一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
ET2022-1(2022-05)

ビジュアル型言語からテキスト型言語への移行のための中間コンテンツの提案と評価

梅澤　克之† 　石田　昂大†† 　中澤　　真††† 　平澤　茂一††††

† 湘南工科大学　〒 251–8511 神奈川県藤沢市辻堂西海岸 1–1–25
†† 株式会社アローズ・システムズ　〒 220–6007 神奈川県横浜市西区みなとみらい 2–3–1
††† 会津大学短期大学部　〒 965–0003 福島県会津若松市一箕町大字八幡門田 1–1
†††† 早稲田大学　〒 169–8555 東京都新宿区大久保 3–4–1
E-mail: †{umezawa,aaa}@info.shonan-it.ac.jp, ††nakazawa@jc.u-aizu.ac.jp, †††hira@waseda.jp

あらまし　近年，プログラミングの入門としてビジュアル型のプログラミング言語（以降，ビジュアル型言語と呼ぶ）が使われるようになってきている．その後は C 言語や Java 言語などのテキスト型プログラミング言語（以降，テキスト型言語と呼ぶ）に移行していくことになる．しかしシームレスな移行方法は確立されていない．本研究プロジェクトでは，ビジュアル型言語からテキスト型言語への移行の方法論を確立することを目的とする．本研究が確立されると，プログラミング言語の初学者がビジュアル型言語による学習から始めて，その後，シームレスかつ自発的にテキスト型言語の学習に遷移できるようになることが期待できる．本研究では，ビジュアル型言語の学習とテキスト型言語の学習の間に，我々が提案する中間コンテンツを用いた学習を挟む方がその後のテキスト型言語の理解度が高まるということを実証実験を通して示す．さらに，提案する中間コンテンツが，ビジュアル型言語とテキスト型言語の中間に位置づけられる特徴を有することをアンケートにより評価する．
キーワード　ビジュアル型プログラミング言語，テキスト型プログラミング言語，学習分析

A Proposal and Evaluation of Intermediate Content for Transition from Visual to Text-Based Languages

Katsuyuki UMEZAWA†, Kouta ISHIDA††, Makoto NAKAZAWA†††, and Shigeichi HIRASAWA††††

† Shonan Institute of Technology　Tsujido-Nishikaigan 1–1–25, Fujisawa, Kanagawa 251–8511, Japan
†† Arrows Systems Co. Ltd.　Nishi-ku Minato-Mirai 2–3–1, Yokohama, Kanagawa 220–6007, Japan
††† Junior College of Aizu　Monden 1-1, Yahata, Ikki-Machi, Aizuwakamatsu, Fukushima 965–0003, Japan
†††† Waseda University　Okubo 3–4–1, Shinjuku, Tokyo 169–8555, Japan
E-mail: †{umezawa,aaa}@info.shonan-it.ac.jp, ††nakazawa@jc.u-aizu.ac.jp, †††hira@waseda.jp

Abstract　Beginners in learning programming learn visual programming languages such as Scratch, while experts use text programming languages such as C and Java. However, a seamless transition from a visual programming language to a text programming language has not been established. Our research project aims to establish a methodology for the transition between the two types of languages. In this study, as part of our project, we will focus on the features that an intermediate language between a visual programming language and a text programming language should have. In this study, we demonstrate that learning with our proposed intermediate content between visual and text-based language learning enhances the subsequent comprehension of the text-based language. Furthermore, the proposed intermediate content will be evaluated by means of a questionnaire to ensure that it has characteristics that are intermediate between visual and text-based languages.
Key words　Visual-based Programming, Text-based Programming, Learning Analytics.

1. はじめに

近年，プログラミングの入門としてビジュアル型のプログラミング言語（以降，ビジュアル型言語と呼ぶ）が使われるようになってきている．その後は C 言語や Java 言語などのテキスト型プログラミング言語（以降，テキスト型言語と呼ぶ）に移行していくことになる．しかしシームレスな移行方法は確立されていない．

我々は，ビジュアル型言語からテキスト型言語への移行の方法論を確立することを目的とする研究プロジェクトを開始している．具体的には，ビジュアル型言語とテキスト型言語の学習の利点を有し，両者の差異を埋める教育コンテンツ（中間型言語とよぶ）を検討・試作し，実証実験を通して，学習結果だけでなく学習中の学習状態を評価する．これにより，その中間型言語の有効性を評価し，今後の初等・中等プログラミング教育に役立つ教育コンテンツ（中間型言語）を完成させる．ビジュアル型言語とテキスト型言語の溝を埋める想定の中間型言語の評価に関しては，今までは，学習後のアンケートや成績など，学習効果の結果のみに着目しており，理解できるようになったか否かを評価するのみであったが，これらの評価方法では，今回提案する中間型言語の効果を正確に測ることが出来ない．我々の研究プロジェクトでは，学習後の従来の評価方法に加えて，学習中の脳波や視線，表情などの生体情報および学習中の学習履歴を用いて学習状態を計測し，中間型言語がビジュアル型言語とテキスト型言語の中間的な役割を果たし，スムーズな移行に貢献しているかを分析・評価する．本研究が確立されると，プログラミング言語の初学者がビジュアル型言語による学習から始めて，その後，シームレスかつ自発的にテキスト型言語の学習に遷移できるようになることが期待できる．

