OCaml Blockly チュートリアルの改良と専用記述言語

田村 優衣　(指導教員：浅井 健一)

1　はじめに

OCaml Blockly [4] は，OCaml のビジュアルプログラミングエディタである．エラーを起こすプログラムは構築できないため，プログラミング初学者が簡単にプログラミングを学ぶことが可能である．

さらに，簡単なゲーム作成を目標としたチュートリアルサイト [2, 3] が作成されている．チュートリアル (図 1) では，操作方法を学びながら，プログラミングを学ぶことができる．このチュートリアルを活用するためには，新規作成が不可欠であるが，現行の実装では，開発者以外がチュートリアルを作ることは極めて困難となっている．

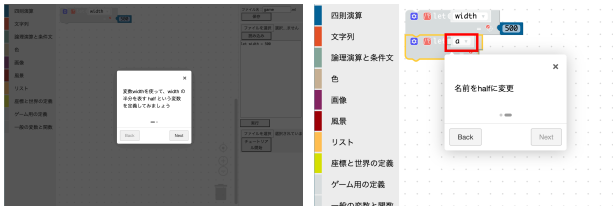

図 1: チュートリアル

本研究では，OCaml Blockly のチュートリアル作成の簡易化を目指し，実装プログラムの改良とチュートリアル記述の専用言語を作成した．

2　現行のチュートリアル実装と問題点

主な使用言語は，JavaScript である．特に，ガイドの表示には，Intro.js [1] という簡単にステップ式チュートリアルが作成できるライブラリを使用している．

チュートリアルの内容は，オブジェクトのリスト形式で記述されている．例えば「四則演算メニューを開いて，整数ブロックを既にワークスペースにある四則演算ブロックの右側にドラッグする」というチュートリアルの記述は，図 2. のようになる．

```
{
  "text": [[],[],[],],
  "category": 0,
  "block": 0,
  "id": 1,
  "target": [0, "B"],
},
```

図 2: オブジェクトの例

現行のチュートリアル記述は，実装プログラムと深く関わっているため，全てを熟知した上で新たにチュートリアルを作成するのは現実的ではない．

3　提案する記述方法

現行の問題点を踏まえて，直感的なチュートリアル記述を目指し 3 つの目標を掲げる．

- 実装プログラムの理解を不要にする
- ユーザーの操作と一対一に対応
- 操作手順に基づいた記述

これらの目標を踏まえて，図 2 のオブジェクトを，図 3 の形式で記述できるようにする．

```
openMenu("四則演算");
dragToTarget("NUM","整数",["PLUS","RIGHT"]);
```

図 3: 記述方法の提案（TypeScript）

しかし，これは実装プログラム内に記述するため，変更する度にビルドする必要がある．また，この例では既に四則演算ブロックがワークスペースにあるはずだが，そのことがチュートリアル記述から読み取れない．そこで，さらに 2 つの目標を設定する．

- 実装プログラムとチュートリアル記述を切り離す
- チュートリアルに必要な要素を 1 つにまとめる

これらを実現するために、専用の記述言語を作成する．

4　実装プログラムの改良

本研究では，チュートリアル実装の大幅な改良を行なった．改良の目的は 3 つに分けられる．

- プログラムの可読性向上 (4.1, 4.2 節)
- チュートリアル記述の実現 (4.3, 4.4 節)
- 新機能の実装 (4.5, 4.6 節)

この節では，改良点の詳細を示す．

4.1　使用言語の変更

使用言語を TypeScript に変更し、型システムを活用する．OCaml Blockly の型定義ファイルを作成し，明示的な型付けをすることで，プログラムの可読性を向上させた．型エラーによって，実行前に誤りを検出できたため、開発が円滑に進んだ。.

4.2　メニュー上のブロックの管理方法

これまで，オブジェクトの category と block の値を用いることでブロックを指定していたが，番号での指定はプログラムの可読性が低くなる．本研究では，メニューやブロックの指定をテキストで行う．このテキストは吹き出しに表示されるものと同じである．リストで管理していたデフォルトのブロック情報をオブジェクトで管理することで，プログラムの可読性が向上した．例えば，OCaml Blockly で定義されているブロックの型が必要な場合、以下のように参照する．

旧: `blocklst[category][block][0]`
新: `getBlockObj[blockName].type`

4.3　操作ごとに独立したチュートリアル関数

操作と一対一対応する記述を可能にするために、チュートリアルに必要な OCaml Blockly の操作を精査し、操作ごとにを独立した関数として実装した。これらチュートリアル関数と定義する。これにより，操作手順通りの記述も可能になった．図 3. では，`openMenu`，`dragToTarget` がチュートリアル関数である．

