Scratch 課題におけるブロック使用傾向とテキストベースプログラミング言語習得の関連性の考察

Analysis of the Relationship between Block Usage Patterns in Scratch and the Learning of Text-Based Programming Languages

石郷 祐介
ISHIGO Yusuke

要旨：本学 情報メディア学科では，１年次前期の必修講義「プログラミング入門」でビジュアルプログラミング言語「Scratch」を用いてプログラミングの基本概念を習得する．その後，１年次後期の必修講義「プログラム演習Ⅰ」にて，テキストベースプログラミング言語「C 言語」の習得を目指している．これまで，「Scratch」からテキストベースプログラミングへの移行に際して，「Scratch」課題におけるブロック数を指標に指導を行ってきた．本研究では，ブロック数に加え，どの種類のブロックの使用が，移行学習の際に影響を与える要因となるかを Random Forest 手法を使って分析した．

Abstract: In the Department of Information and Media Studies, first-year students learn basic programming concepts using the visual programming language “Scratch” in the required course “Programming Introduction” during the first semester. In the second semester, they learn the text-based programming language “C” in the required course “Programming Exercise I.” I have been supporting the transition from Scratch to text-based programming, using the number of blocks used in Scratch works as an indicator. In this study, we applied the Random Forest method to analyze the number of blocks and the types of blocks used in Scratch works, in order to identify which factors influence learning during the transition to text-based programming.

キーワード：プログラミング学習 , ビジュアルプログラミング , 移行学習 , 概念的変換 , 教育工学
Key Words：Programming Education, Visual Programming, Transitional Learning, Conceptual Transfer, Educational Technology

１．はじめに

筆者は，情報メディア学科の１年次前期の必修講義「プログラミング入門」を担当している．本講義は，プログラミング未経験の学生を対象として，主に手続き型プログラミング言語を通じたプログラミング的思考の習得を目的として開講している．ここでいうプログラミング的思考とは，文部科学省が「自分が意図する一連の活動を実現するために，どのような動きの組合せが必要であり，一つ一つの動きに対応した記号を，どのように組み合わせたらいいのか，記号の組合せをどのように改善していけば，より意図した活動に近づくのか，といったことを論理的に考えていく力」と定義しているものである[1]．

講義では，マウスによるドラッグアンドドロップ操作でプログラムを構築できるビジュアルプログラミング言語「Scratch」[2]を用いている．前半では，Scratch の基本的な「ブロック（手続き型プログラミング言語における「関数」に相当する）」の使い方を，小規模な課題を通じて理解する．後半では，前半で学んだ「ブロック」の使い方を応用し，学生が各自選んだ既存のゲームの動きを Scratch で再現する課題に取り組む．この課題は，太田ら（2016）が分類する Scratch におけるアクティブラーニング形態の中で，メイキング型の学習にカテゴライズされる[3]．さらに，１～２年次に開講される「プログラム演習Ⅰ・Ⅱ」では，テキストベースプログラミング言語の C 言語を習得する．そのため，「プログラミン

（2025年10月20日受付，2025年12月22日受理）

グ入門」は，ビジュアルプログラミングからテキストベースプログラミングへの橋渡しとしての役割も担っている．

本稿では，「プログラミング入門」において実施したScratch課題と，最終課題として行うテキストベースプログラミング課題の成績データを用い，ビジュアルプログラミングにおけるどの学習要素がテキストベースプログラミングの理解に関連しているのかを明らかにすることを目的する．

Scratch課題からは，総ブロック数（及びカテゴリーごとのブロック数），スプライト数，変数，ネスト構造の深さ等，プログラム構造を表す複数の特徴量を抽出した．また，テキストベース課題は，コンピュテーショナル・シンキング概念に基づく評価ルーブリックにとり，学習者の理解度を定量的に得点化した．これらのデータをもとに，Random Forest手法を用いて，各要素間の寄与率を算出した．本分析を通して，今後のScratch課題の設計及び課題指導の改善に資する知見を得ることを目指す．

2．関連研究

メイキング型の課題指導においては，受講者がそれぞれ異なるプログラムを開発するため，指導者は各プログラムを確認し，学習者の習熟度や達成度を把握する必要がある．

Scratchにおける習熟度評価ルーブリックの確立を目的とした研究として，Moreno-León らは，ブロックの種類ごとの使用を点数化し，学習者に自動的なフィードバックを行うシステム「Dr.Scratch」を開発・公開している[4]．

さらに，太田ら（2019）は，コンピュテーショナル・シンキング概念（Computational Thinking Concepts）に基づき，各能力を段階的に区別した新たな評価基準（CTC評価項目）を作成した．小学４～６年生が作成したScratchのプログラム871本を対象に分析を行い，この評価基準に基づきプログラムを点数化することで，学習者の習熟度を客観的に評価できることを示した．