本研究では，上述の研究プロジェクトの一部として，まず中間コンテンツの提案に着目し，ビジュアル型言語の学習とテキスト型言語の学習の間に，我々が提案する中間コンテンツを用いた学習を挟む方がその後のテキスト型言語の理解度が高まるということを実証実験を通して示す．さらに，提案する中間コンテンツが，ビジュアル型言語とテキスト型言語の中間に位置づけられる特徴を有することをアンケートにより評価する．

2. 従来研究

2.1 ビジュアル型言語ついて

ビジュアル型言語は，ブロックベースの命令型言語，またはフローベースの関数型言語の 2 つの大きなカテゴリに分類される．

Mason et al. [1] は，可能な限り類似するように設計された 2 つの環境のいずれかでいくつかの単純な問題をプログラムするような数百人規模の実験を行い，2 つのカテゴリの相対的なメリットを評価する実証研究を行った．

Robinson et al. [2] は，ブロック型言語（Scratch）からテキスト型言語への移行について研究を行った． Scratch の魅力的な機能の 1 つは，学習者が構文エラーを回避して簡単に理解して使用できることである．Scratch はプログラミングロジックの学習に重点を置いており，学習者は論理的に考える方法を学ぶが，プログラムの作成方法を学べるものではないと述べている．

2.2 ビジュアル型言語とテキスト型言語の比較研究

Mladenović et al. [3] は，プログラミングの基本概念の 1 つであるループについて，学生の誤解に関する調査を 207 人の小学生を対象に実施した．生徒たちは，ブロック型言語（Scratch）とテキスト型言語（Logo および Python）の 3 つのプログラミング言語を学んだ．ブロック型言語では学生のループに関する誤解が最小限に抑えられることがわかった．ネストされたループなどタスクが複雑になるにつれて，違いがより明確になった．この研究では，構文の問題が解消されるためプログラミングの初学者向けにはビジュアル型言語を使用する正当性を主張している．ただしこの研究では両言語のギャップを埋める事には言及していない．

Navarro-Prieto et al. [4] は，1 つの実験を行って，ビジュアル型言語がテキスト型言語よりも理解しやすい理由を画像処理の心理学の観点で説明している．画像処理は意味情報へのより迅速なアクセスを容易にするため，ビジュアル型言語は，テキスト型言語よりもデータフロー関係を迅速に心的表現を構築できるという仮説を検証した．テキスト型言語のプログラマーは最初に制御フローの心的表現にアクセスし，次にデータフローの心的表現にアクセスすることが分かった．

Xu et al. [5] は，既存の学術データベースを調査するという方法で，ブロック型言語のプログラミング環境とテキスト型言語のプログラミング環境が，学生の認知的と感情的な学習成果の両方に及ぼす全体的な影響を調べた．初心者プログラマーのためのブロック型言語の有用性（utility）と有効性 (efficacy) の統計的優位性を示すことはできなかったが，ハイブリッド型の更なる調査の必要性を述べている．

2.3 ハイブリッド型言語の提案

Daskalov et al. [6] は，初学者向けのテキスト型言語とビジュアル型言語のハイブリッド型言語を使用するための環境を提案している．ブロック型ではなくフロー型のビジュアル言語とテキスト型言語のハイブリッド環境であり，初学者のプログラミングトレーニングに適していると主張している．

Weintrop [7] は，テキスト型言語とビジュアル型言語とハイブリッド型言語の比較を行った．結論として，ハイブリッド型言語が両方の祖先の特徴を示している一方で，特定の領域でブロック型言語とテキスト型言語よりも優れていることを明らかにした．

Weintrop [8] は，PencilCode を使用して，ブロック型言語，ハイブリッド型言語，テキスト型言語を使用して移行のスキルの影響を比較した．3 つグループの学習者がそれぞれの言語で 5 週間の学習をして第 6 週目以降はすべてグループが Java を使用したテキストベースに移行した．ブロック型言語の学習者の方がテキスト型言語の学習者よりもループや変数を理解する姿勢が優れていた．ブロック型言語の欠点は，大規模で複雑なプログラムを構築するのが難しいことが分かった．また，ハイブリッド型は，テキスト型とブロック型を切り替えて使う必要

