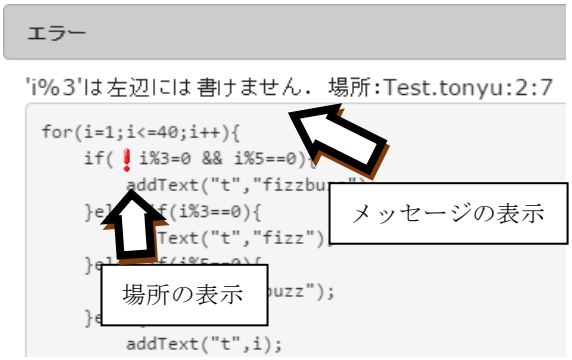

図 4 Bit Arrow のエラーダイアログ

(2) 入力補助によるエラー発生の防止

初学者の打ち間違いや打ち忘れなどが元となることが多い文法エラーの中には，エラーの発生をあらかじめ防ぐことができるものもある．例えば，if 文や for 文のブロックを表す波括弧を開いた後に閉じ忘れることがある．波括弧の数が合わないとき，その原因がプログラムの途中にあったとしても，エラーメッセージではプログラムの最後で閉じ波括弧が足りないかのような指摘をされることもある．このときプログラムの動作を正確に把握していなければ適切な場所に閉じ括弧を入れる修正を行うことが難しい．そこで，図 5 に示すように波括弧を開いた後に改行するタイミングで対応する閉じ波括弧を自動入力することで，打ち忘れを防ぐ．波括弧のほかにも，丸括弧や HTML の閉じタグ，ダブルクォーテーションのように対応して使われる記号などを自動で入力するなどで，エラー発生の減少を支援する．

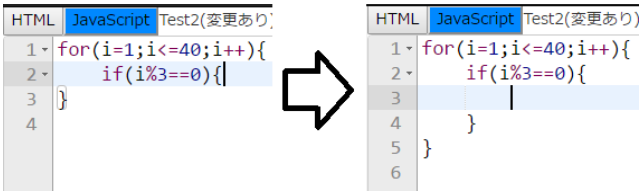

図 5 対応する括弧の自動入力

(3) 自動インデントによるブロックの範囲の把握

反復や分岐を多く使うプログラムを扱うと，ブロックの数が増える．同時に入れ子構造も使われるようになる．このとき，適切なインデントを付けることでプログラムが見やすくなる．しかし，初学者がプログラムのインデントを手動で正しく付けることは難しく，プログラムの途中に新しいブロックを入れたり，プログラムの一部を新しくブロックで囲んだりすることで，ブロックの範囲が分かりにくくなってしまう．ブロックの範囲が分からないと，プログラムの修正もしにくくなる．この問題に対し，Bit Arrow では図 6 に示すように実行や保存を行うタイミングで自動的にインデントを付ける．波括弧の数が合わないときにも，インデントを見ることでどこを閉じ忘れているかなどに気付きやすくする．また，実行はできたが想定した動きをしない論理エラーのときにも，ブロックの範囲を見やすくすることで対応しやすくなる．これらのことが，エラー発生数の減少と，エラー発生時の修正時間の短縮を支援する．

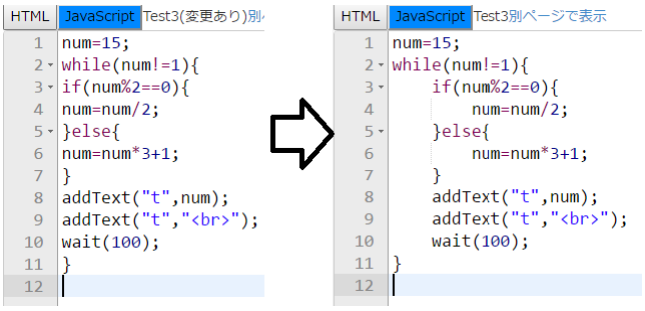

図 6 自動インデント

3.3 初学者用 JavaScript

(1) オブジェクト指向を使わないプログラム

教科書に掲載されたプログラムなどには，結果などの表示に `document.write` が用いられている．また，動きのあるプログラム作成のために通常の JavaScript で DOM 要素の操作などを記述すると，`document.getElementById`