本研究では，Scratch課題のデータ解析において，筆者が作成した解析プログラムを用いている一方，テキストベースプログラミングの習熟度測定には，太田らのCTC評価項目を基盤とした評価手法を採用している．

また，市川（2020）は，テキストベースプログラミング言語の習得を目的としたScratchによるプログラミング導入教育の実践と評価を行っている[6]．専門学校において，Scratchを用いたオリジナルゲーム制作課題を実施し，プログラム中の総ブロック数とテキストベースプログラミングに関するペーパーテストの得点との関係を分析した．その結果，総ブロック数とペーパーテスト得点との間に正の相関があることを示し，さらに，1000個以上のブロックを使用したScratch作品制作が，テキストベースプログラミングの習熟度向上に寄与する傾向があると報告している．

講義「プログラミング入門」後半のScratchプログラム課題では，この指標を参考に，学生に対して1000個以上のブロックを用いた作品制作を推奨している．しかし，市川の研究ではブロックの種類ごとの使用状況が考慮されていないため，学習者が不必要なブロックを過剰に追加した場合や，プログラム概念の獲得に直接結びつかないブロックの数も総ブロック数に含まれている可能性がある．

そこで，本研究では，使用ブロックの種類に着目した詳細な分析を行い，どのブロックの使用がテキストベースプログラミング言語への移行に影響を与えているのかを明らかにする．

3．研究方法

3.1　Scratchプログラムの解析

講義「プログラミング入門」では，Scratchの基本的なブロックの使用方法を学習した後，2025年５月～６月の約１ヶ月間にわたり，次のような課題を受講生（140名）に課し，その成果物として制作されたプログラムの定量的な解析を行った．

課題テーマ：「好きな2Dゲームの動きをScratchで再現する」

条件：

・１ステージのみや戦闘部分のみでも可とする
・インターネット上のチュートリアルや他者の作品をリミックスとして使用することは禁止する
・パズルゲームは難易度が高いため選択対象外とする

Scratchは，プログラム内容を「.sb3」形式で出力する．このsb3ファイルには，Scratch内で使用したスプライトの画像情報と，各スプライトに配置されたブロックの構造情報等が含まれる．本研究では，このsb3ファイル内に含まれるproject.jsonを解析するために，Pythonを用いて独自の解析プログラムを作成した[7]．

この解析プログラムを用いて，受講生が制作したScratchプログラムから，以下の情報を抽出した．

（１）　総ブロック数

（2） 変数の数
（3） スプライト数
（4） 種類ごとの使用ブロック数（動き・見た目・音・イベント・制御・調べる・演算・変数）
（5） 最大ネスト数
（6） 平均ネスト数

（1）の総ブロック数については，実際にプログラム実行時に動作する有効ブロックのみをカウント対象とした．ここで，非有効ブロックとは，先頭に「イベント」カテゴリーのブロックまたは「メッセージ受信」ブロックが接続されていないブロック郡と定義した．

（4）のブロックの分類は，Scratch 公式のカテゴリー区分に準拠した．また,ユーザ定義ブロックについては，講義内で扱っていないため，受講生の一部が自主的に学習し使用している例が見られたものの，本研究の解析対象からは除外した．

抽出した要素の基本統計量を，表1に示す．

表1　Scratch プログラムから抽出した各要素

	最大値	最小値	中央値	標準偏差
総ブロック数	8228	45	534	1208
変数の使用ブロック数	73	1	7	14
スプライト数	138	2	18	25
動きブロック数	1722	0	41.5	237
見た目ブロック数	1731	0	121	282
音ブロック数	1731	0	3	77
イベントブロック数	894	6	80	153
制御ブロック数	2780	5	129	341
調べるブロック数	1633	0	12	159
演算ブロック数	1808	0	34	229
変数の定義数	400	0	15	75
ネストの深さ(最大)	69	1	4	6
ネスト数の深さ(平均)	3.95	1	1.5	1

3.2　テキストベースプログラミング課題の評価方法

Scratch による課題の後，受講生は JavaScript を用いた Web ベースのクリエイティブコーディング環境である「p5.js」を通して，テキストベースプログラミングを学習し，ビジュアルプログラミングからテキストベースプログラミングへの移行を進めた．学習内容としては，Scratch で学んだブロックとの比較を通じて，構造化定理（順次・分岐・反復）を中心とした JavaScript の構文を学習した．その後，p5.js を用いた下記の５つの課題を出題した（表2）．

表2　テキストベースプログラミング確認課題

課題1	複数の円が重なった図形を p5.js を使って描いてください（参考画像あり）.
課題2	四角２つと円２つを使って，ロボットの顔を p5.js を使って描いてください（参考画像あり）.
課題3	斜めに動く四角を p5.js を使って描いてください（参考動画あり）.
課題4	動画のように縦に動いて画面下で跳ね返る四角を p5.js を使って描いてください.
課題5	課題４の画面下で跳ね返る四角を応用して上下左右に跳ね返る円を p5.js を使って描いてください.