があったため，公正な比較が出来なかった．

Alrubaye et al. [9] は，ブロックのみ表示にする，テキストのみの表示にもできる，ブロックの領域とテキストの領域を並べて表示できる，というようなツールを作成して移行を評価した．ブロックベースからテキストベースの学習アプローチと比較した場合，平均してハイブリッドベースのアプローチは，プログラミングの基礎，暗記，移行の容易さに関する学生の理解を 30

2.4 ビジュアル型言語とテキスト型言語間のギャップに関する研究

Tóth et al. [10] は，ビジュアル型言語とテキスト型言語間のギャップの存在を指摘した．この研究ではビジュアル型言語（MIT App Inventor 2）から Java Bridge Code Generator を知識伝達のメディエーターとして使用して，テキスト型言語（Android Studio）に移行する方法を検証した．この論文では Java Bridge Code Generator が，ビジュアル型言語ととテキスト型言語の間のギャップを埋めるのに役立ったと主張している．

Weintrop et al. [11] はビジュアル型言語から入ったが学習者とテキスト型言語から入った学習者がテキスト型言語に移行した後の知識移転の変化を実験で評価し，両者に有意差が無い事を示している．移行ではなくテキスト型言語のスキルを修得後の比較をおこなった．

また，我々は，脳波計を用いて学習者の学習状況を把握する方法の研究を推進中である．我々は，課題遂行中の脳波情報を取得し，難しい課題に取り組むと β/α の値が高くなることを示した [12] [13]．この研究の過程で，ビジュアル型言語 (Scratch) の課題に取り組んでいる際の脳波とテキスト型言語 (C 言語) に取り組んでいるときの脳波に差異があることを確認した．具体的には，ビジュアル型言語に関しては課題の難易度が高くなっても β/α の値が高くなることはなかった．これによりビジュアル型言語とテキスト型言語の学習過程には異なる思考が行われている可能性が示唆された [14].

3. 本研究プロジェクトの概要

本研究プロジェクト全体の目標としては，ビジュアル型言語とテキスト型言語の学習の利点を有し，両者の差異を埋める教育コンテンツ（中間型言語とよぶ）を検討・試作し，実証実験を通して，学習結果だけでなく学習中の学習状態を評価する．これにより，その中間型言語の有効性を評価し，今後の初等・中等プログラミング教育に役立つ教育コンテンツ（中間型言語）を完成させる．まず，ビジュアル型言語とテキスト型言語の溝を埋める想定の中間型言語の検討・施策し，それを用いて実証実験を行う．評価に関しては，今までは，学習後のアンケートや成績など，学習効果の結果のみに着目しており，理解できるようになったか否かを評価するのみであった．これらの評価方法では，今回提案する中間型言語の効果を正確に測ることが出来ない．本研究では，学習後の従来の評価方法に加えて，学習中の脳波や視線，表情などの生体情報および学習中の学習履歴を用いて学習状態を計測し，中間型言語がビジュアル型言語とテキスト型言語の中間的な役割を果たし，スムーズな移行に貢献しているかを分析・評価する．本研究の全体概要を図 1 に示す．

3.1 プログラミング間の差異を埋めるプログラミング学習コンテンツの開発

現状のビジュアル型言語は見た目が楽しい，すぐ実行して動作する，文法エラーが存在しないなどの特徴がある．これに対してテキスト型言語は，文字ばかりであり，1 文字でも間違えれば動作せず，グラフィカルなことをやろうとするととても手間がかかる，というような特徴がある．提案する中間型言語は，シンプル（テキスト型言語特有の追加知識不要），素早いフィードバック（実行結果がすぐわかる），文法エラーが起こりにくく，論理エラーの箇所がわかり易い，というような特徴を持つものを試作する．

3.2 学習時の生体情報（脳波や視線）を取得する実証実験

3.1 で作成した中間型言語，およびビジュアル型言語，テキスト型言語のそれぞれを用いて，学習時に脳波や視線等を取得する実証実験を実施する．具体的には，湘南工科大学情報工学科および会津大学短期大学部産業情報学科にて，基礎プログラミング実習等の授業，あるいは付属高校技術コースの生徒，研究室大学生を対象に，Scratch，プログラミン，Blockly 等のビジュアル型言語と Java，JavaScript, C 言語等のテキスト型言語を用いた実証実験を実施する．なお，実験参加者に関しては経験者と初学者の両方を実験対象とすることで，より効果的に各言語の差異を明らかにできると考える．

