OCaml Blockly を使用した代数教材の作成

守本 梨紗　(指導教員：浅井 健一)

1　はじめに

本研究をはじめたきっかけとして、OCaml Blockly を使用して「ゲームをしながらネット上で感覚的に学べる教材」を作ってみたいという考えから、視覚的にわかりやすく表現できる代数教材にチャレンジしてみることになった。7 節で関連研究を紹介する。

本研究の目的は、OCaml Blockly を使用して、中高生を対象に、「楽しく、関数的な感覚が身につくゲーム」を作ることである。

2　OCaml Blockly とは

Blockly とは、Google が開発した、テキスト形式ではなくブロックなどのオブジェクトを組みたててプログラミングができる、ビジュアルプログラミングエディタである [3]。

OCaml Blockly とは、この Blockly をベースとして実装された、型システムや変数束縛を直感的なユーザインタフェースとして備えた、OCaml ビジュアルプログラミング環境のことである [4]。

3　作成したゲームについて

セクション 1 とセクション 2 に分かれており、セクション 1 では $f(x,y)=(x+dx,y+dy)$ で表される一次関数、セクション 2 では $y=f(x+dx)$ で表されるいろいろな関数について学ぶことができる。

セクション 1 は、切片などの考え方も導入した一次関数に視点をおいたゲームである。与えられた dx、dy と初期位置を変えていくことで飛行機がさまざまな直線を描き、ゲームクリアを目指す。

最初に飛行機を置く座標（初期座標）を指定する方法を学ぶ。次に現在の飛行機の座標を与えられたら 1 秒後の飛行機の座標を返す関数を $f(x,y)=(x+dx,y+dy)$ の形で定義する方法を学ぶ。これを指定すると飛行機は 1 秒ごとに (dx,dy) の方向に進むようになる。これを使って原点から目的の座標に飛行機を飛ばすようにする。これを通して、1 次関数を飛行機の移動という形で直感的に理解できるようにする。

stage1... 初期座標を変える
stage2-1,2... 横/縦に動かす
stage3...x 軸/y 軸上の点からゴールを目指す
stage4... 障害物を避けて自由にゴールを目指す

セクション 2 では、一次関数の形だけではなく、好きな関数 $f(x)$ を設定していくことでゲームクリアを目指す（現時点では二次関数まで設定可能）。

最初に飛行機を置く座標（初期座標）を指定する方法を学ぶ。次に 1 秒ごとに x 軸方向に dx ずつ進むとして、現在の飛行機の x 座標と dx を与えられたら、1 秒後の飛行機の y 座標を返す関数を、$y=f(x+dx)$ の形で定義する方法を学ぶ。これを指定すると (x,y) にいた飛行機は 1 秒ごとに $(x+dx,f(x+dx))$ へ動き続けることになる。これを使って原点から目的の座標に飛行機を飛ばすようにする。これを通して、関数 $f(x)$ を飛行機の移動という形で直感的に理解できるようにする。

stage1... 初期座標を変える
stage2... 横に動かす
stage3-1...$(0,0)$ からゴールを目指す
stage3-2...$f(X)=g(X)+a$
stage4... 障害物を避けて自由にゴールを目指す

4　遊び方について

ゲームのトップページ (http://pllab.is.ocha.ac.jp/morimoto/) から、スタートを押す。遊ぶステージごとに画面右側のプルダウンから該当のブロックを選択し、読み込み、出てきたブロックを編集して実行する。

5　研究の概要

研究の概要は大きく二つに分けられる。一つ目はゲームの中身を作ること、二つ目は、ゲームをする際ユーザーが実際に触る画面を作ることである。ここでは、特に自分なりに工夫した機能のみ、いくつか紹介する。

5.1　ゲームの中身について

1. ゲームが実行される画面のデザイン
2. y 軸の方向を上向きに、かつ、中心の座標を基点に画像を表示できるよう仕様変更
3. セクションごとの関数設定
・セクション 1　$f(x,y)=(x+dx,y+dy)$
・セクション 2　$y=f(x+dx)$
4. 教材としてのストーリーを考案（3 節参照）

5.2　ユーザー画面について

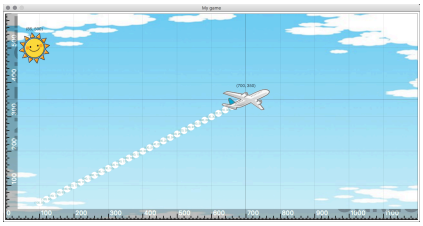
1. ゲームに必要なコードのみブロックにする
2. 遊び方のページを作る
・ゲームを進めるための基本操作や、Blockly の操作方法についてのページを作成した。
3. ローディング画面の作成
・現状、ブロックを読み込むまでに時間がかかり、読み込み中の操作によってバグが多発していたため、読み込み中操作ができないようにした。
4. プルダウンを作る
・stage ごとに該当のブロックを選択できる。
5. 実行画面を外部出力させる
・実装前はゲーム実行画面の全体を見ることができなかったため、外部出力させた。
6. ローカル環境からサーバーへアップロード