命令を使い要素を取得し，取得した要素に対して操作を行う必要がある．これらのプログラムにはオブジェクト指向の考え方が含まれており，オブジェクト指向についてはじめに教えなければならなくなる．また，これにより記述しなければならないプログラムも長くなってしまい，初学者の使用に適さない．例えば，DOM 要素を移動させようとしたとき，通常の JavaScript では要素を取得するプログラム，要素の座標を変えるプログラムを分けて書かなければならない．これを初学者に書かせると，タイプミスによるエラーの増加や覚えなければならない項目の増加などの問題がある．オブジェクト指向の考え方は，プログラミングにある程度慣れてからの学習が望ましいと考え，Bit Arrow 版 JS ではオブジェクト指向の考え方を意識せずにプログラムを記述できるようにした．例えば，Bit Arrow 版 JS で DOM 要素を移動させようとすると，move 命令一つだけで記述することができる．要素を操作する命令では，命令の引数に要素名を指定することとした．これにより，どの要素を操作するかを分かりやすく把握しながら，オブジェクト指向を意識せずに要素に対する操作を行うことができる．

次に，ログから実際に利用されていたプログラムの例と，そのプログラムを通常の JavaScript に書き換えたものを挙げる．ログから収集したプログラムの中には，じゃんけんのプログラムがあった．ランダムで 0 以上 3 未満の数値を生成し，その値によって画面上にグー，チョキ，パーが表示されるものである．通常の JavaScript でこれを記述しようとすると，図 7 のようなプログラムになる．このプログラムでは，乱数を取得するためにまず Math.random で 0 から 1 未満の小数を生成し，その値に 3 をかけて乱数を 0 から 3 未満にするというステップを踏まなければならず，この計算式についても説明しなければならないため手間が多い．また，もし整数の乱数値を求めたいときには，これに Math.floor 命令を付け足す必要がある．

```
<html>
<body>
<script>
jcode=Math.random()*3;
if(jcode==0){
    document.write("Gu-");
}else if(jcode==1){
    document.write("Choki");
}else{
    document.write("Pa-");
}
</script>
</body>
</html>
```

図 7 これまでの環境のプログラム例(じゃんけん)

このじゃんけんプログラムを，Bit Arrow 版 JS で実装すると図 8 のようになる．乱数の取得には rnd 命令が用いられている．この命令は，0 から引数未満の整数を返すものである．また，通常の JavaScript では，document.write 命令を用いていた画面への表示は， setText 命令で，HTML 側に作成しておいた name 属性が n の要素に結果が表示されるようになっている．これにより，通常の JavaScript で使用されていたオブジェクト指向を用いずにプログラムの記述ができる．

```
jcode=rnd(3);
if(jcode==0){
    setText("t","Gu-");
}else if(jcode==1){
    setText("t","Choki");
}else{
    setText("t","Pa-");
}
```

図 8 Bit Arrow 版 JS のプログラム例(じゃんけん)

(2) イベントループの記述

JavaScript は，ボタンが押されたときや，一定の時間経過などのイベントによりプログラムが実行される．しかし，イベント駆動はオブジェクト指向と同様に教科書での説明もなく，プログラミングの基本的な概念を習得した後の学習項目である．例えばゆっくり動くようなプログラムを記述しようとすると，JavaScript では一定間隔でプログラムを実行する setInterval のようなイベント駆動を用いなければならない．

ログから収集したプログラムの中に，FizzBuzz のプログラムがあった．FizzBuzz のプログラムは，1 から順にカウントアップしていき，3 の倍数では Fizz, 5 の倍数で Buzz, 15 の倍数で FizzBuzz と表示し，それ以外は数値を表示するものである． 図 9 に，Bit Arrow 版 JS で記述された FizzBuzz のプログラムを示す．従来の環境では document.write を用いる必要があるのに対し，Bit Arrow では，表示に addText を用いているという差は，じゃんけんのプログラムと同様であるが，大きな違いは wait 命令にある．通常の JavaScript によるプログラムでは，for 文で 40 回分の繰り返しを行った実行結果がすべて一度に表示される．しかし，Bit Arrow 版 JS で記述された図 9 のプログラムでは，wait 命令が使われている．wait 命令は，指定された時間だけ処理を中断するため繰り返しをゆっくり行うことができ，順番に一つずつ出力結果が足されていくように表示される．この命令を入れるだけで，実行結果に動きをもたせることができる．このような書き方は，通常の JavaScript では書くことができない．同じようにゆっくりと結果を表示させようとすると，setInterval を用いなければならず，これを用いると反復の for 文や while 文を使うことができない．また，setInterval を用いるとプログラムは永久に実行が続く

が，図 9 のプログラムのように決まった回数だけゆっくり繰り返そうとすると，一定回数で setInterval のイベントを停止させるような命令を書き足さなければならない．

```
for(i=1;i<=40;i++){
    if(i%3==0 && i%5==0){
        addText("t","FizzBuzz");
    }else if(i%3==0){
        addText("t","Fizz");
    }else if(i%5==0){
        addText("t","Buzz");
    }else{
        addText("t",i);
    }
    addText("t","<br>");
    wait(100);
}
```