3.3 各言語学習時の差異の分析・抽出し，学習プロセスの溝の明確化

上記 3.2 の実験結果を用いて各言語の学習時の学習状態の差異を分析する．学習時の編集履歴と脳波を計測することによって，学習者がどのような課題でつまずいているのか，あるいは逆にどのようなときに学習意欲が高まっているのか等を明らかにした上で，学習者の編集プロセスとその際の脳波によって学習者を分類（クラスタリング）し，比較・分析を行っていく．

4. 中間コンテンツの提案

4.1 概　　要

現状のビジュアル型言語は見た目が楽しい，すぐ実行して動作する，文法エラーが存在しないなどの特徴がある．これに対してテキスト型言語は，文字ばかりであり，1 文字でも間違えれば動作せず，グラフィカルなことをやろうとするととても手間がかかる，というような特徴がある．提案する中間コンテンツは，シンプル（テキスト型言語特有の追加知識不要），素早いフィードバック（実行結果がすぐわかる），文法エラーが起こりにくく，論理エラーの箇所がわかり易い，というような特徴を持つ必要があると考えた．

4.2 中間コンテンツの実現

前述の特徴を持ったコンテンツとして， JSFiddle [15] を活用して JavaScript で音楽を作るという学習コンテンツを作成した．図 2 に示すように，JSFiddle は Web ブラウザのみを用いて JavaScript のコーディングから実行結果の表示までを 1 つの画面で行える Web 版の統合環境である．この JSFiddle に音を出すためのライブラリ（Beeplay）を追加することで簡単に音楽を作れる．例えば “ドレミ” と音を出すには図 3 のように

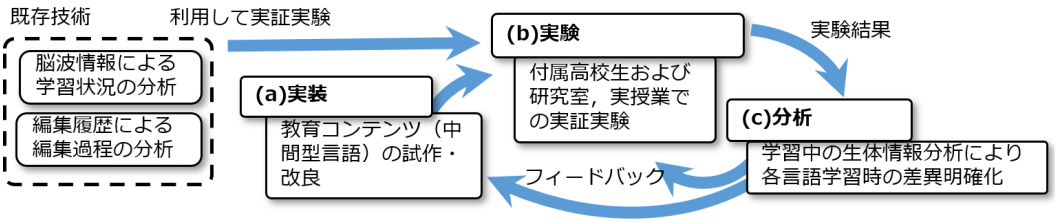

図 1　本研究プロジェクトの全体概要

コーディングすれば良い．

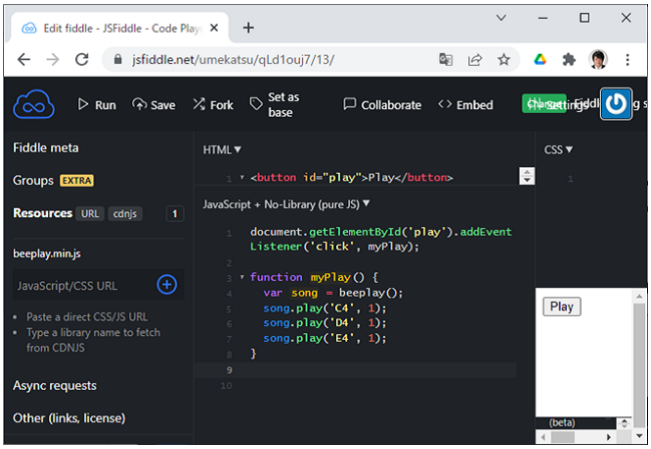

図 2　JSFiddle の画面

```
function myPlay() {
  var song1 = beeplay();
  song1.play('C4', 1);
  song1.play('D4', 1);
  song1.play('E4', 1);
}
```

図 3　ドレミの音を出すプログラム

上記，音を鳴らすプログラムは，play というメソッドのみで実現できるため，多くの単語（予約語）や文法を覚える必要がなく，実行したらすぐ音が鳴り，結果が素早く確認できる．さらに，期待した音と違う音がなってしまうような論理エラー（プログラムは間違えていないのでコンパイラはエラーを出力しない）の場所の音を聞いて容易に判断できる．また，音楽は繰り返し構造があり，また，2 回目の繰り返しの時には異なる音を鳴らす，というようなこともある．これらは繰り返し（for 文）や条件分岐（if 文）としてプログラムで表せる．さらに小節毎に繰り返すような構造もあり，小節を関数（function）として定義することもできる．このように音楽とプログラミングはとても相性が良いと思われる．また，なんといっても自分の好きな音楽が徐々に出来上がっていくのはとても楽しい．