それぞれ順次処理（課題1，課題2），無限ループ（課題3），分岐（課題4，課題5）を確認する趣旨として作成した．

受講生がビジュアルプログラミングからテキストベースプログラミングへの概念的変換（conceptual transfer）を，どの程度達成しているかを分析した．分析に際しては，太田ら（2019）が提案した CTC 評価項目を参考にした．ただし，CTC 評価項目は，Scratch プログラムを対象として設計されているため，本研究ではその枠組みを踏まえつつ，テキストベースプログラミングに適用可能な評価ルーブリック（表3）を新たに作成した．なお，「順次」「分岐」「ループ」の各項目におけるレベル２は，本講義の時間の制約により，if-else 構文やループ回数の指定を扱うことができなかったため，設定していない．

作成したルーブリックに基づき，５課題を合計26点満点で得点化し，受講生の理解度を評価した．

表3　テキストベースプログラミング理解度を評価するためのルーブリック

	レベル１（１点）	レベル２（２点）
順次	関数の呼び出し順序の理解	–
分岐	if 文を使った条件分岐の理解	–
ループ	ループを意識したプログラムの記述	–
データ利用	変数利用（定義・代入）	式の中で算術演算子を使用（インクリメント・デクリメントを含む）
モジュール	関数スコープの理解	パラメータ付き関数の理解

各項目について，達成に応じて点数を加点した．

レベル２を設定している項目については，レベル２が達成されていれば２点を付与するように加点した．

４．分析と考察

受講生140名のうち，未提出者および課題条件を満たしていない者を除外し．分析に必要なデータが揃っていた120名を有効サンプルとして抽出した．これらの受講生を対象として，3.1で示したScratchプログラムの各特徴量と，3.2で示したテキストベースプログラミング課題の得点との関係を分析した．

本分析では，サンプルデータ120名のテキストベースプログラミング課題の得点（０～26点）に偏りがあったことから，連続値の予測を行う回帰モデルではなく分類モデルを採用した．具体的には，26点の点数を半分に分け，０～16点（下位クラス），17～26点（上位クラス）の２クラスに分類した上で，Random Forest手法による分類モデルを構築した．

ハイパーパラメータはn_estimators=300（決定木数），max_depth=5（最大深さ），min_samples_split=10（内部ノードの最小分割サンプル数），min_samples_leaf=5（葉ノードの最小サンプル数）とした．これらの条件により作成した学習済みモデルに基づき，各特徴量がテキストベースプログラミング課題の得点に与える寄与率（feature importance）を算出した．

分析結果を，図１に示す．

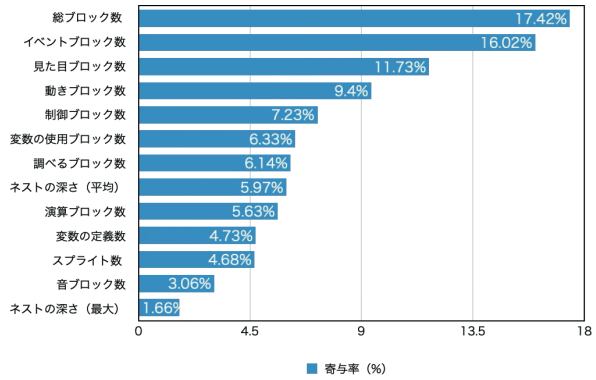

図１　特徴量から算出した寄与率

本分析におけるRandom Forestの性能指標として，学習データに対する正解率（Accuracy）は90%であった．

また，モデルの過学習の有無および汎化性能の安定性を検証するために，５分割層化交差検証（Stratified k-fold Cross Validation, k=5）を実施した．具体的には，上位クラスおよび下位クラスの２クラスに分類し，各分割においてクラス比率が可能な限り維持されるよう層化した上で，データ全体を５つのFoldに分割した．各Foldでは，全体の約1/5を検証用，残り約4/5を学習用として用い，これを５回繰り返すことで，全サンプルが１度ずつ検証用データとして評価される構成とした．検証結果を表４に示す．

表４　５分割層化交差検証における性能評価の結果

	Accuracy（%）	macro-F1
Fold1	83.33	0.454
Fold2	91.67	0.478
Fold3	87.50	0.466
Fold4	91.67	0.727
Fold5	87.50	0.466