図 9 Bit Arrow 版 JS のプログラム例(FizzBuzz)

このように，Bit Arrow 版の JavaScript は，これまで使われてきたプログラムに少し書き足すだけで，プログラムに動きを付けることができ，実行結果で学習者の興味を惹くような工夫ができる．

(3) 要素の操作を簡単にするライブラリ

前述したとおり，JavaScript は HTML の要素を操作することなどで用いられるにもかかわらず DOM 要素を動かすためにはオブジェクト指向を用いたり，動きを付けるためにイベント駆動を用いたりと高度なプログラミングスキルが必要とされる．そこで，Bit Arrow では，HTML の要素を簡単に操作できるライブラリを用意した．ライブラリは，昨年度の実践で用いたもので，ゲーム開発環境 Tonyu System[9]で用いられる Tonyu 言語を元としている．その後 AltJS として Web ブラウザ上で動作する Tonyu System 2 が開発されている[10]. Bit Arrow 版 JS のライブラリの命令一覧を表 1 に示す．

表 1 Bit Arrow 版 JS のライブラリ

命令	説明
move(要素名,x,y)	要素を x,y 座標に移動
rotate(要素名,角度)	要素を指定の角度回転
resize(要素名,横,縦)	要素を指定の比率に拡大/縮小
setText(要素名,文字)	要素に文字を設定
addText(要素名,文字)	要素に文字を書き足す
onClick(要素名,命令)	要素が押されたときの動作を設定
onTouch(命令)	画面が押されたときの動作を設定
transform(要素名, 角度,横,縦)	要素の rotate と resize を同時に行う
wait(時間)	指定された時間(ms)処理を中断
rnd(値)	0～値未満の整数を返す
setBGColor(色)	指定の色に背景色を変える

HTML は，画面の見た目を設定するもので，JavaScript は DOM 要素の操作を行うものであり，別の働きをする HTML と JavaScript を分けて記述させることで，プログラムを分かりやすく見せることができると考えた．また，一つのファイルに HTML と JavaScript を記述すると，ファイルの中身が複雑に見え，特にプログラムが長くなるにつれ学習者の理解の妨げになる可能性がある．Bit Arrow では，これまで行われてきた演習と同じように HTML の script タグに JavaScript を記述するだけでなく，HTML と JavaScript を分けて記述させることもできるようにした．

3.4 ログの収集

本環境では，実行単位で実行された時間や実行結果，プログラムのログを収集する.収集したデータを次に挙げる．

(1) ユーザ名
(2) 実行を行った時刻
(3) 実行結果(実行成功，エラー)
(4) エラーメッセージ
(5) 実行時のプログラム

ログは実行ごとに収集し，学生の学習過程を追跡できるようにした．これらのデータを元に，利用状況の調査を行う．

4. Bit Arrow の利用とログからの評価

4.1 Bit Arrow の利用状況

Bit Arrow で Bit Arrow 版 JavaScript を利用していた 170 名の実行時のログを調査対象とした．なお，3.3 節で例示したプログラムは，この 170 名の利用者が実際に記述していたものである．このユーザが利用していた 10/6 から 11/8 までのログを調べると，実行の総数は，18282 回となり，約一ヶ月間で一人当たり 107.5 回の実行を行っていた．また，この実行のうち，実行成功と文法エラーの回数の割合を調べたところ, 表 2 の結果となった. 実行成功回数とは，実行を行ったとき，エラーが起きずに実行結果が画面に表示された回数である．実行成功の割合が約 79%となり，エラーの割合が少なかった．Bit Arrow はひとつのウィンドウですぐに実行画面が見られることから手軽に実行することができ, 実行回数の増加にもつながっていると考えられる．

表 2 実行とエラーの回数と割合

実行成功	文法エラー
14483 (79.2%)	3799 (20.8%)