5.　実験で使用するツール

本章では実験で使用するビジュアル型言語や中間コンテンツなどについて説明をする．

5.1　Google Blockly

Google Blockly [16] とは Google が提供するビジュアル型プログラミングを作成できるライブラリである．Blockly Games とは Google Blockly を利用して作成された Web 上で動くビジュアル型プログラミングが学習できるコンテンツである．Blockly　Games のプログラミング言語としての特徴として，実行結果が見た目ですぐわかる，音を実行結果として聞くことができる，エラーが分かり易いなどが挙げられる．メニュー画面上でパズル，迷路，音楽などの各種コンテンツを選択できる．

5.1.1　Blockly Games パズル

図 4 に Blockly　Game のパズルの画面を示す．このパズルではビジュアル型プログラミングに慣れてもらうことが目的である．操作は簡単でマウスでブロックをドラック＆ドロップで操作が可能である．例えば「猫」と書かれたブロックに猫の画像を繋げ，猫の特徴が書かれたブロックを繋げることでパズルを完成させる．足の数をプルダウンから選択し回答を押すことで答え合わせができる．

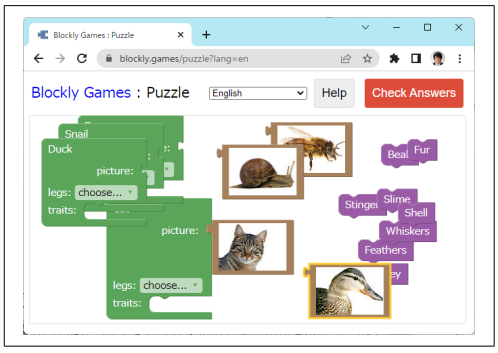

図 4　Blockly Games（パズル）の画面

5.1.2　Blockly Games 迷路

図 5 に Blockly　Game の迷路の画面を示す．このコンテンツは人型のキャラクターを黄色の線の上を進んで行くものである．右側のフィールドに「まっすぐ進む」などのブロックをドラック＆ドロップすることでゲームが進行していく．この迷路では簡単な繰り返し処理と条件分岐を学習することができると考えた．

5.1.3　Blockly Games 音楽

図 6 に Blockly Game の音楽の画面を示す．このコンテンツは音符ブロックを右側のフィールドにドラック＆ドロップし開始ブロックに繋げることで音がなるようになる．左側にある楽譜に合うように音符ブロックを適切に変更し，繋げていくことでゲームが進行していく．このコンテンツは簡単な関数に関し

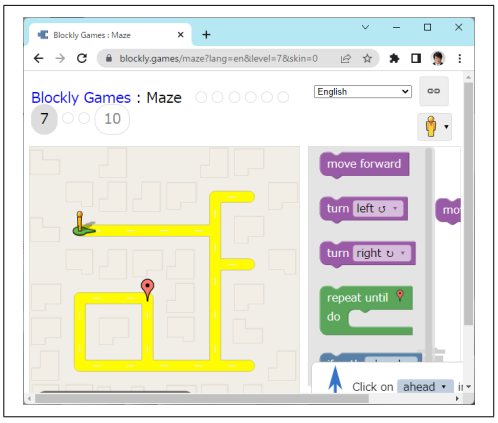

図 5　Blockly Games（迷路）の画面

て学習することができる．

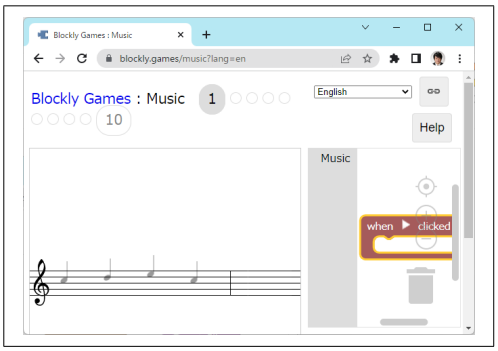

図 6　Blockly Games（音楽）の画面

5.2　JSFiddle

4.2 節で述べたように，JSFiddle は、Web ブラウザのみを用いて JavaScript のコーディングから実行結果の表示までを行える Web 版の統合環境である．

5.3　Python

Python はテキスト型プログラミング言語の一種である．C 言語や java 言語と違い比較的簡単な記述でプログラムを作成できる．そのため今回の実験のテキスト型言語として使用した．このコンテンツの目的は前述の Blockly Games，JSFiddle の学習効果を確認するためのものである．今回の Python は，Google Colaboratory [17] を利用した．Google Colaboratory は，ブラウザ上で Python を記述，実行できる統合環境である．Google Colaboratory 上の Python の問題の一部を図 7 に示す．