その結果，正解率は平均88.33%（標準偏差3.49%），macro-F1は平均0.519（標準偏差0.117）となり，学習データと比較して性能が大きな低下は見られず，本モデルが一定の汎化性能を有していることが示された．ただし，テキストベースプログラミング課題の得点分布が上位クラスに偏っていたため，下位クラスのサンプルがFoldごとに変動しやすく，その影響によりmacro-F1の標準偏差の増大として現れたと考えられる．

分析の結果，「総ブロック数」（17.42%）が最も高い寄与率を示し，次いで「イベントブロック数」（16.02%），「見た目」（11.73%）が続いた．これら３項目で全体の寄与率の約45%を占めており，受講生が作成したScratchプログラムにおける構造的規模と制御構造の複雑さが，テキストベースプログラミング課題の理解度に強く関連していることが示唆される．

まず，「総ブロック数」の寄与が最も高かったことから，プログラム全体の構成要素の多さや作業量が一定の学習成果と関連していると考えられる．これは，市川（2020）の研究で報告された「総ブロック数とJavaScript理解度との正の相関」と一致しており，学習者が多くのブロックを操作する過程で，プログラミング的思考の深化が促進された可能性がある．

次に「イベントブロック数」及び「見た目」の寄与が高かった点について考察する．イベントブロックは，Scratchにおいてイベント駆動型処理構造や並列処理構造を担う要素であり，これを適切に活用できている学習者は，プログラムの流れや状態管理をより正確に理解していると考えられる．また，見た目ブロックについては，Scratchではスプライトの座標・向き・大きさなどの状態をシステム変数から参照可能であるため，ユーザ定義変数を用いずとも複雑な処理が実装できる．そのことか

ら，見た目ブロックは，ユーザ定義変数の代替的役割を担っている可能性があり，結果として「変数の使用ブロック数」や「変数の定義数」の寄与率が相対的に低かったと考えられる．一方で，「スプライト数」の寄与率は低く，多くの種類のオブジェクトを扱うこと自体は理解度と強く結びついていないことが示唆される．限られたスプライト数であっても，複雑な動きを設計することが重要であると考えられる．

５．まとめ

本研究では，ビジュアルプログラミングからテキストベースプログラミングへの移行過程における学習者の理解構造を明らかにすることを目的として，講義「プログラミング入門」における Scratch 課題とテキストベースプログラミング（p5.js）課題のデータを分析した．Scratch プログラムから抽出した構造的特徴量と，テキストベースプログラミング課題の得点との関係を，Random Forest 手法を用いて定量的に評価した．

その結果，「総ブロック数」「イベントブロック数」「見た目ブロック数」が，特に高い寄与率を示し，これら３項目で全体の寄与率の約45％を占めた．これにより，Scratch におけるプログラムの構造的規模（総ブロック数）に加え，イベント制御や状態管理に関するブロックの使用量（イベント・見た目ブロック数）が，テキストベースプログラミング課題の得点と関連していることが示唆された．これらの結果から，プログラミング環境（ビジュアル／テキストベース）にかかわらず，プログラム全体の構造を把握し，意図したとおりに動作させるための論理的構成力が関連している可能性を示している．

本研究の結果から，ビジュアルプログラミング教育においては，単にブロック数を増やすのではなく，イベント処理や単一のスプライトに対して複雑な動作を設計させる課題設計を行うことが，テキストベースプログラミングへの効果的な橋渡し（概念的転移）となることが示唆された．

一方，本研究におけるテキストベースプログラミング課題は，理解度を限定的な側面から評価するものであったため，Scratch 課題と同様のメイキング型課題をテキストベースプログラミングでも実施し，学習モデルの精緻化および更新を進めていきたい．

参考文献

１）文部科学省：小学校プログラミング教育の手引（第３版）（2020）．
２）Scratch：https://scratch.mit.edu/
2025年10月20日参照
３）太田 剛 , 森本 容介 , 加藤 浩 : プログラム機能の自動分析機能とプログラム概念の自動評価機能を持つ Scratch 用プログラミング学習支援システムの開発 , 情報教育シンポジウム2016論文集2016, 106-113（2016）．
４）Jesús Moreno-León, Gregorio Robles, Marcos Román-González: Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking, Revista de Educación a Distancia (2015)
５）太田 剛，加藤 浩，森本 容介：子供のプログラミング能力の獲得段階に関する定量的分析：小学校４～６年生の Scratch プログラミングを対象として，情報処理学会論文誌教育とコンピュータ，35-43（2019）
６）市川 大祐：テキスト型言語習得を目指した Scratch によるプログラミング導入教育の実践と評価，日本工学教育協会「工学教育」誌，2022年１月号（2022）
７）Github：https://github.com/Yusk1450/scratch_count
2025年10月20日参照