4.2 文法エラーによるつまずきの検証

Bit Arrow は文法エラーが発生した場合，その発生場所の表示などでなるべく早いエラーからの復帰を支援する．利

用者がエラーの修正にかかった時間を調べたところ，文法エラーが発生したとき，そのエラーを修正するまでにかかった平均時間は，約 105.0 秒であった．これは，エラーが連続で発生したかどうかなどは考慮せず，前回実行時が文法エラーでない状態のときに，文法エラーが発生した時間から，次に実行成功するまでの時間を計測している．エラー発生から約1分45秒で実行を成功することができていることから，学習者が自力でエラーから復帰することができていると考えられる．また，文法エラーの内容は，打ち間違いによるものが多く，発生したエラーのうち 89%に対してその発生場所を通知することができていたと考えられる．この復帰時間から，学習者が自力でエラーから抜け出すことができ，その結果エラーからの復帰にかかる時間を短くすることができていると考えられる．素早い文法エラーからの復帰を支援することで，プログラムの構造などの本質的な部分の理解に費やす時間を増やすことができる．また，生徒が自力でエラーから抜け出すことを支援することで，授業などにおいては教員の負担も軽減させることができると考えられる．一方，発生場所の通知ができなかった約 10%のエラーは命令名の間違いなど実行時に発生するものであった．実行中に起こるエラーについてもその発生場所の表示を行うことで，エラーからの復帰をさらに支援することができると考えられる．また，一部には JavaScript ファイルに HTML を記述したことによるエラーもあった．HTML と JavaScript のファイルを分けることでプログラムを分かりやすくする狙いがあったが，このログで用いられていたじゃんけんや FizzBuzz のような短いプログラムにおいては，一つのファイルに書くことのほうが簡単になると思われる．

4.3 論理エラーの検証

論理エラーは，実行は成功するが期待した動作をしないエラーであるため，実行できたかどうかを見るだけでは検出することができない．そこで，利用者のログから，図 10 など数種類のプログラムを作る過程の実行成功時のログを観察した．そのログの中から，多かった論理エラーの主な原因とそれが起きていた回数を表 3 に示す．もっとも多かった論理エラーは，必ず改行してしまうというもので，図 10 で示した最後の if 文が抜けていたものである．これは，図 10 のプログラムを作成する前に，改行部分がないプログラムを作成していたことから，それを改良する過程で起こった論理エラーであると考えられる．また，次に多かった論理エラーは二重ループのループ変数に同じ変数を使っていたことであった．これは，図 10 と同じ出力結果のプログラムを作るために，if 文ではなく二重ループを用いようとした利用者のログに見られた．他にも，表 3 に示したとおり分岐や反復の使い方が原因となっている論理エラーがいくつか見られた．こうした分岐や反復の使い方に起因する論理エラーの原因を考えさせることで，分岐や反復の概念を学習させることができていると考えられる．

```
for(i=1;i<=50;i++){
    if(i%2==0){
        addText("t","<img
                src=images/neko1.png>");
    }else{
        addText("t","<img
             src=images/bluefish.png>");
    }
    if(i%10==0){
        addText("t","<br>");
    }
}
```

図 10 交互に画像表示するプログラム

表 3 主な論理エラーの原因と出現回数

原因	回数
必ず改行してしまう	273
二重ループに同じ変数を使用	88
ループ条件の誤り	53
表示する画像の個数違い	45
ループ変数加算忘れ	43
ループ変数初期化なし	15
分岐判定とループ変数が違う	14

図 11 に，実行の割合を示す．ここでは図 10 などのプログラム作成に取り組んでいた間の結果を示すため，4.2 で述べた全体の文法エラーの割合とは異なる．文法エラーの割合は 35.7%と，全体の文法エラーの割合に比べ増加したが，作成途中の実行や，完成したプログラムの実行などが残りを占めた．

図 12 に，作成途中の実行結果の内訳を示す．図 10 のプログラムを作成中の実行ログから，間違えている箇所を分岐の使い方，反復の使い方，URL の間違いなど文法エラーにならない打ち間違い，余分な addText などにより表示される結果が異なる表示の間違いに分け，それぞれの割合を示す．最も多いのは表示される結果が異なるものであった．これは以前に作った FizzBuzz のプログラムを改造しながら作成している利用者が多く，FizzBuzz の数値の表示などが残っていたことなどが原因で，28.5%を占めた．この間違いに対応するために，表示がどのタイミングで行われているかなどのアルゴリズムを考えさせることができる．また，表 3 で挙げたような分岐や反復の使い方に問題がある論理エラーも多く，分岐と反復を組み合わせた使い方の問題と合わせて 44.3%となり，作成中の実行の半分近くで分岐や反復の使い方や動作について考えさせることができた．一方，画像を表示するプログラムであるため，画像の URL