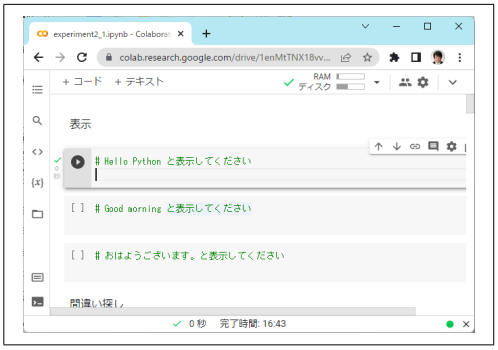

図 7　Google Colaboratory 上の Python の画面

6.　実 験 方 法

6.1　実験の概要

今回の実験ではプログラミング学習をする上で中間コンテンツの効果があるかを調べるため参加者を二つのグループに分け実験を行った．グループ A は中間コンテンツを挟んで学習をする実験を行った．また，グループ B は中間コンテンツを挟まずに学習し，その後グループ A と比較をするため中間コンテンツを挟んで学習する実験を行った．実験の流れを図 8 に示す．グループ A には 9 名，グループ B には 12 名が参加した．なお，Blockly Games の実験では，グループ A はパズル，迷路，音楽の問題を行った．それに対してグループ B は，1 回目の Blockly Games の実験でパズル，迷路を行い，2 回目の Blockly Games の実験で音楽を行った．それぞれの課題の詳細は次節以降に示す．

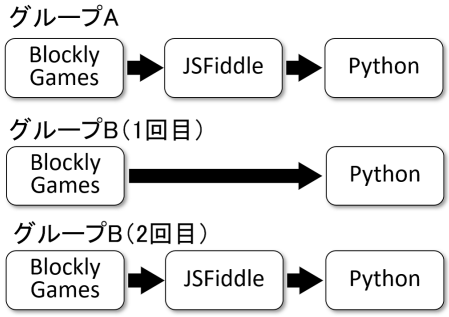

図 8　実験の流れ

6.2　ビジュアル型言語（Blockly Games）を利用した学習

ビジュアル型言語を利用した学習では 5.1 節で示した Blockly Games を利用する．パズルに関しては 4 問中 4 問まで，迷路に関しては 10 問中 10 問まで，音楽に関しては 10 問中 9 問まで回答してもらうこととする．音楽を 9 問までしかやらないのは 10 問目が自由に音楽を作成する問題のため割愛する．この実験では，Blockly Games のシステム j から使い方をレクチャーしてくれるため，実験者から操作方法を説明することはしない．使い方が分からない場合やわからない問題が出てきたときは実験者が説明を行った．

6.3　中間コンテンツ（JSFiddle）を利用した学習

中間コンテンツを利用した学習では，5.2 節で示した JSFiddle を利用する．この実験はグループ A，グループ B 共に同じように学習してもらう．ここでは，この実験では，C メジャースケールを鳴らす問題，for 文を使って音楽を繰り返す問題，if 文を使って条件分岐して音楽を鳴らす問題，for 文の中に if 文を組み込む問題 (2 問)，前節の Blockly Games で回答した音楽を JSFiddle で再現する問題の計 6 問に回答してもらう．ここでは，最初に実験者から JSFiddle の使い方やプログラムの記述の仕方をレクチャーする．その後問題に取り組んでもらい，わからない問題が出てきたときは参考資料を見てもらうか直接聞いてもらうこととした．

6.4　テキスト型言語（Python）を利用した学習

テキスト型言語を利用した学習では，5.3 節で示した Python

を利用する．この実験では，問題数は 30 問で制限時間は 20 分間である．ここでは基本的なプログラムについて回答してもらう．具体的には print 文や変数の基本的な使い方，for 文，if 文，for 文の中に if 文を入れ込む，などの問題に回答してもらう．またグループ B では一回目と二回目では難易度は同じ程度で別の問題を用意した．グループ B の二回目はグループ A と同じ問題である．わからない問題が出てきた時は参考資料を見てもらうか，後回しにしてもらい時間があればやってもらうこととした．正答数で評価を行う．

7. テキスト型言語の正答数の評価

7.1 実験結果

テキスト型言語 Python を利用した学習で 30 問ある問題を 20 分間で回答してもらった．問題の正答数を表 1 に示す．なお，参加者 18 に関しては正確な回答データを採取できなかったため実験失敗として集計していない．