4.4 ブロック変数とブロック環境の導入

これまで，ワークスペース上のブロックはリストで管理し，番号でブロックを指定していた．これは，リストの中身を常に把握しておく必要があり不便である．そこで，ブロック変数とブロック環境を導入した．ブロック環境は，ブロック変数を key としてブロック情報を管理する．新しいブロックをワークスペースに出すチュートリアル関数では，引数としてブロック変数を受け取り，ブロック環境に追加する．図 3. では，`dragToTarget` で整数ブロックに `NUM` というブロック変数を付けている．

4.5 初期コードのブロック変数を定める

チュートリアルでは，学ぶことに焦点を当てるために，最初からブロックが表示されることがある．これらのブロックをチュートリアルで使用するには，ブロック変数を与える必要がある．初期ブロックは，チュートリアル開始前に OCaml のコードを読み込み，ブロックに変換して表示しているので，OCaml コードに Attributes の形でブロック変数を書くようにした．OCaml の Attributes は，コードに型チェックの影響を受けずに追加情報を付与する機能であり，`[@と]` で囲われた中に記述する．例えば，図 4. のようなブロックで，グレーの関数定義ブロックを「`ADD10`」，青の四則演算ブロックを「`PLUS`」とするには，次のように書く．

```
let[@ADD10] add10 x = (x + ?)[@PLUS]
```

図 4: 関数定義ブロックと四則演算ブロック

4.6 スコープお砂場に対応

OCaml Blockly の機能の一つであるスコープお砂場では，そこで定義されているブロックを使うことができる．図 5. のように，既にブロックが接続されているコネクタに新たなブロックを接続すると，元のブロックがスコープお砂場に押し出される．現行のチュートリアルでは，スコープお砂場に押し出されたブロックを指定する方法がないため，そのブロックを使ったチュートリアルを続行することができなくなってしまう．

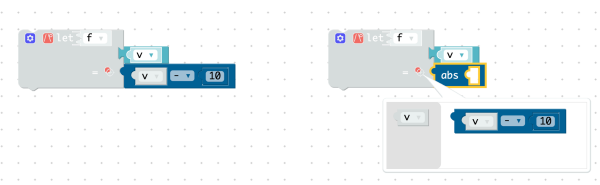

図 5: ブロックがスコープお砂場に押し出される

これを解決するために，ブロックを接続する操作があるチュートリアル関数に機能を加えた．スコープお砂場にブロックが移動した時に，該当のブロックをブロック環境に追加し，その後のチュートリアルで使用可能にする．そして，スコープお砂場に押し出されたブロックを使うために，スコープお砂場からブロックを移動させるチュートリアル関数を作成した．

5 専用記述言語

5.1 設計

初期コードとチュートリアル記述が書かれたファイルを読み込むことで，チュートリアルが開始する仕組みとする．先頭の %{ と%} の中に，OCaml コードを記述する．チュートリアル記述は，チュートリアル関数と対応した，自然言語に近い文法にすることで直感的な記述を目指した．

```
%{
let[@ADD10] add10 x = (x + ?)[@PLUS]
%}
open 四則演算メニュー;
literal [四則演算ブロックの右側にはめます];
NUM = drag 整数 to (RIGHT of PLUS);
complete;;
```

図 6: 専用記述言語の提案

5.2 実装

図 6. の専用記述言語を，TypeScript の実装プログラムで処理できる形式に変換するために，コンパイラを作成した．変換する際に，実行時エラーが起こり得るプログラムはコンパイルエラーとなるようにした．

6 まとめと今後の課題

本研究における成果は 2 つある．まず，チュートリアルの実装プログラムの改良を行なった．これにより，操作と一対一対応かつ逐次的なチュートリアル記述を可能にした．そして，専用の記述言語を作成した．実装と記述を分離することで，チュートリアル作成者が実装プログラムに触れる必要がなくなった。

現行のチュートリアルサイトには 29 個のチュートリアルがある．現在これらの全てを，実装プログラム内に記述する方法 (図 3) で再現できている．チュートリアル記述言語は実装途中で，DSL 用コンパイラの作成に取り組んでいる．

今後は，チュートリアル実行時のエラーの処理を実装する．現状のチュートリアルでは，想定外の操作をすると最初からになってしまうので，適切なステップでチュートリアルに復帰できる仕組みを整える．また，実行時にエラーが生じるようなチュートリアル記述を，コンパイルの際にチェックできるように DSL 用コンパイラを改善する．

参考文献

[1] Intro.js. `https://introjs.com/`.

[2] 柴田真琴. OCaml Blockly のチュートリアルサイト. `http://pllab.is.ocha.ac.jp/~asai/jpapers/ppl/24/kyozai/demos/tutorial/tutorial.html`.

[3] 柴田真琴. OCaml Blockly のチュートリアルサイトの理解度向上に向けた改善. 修士論文, お茶の水女子大学, 2023.

[4] 松本晴香, 浅井健一. Blockly をベースにした OCaml ビジュアルプログラミングエディタ. 第 21 回プログラミングおよびプログラミング言語ワークショップ, 2019.