の打ち間違いなどに起因するプログラミングの本質とは関係のないエラーも 22.1%あった．

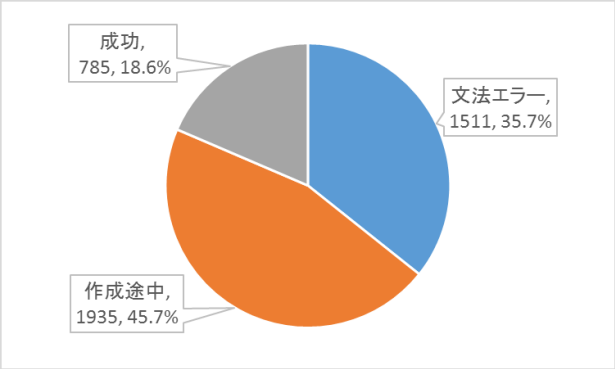

図 11 プログラム作成中の実行結果の割合

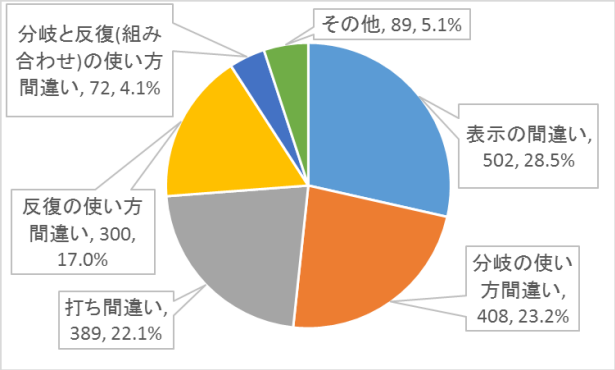

図 12 作成途中の実行結果の内訳

Bit Arrow を利用することで文法エラーよりも作成途中のプログラムの実行が多く，利用者にプログラミングの構造の理解に時間を使わせることができたといえる．作成中の実行結果では，分岐・反復の問題と表示の間違いが原因の論理エラーが合わせて約 3/4 となり，Bit Arrow を利用することで，分岐や反復の使い方やアルゴリズムの理解に時間を使うことができたといえる．一方，作成途中のプログラムには文法エラーにならない打ち間違いもあり，これを減らすことで，よりプログラミングの構造の理解に時間を使うことができると考えられる．

5. おわりに

本論文では，プログラミング学習支援環境の設計と評価について述べた．実現した機能は，Web ブラウザでのコーディング・実行，エラーの通知，入力補助，ログの収集である．これらの機能により，特別な準備を必要としないため，以前までの環境と同様導入が容易であるが，エラーの通知やインデントなどの入力補助機能から，これまでの環境と違い初学者の文法エラーへの対応を支援することができる．本環境の利用者のログから，文法エラーの割合が少なく，エラーからの復帰時間が短いことがわかり，学習者が自力でエラーを修正できていると考えられることから，本環境の学習支援はプログラミング教育に有用である．

今後の課題は，実行中に発生したエラーの場所の表示と，打ち間違いなどプログラミングの本質でないことに起因する論理エラーからの復帰の支援が挙げられる．

参考文献

1) 幼稚園、小学校、中学校、高等学校及び特別支援学校の学習指導要領等の改善及び必要な方策等について（答申）（中教審第 197 号）．
http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo0/toushin/1380731.htm
2) 土肥紳一, 今野紀子: Processing による高校生を対象としたプログラミング入門体験 2，情報教育シンポジウム 2014 論文集，Vol. 2014，No. 2，pp. 119-126(2014).
3) Mohammed Al-Bow, et al. : Using game creation for teaching computer programming to high school students and teachers, ACM SIGCSE Bull. Vol. 41, No. 3, pp. 104-108(2009).
4) Papancea, et al. : An open platform for managing short programming exercises. ICER '13 Proceedings of the ninth annual international ACM conference on International computing education research, pp. 47-52(2013).
5) 水越敏行, 村井純, 生田孝至ら: 情報の科学, 日本文教, (2012).
6) 赤堀侃司，永野和男，東原義訓ら: 情報の科学，東京書籍，(2012).
7) 加藤 利康 ，石川 孝: プログラミング演習支援システムにおける学習状況把握機能の提案，情報処理学会研究報告コンピュータと教育，Vol. 2013-CE-120，No. 2，pp. 1-8(2013).
8) 長島和平，長慎也，間辺広樹，並木美太郎，兼宗進: JSLesson - 高校生向け JavaScript 学習環境，情報処理学会研究報告コンピュータと教育，Vol. 2016-CE-134，No. 16，pp. 1-9(2016).
9) Shinya Cho, Susumu Kanemune: Tonyu System - Action Game Development System, The Journal of Game Amusement Society, Vol. 3, No. 1(2011).
10) 長島 和平，長慎也: Tonyu System 2 ゲーム制作を通じたプログラミング学習に適したフレームワーク，情報処理学会研究報告コンピュータと教育，Vol. 2015-CE-129，No. 2，pp. 1-8(2015).