表 1　テキスト型言語 (Python) の正答数

グループ A	正答数	グループ B	一回目正答数	二回目正答数
参加者 1	25	参加者 10	26	30
参加者 2	25	参加者 11	24	30
参加者 3	26	参加者 12	24	28
参加者 4	28	参加者 13	25	29
参加者 5	25	参加者 14	29	29
参加者 6	26	参加者 15	22	29
参加者 7	22	参加者 16	14	16
参加者 8	28	参加者 17	21	24
参加者 9	27	参加者 18	-	-
		参加者 19	21	29
		参加者 20	20	24
		参加者 21	21	20
平均値	25.78	平均値	22.45	26.18

中間コンテンツを挟んだグループ A の正答数と中間コンテンツを挟まないグループ B の一回目の正答数を比較するとグループ A の方が高い．またグループ B 内でも中間コンテンツを挟まない一回目と中間コンテンツを挟んだ二回目の正答数を比較すると二回目の方が高い．つまり中間コンテンツを挟んだほうが，その後のテキスト型言語の理解度が高くなったと言えそうである．またグループ A の正答数とグループ B の二回目の正答数を比較すると，同じような点数になっている．つまり，これによりグループ B の理解度が低いわけではないとも言えそうである．

7.2 実験結果の分析

上記を検証するために検定を行う．まず，グループ A とグループ B のテキスト型言語 Python の平均正答数に差があるか否かを検定する．まず，グループ A とグループ B の正答数の分散に差があるかを調べるため F 検定を行った．有意水準 5% で F 検定を行ったところグループ A とグループ B の一回目では，$p = 0.024$，またグループ A とグループ B の二回目では，$p = 0.008$ となり，ともに p 値 < 0.1 より等分散ではないことが分かった．この結果を受けて，分散が等しくないと仮定した 2 標本による t 検定行った．グループ A とグループ B の一回目の平均正答数の有意水準 5%の片側 p 値は，$p = 0.012(< 0.05)$ となった．これによりグループ A とグループ B の一回目の正答数の平均値に差があるといえる．また，グループ A とグループ B の二回目の平均正答数の有意水準 5%で両側 p 値は，$p = 0.796(> 0.05)$ となった．これにより，グループ A とグループ B の二回目の正答数の平均値に差があるとは言えない，という結果になった．これらの結果により，前述の仮定（中間コンテンツを挟んだほうがテキスト型言語の理解度が上がる．またグループ B の理解度が低いわけではない．）が確認できた．さらに，グループ B の一回目と二回目の問題の平均正答数の差が統計的に有意かを確かめるため有意水準 5%で両側検定の t 検定を行った．参加者は同一人物なので対応のある t 検定を行ったところ片側 p 値は，$p = 0.001(< 0.05)$ となり，グループ B の一回目と二回目の平均値には差があると言える．

8. アンケートによる評価

提案の中間コンテンツの特徴を確認するために，実験終了後にアンケートを実施した．

8.1 アンケート項目

アンケートは下記の設問とした．回答は全て 4 段階とした．

設問 1:　JsFiddle での音楽作成は楽しかったですか

回答 1:　楽しかった

回答 2:　少し楽しかった

回答 3:　あまり楽しくなかった

回答 4:　楽しくなかった

設問 2:　：JsFiddle での音を鳴らすプログラムは，覚えなければならない英単語 (play など) は多いと思いますか，少ないと思いますか

回答 1:　とても少ない

回答 2:　少ない

回答 3:　多い

回答 4:　とても多い

設問 3:　JsFiddle での音を鳴らすプログラムは，結果がすぐわかりますか（実行後すぐ音を確認できるなど）

回答 1:　実行結果がすぐわかる

回答 2:　やや実行結果がすぐわかる

回答 3:　やや実行結果がすぐわからない

回答 4:　実行結果がすぐわからない

設問 4:　JsFiddle での音を鳴らすプログラムは，間違えている箇所がわかり易いですか（音がずれている箇所が耳で聞いてすぐわかるなど）

回答 1:　間違えている箇所がわかり易い

回答 2:　やや間違えている箇所がわかり易い

回答 3:　やや間違えている箇所がわかりにくい

回答 4:　間違えている箇所がわかりにくい

8.2 アンケート結果

実験終了後に，中間コンテンツの特徴に関するアンケートを実施した．結果を表 2 に示す．表 2 に示したように，今回実施

した JavaScript で音楽を作るプログラミング授業は，ビジュアル型言語とテキスト型言語の中間に位置づけられる特徴を有することがアンケート結果より明らかになった．

表 2　アンケート結果

設問番号	平均値	最頻値	解釈
設問 1	2.30	2	少し楽しかった
設問 2	2.25	2	覚えるべき英単語は「少ない」と「多い」のちょうど中間あたり
設問 3	1.70	1	実行結果が「すぐわかる」と「ややすぐわかる」の中間あたり
設問 4	2.35	1,2	やや間違えている箇所がわかり易い

9. まとめと今後の課題

本研究では，ビジュアル型言語の学習とテキスト型言語の学習の間に，我々が提案する中間コンテンツを用いた学習を挟む方がその後のテキスト型言語の理解度が高まるということを実証実験を通して示した．さらに，提案する中間コンテンツが，ビジュアル型言語とテキスト型言語の中間に位置づけられる特徴を有することをアンケートにより評価した．

今後の課題としては，我々の研究プロジェクトの全体目標の達成に向けて，参加者を増やしての実験および分析、学習結果の分析に加えて、学習中の脳波などの生体情報を用いた学習状態の分析、実授業への適用などを行っていく予定である．

研究倫理について

今回の実験は湘南工科大学研究倫理委員会の承認を得ている．また実験参加者と実験参加者の保護者から実験参加に関する署名を得ている．

謝　　辞

本研究は JSPS 科研費 JP21K18535，JP20K03082，JP19H01721 の助成を受けたものです．また，本研究の一部は，早稲田理工研特別助定 1010000175806 NTT 包括協定共同研究，および，経営情報学会「ICT と教育」研究部会の助成を受けたものです．本研究成果の一部は早稲田大学理工総研プロジェクト研究「次世代 e-learning に関する研究」の一環として行われたものです．

文　　献

[1] D. Mason and K. Dave, “Block-based versus flow-based programming for naive programmers,” 2017 IEEE Blocks and Beyond Workshop (B&B), 2017, pp. 25–28, doi: 10.1109/BLOCKS.2017.8120405.

[2] William Robinson, “From scratch to patch: Easing the blockstext transition,” In Proceedings of the 11th Workshop in Primary and Secondary Computing Education, pp. 96–99. ACM, 2016.

[3] Monika Mladenović, Ivica Boljat, Žana Žanko “Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level,” Education and Information Technologies 23(4), pp. 1483–1500, 2018.

[4] Raquel Navarro-Prieto, Jose J. Cañas, “Are visual programming languages better? The role of imagery in program comprehension,” International Journal of Human-Computer Studies, Volume 54, Issue 6, pp. 799–829, June 2001.

[5] Zhen Xu,Albert D. Ritzhaupt, Fengchun Tian, and Karthikeyan Umapathy, “Block-based versus text-based programming environments on novice student learning outcomes: A meta-analysis study,” Computer Science Education, pp. 177–204, Jan 2019.

[6] Rumen Daskalov, George Pashev, Silvia Gaftandzhieva, “Hybrid Visual Programming Language Environment for Programming Training,” TEM Journal. Volume 10, Issue 2, pp. 981–986, May 2021.

[7] David Weintrop, Uri Wilensky, “Between a Block and a Typeface: Designing and Evaluating Hybrid Programming Environments,” IDC ’17: Proceedings of the 2017 Conference on Interaction Design and Children, pp. 183-192, June 2017.

[8] David Weintrop, “Blocks, text, and the space between: The role of representations in novice programming environments,” In 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 301–302. 2015.

[9] Hussein Alrubaye, Stephanie Ludi, Mohamed Wiem Mkaouer, “Comparison of block-based and hybrid-based environments in transferring programming skills to text-based environments,” Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering, pp. 100-109, 2019.

[10] Tomáš Tóth, Gabriela Lovászová, “Mediation of Knowledge Transfer in the Transition from Visual to Textual Programming,” Informatics in Education, DOI 10.15388/infedu. 2021.

[11] David Weintrop, Uri Wilensky, “Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms,” Computers & Education, Volume 142, 103646, 2019,

[12] K.Umezawa, T.Saito, T.Ishida, M.Nakazawa, and S.Hirasawa, “Learning state estimation method by browsing history and brain waves during programming language learning,” Proceeding of the 6th World Conference on Information Systems and Technologies (World CIST 2018)*, p.p. 1307-1316, Mar. 2018.

[13] K.Umezawa, T.Saito, T.Ishida, M.Nakazawa, and S.Hirasawa, “Learning-state-estimation Method using Browsing History and Electroencephalogram in E-learning of Programming Language and Its Evaluation,” Proceeding of the International Workshop on Higher Education Learning Methodologies and Technologies Online (HELMeTO 2020), pp.22-25, Sept. 2020.

[14] K. Umezawa, M. Nakazawa, M. Kobayashi, Y. Ishii, M. Nakano, and S. Hirasawa, “Comparison Experiment of Learning State Between Visual Programming Language and Text Programming Language,” Proceeding of the IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE2021), p.p. 1-5, Dec. 2021.

[15] JSFiddle: https://jsfiddle.net/

[16] https://blockly.games/

[17] https://colab.research.google.com/notebooks/