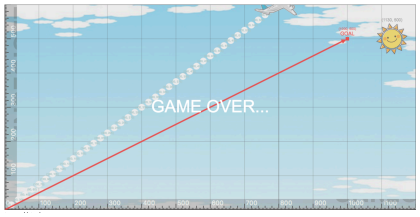
6　具体的な実装内容

前節では、本研究の概要を説明した。本節では、前節で紹介した機能の一部について実装内容を説明する。

6.1　ゲームの中身について

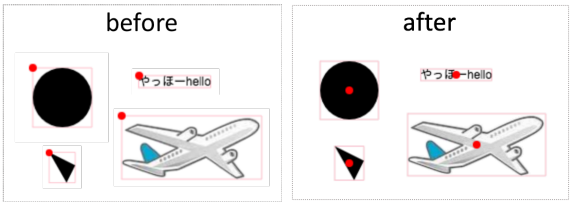
1. ゲームが実行される画面のデザイン
目盛り線や座標、飛行機雲を表示させることで、目に見えてより直感的に扱えるようにした。また、間違えた時、どう違うかを画面に示すようにした。





2. y 軸の方向を逆向きに、かつ、中心の座標を基点に画像を表示できるよう仕様変更
・画像を表示させる関数の修正
・画像の縦幅・横幅を出す関数の作成など

画像を (x, y) に置こうとした時、以下の画像の赤点の位置に (x, y) 座標がくるように画像は置かれる。



実装以前 (上図左) では、画像を表示させる関数に変更を加え y 軸を上向きに変更すると、位置が変わってしまい、また、画像同士の位置関係を知りたい時、それぞれの横幅縦幅から一々計算しなくてはいけなかった。(その上、当初は画像の縦横幅を出す関数がなかったため、位置を直したり計算する術がなかった。)

実装後 (上図右) は、画像の横幅と縦幅に関係なく中心座標のみで画像を置くことができ、そのため画像同士の位置関係も明白になった。

3. 関数設定

代数教育の上では $y = f(x)$ の形が一般的のため、OCaml の性質を生かした $f(x, y) = (x + dx, y + dy)$ とするセクション 1 に加えて、より代数教育の形に寄せたセクション 2 と、二つの視点から関数を学べるようにした。

下の？の部分に学習者はブロックをあてはめ、ゲームクリアを目指す。

・セクション 1

```
let initial_airplane_zahyou = ?
let dx = ?
let dy = ?
let f (x, y) = (x + dx, y + dy)
```

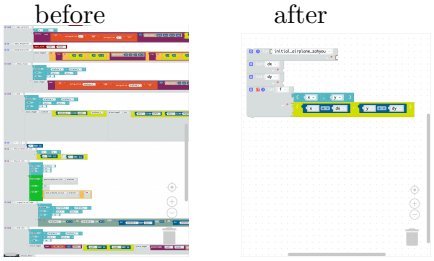
・セクション 2

```
let initial_airplane_zahyou = ?
let f x = ?
let dx = ?
let move_airplane_on_tick (x, y) =
  (x + dx, f (x + dx))
```

6.2 ユーザー画面について

1. ユーザーに必要なコードのみブロックを表示
・このようなゲームを作るには、1 次関数についての部分以外にも、雲や軌跡の表示、tick イベントの処理など多くのことを書かなくてはならない。しかし、これらは 1 次関数を学ぶという目的には関係ないので、その部分のプログラムは見せたくない。そこで、stageX のゲームのソースコードを X-before.ml と X.ml と X-after.ml の三つに分け、X.ml のみブロックとして表示させ、ゲーム実行時には三つのファイルを合わせて実行させる仕様にした。

before　　after



7 関連研究

本研究と同じくブロックを組み合わせる形式で、大人気ゲーム「Minecraft」の世界を動かしながら、パズル感覚でプログラミングを学ぶことができるゲームが実際に公開されている [2]。本研究は、プログラミングではなく代数を学ぶことが目的である。

また、プログラムに繰り返しをコーディングすることなく、学生がビデオゲームの構築と代数的な問題解決を学ぶことができることも示されている [1]。本研究では、ビデオゲームではなくパズルゲームをしながら学ぶことを目的としている。

8 まとめ

本研究では、遊ぶことで関数的な感覚が身につく、OCaml Blockly を使用したゲームを考案した。ゲームクリアしていくことで、セクション 1 では x 軸方向 y 軸方向それぞれの動きについて、セクション 2 では x の一次関数、二次関数について学ぶことができる。

本研究の直近の課題は、ストーリーをより教科書らしくしていくことと、Google Chrome で正常に実行できるようにすること、また、将来的には二次以上の関数についても学べるようにすることや拡大縮小機能をつけて、関数の軌跡を大きくみることを可能にすること、などが挙げられる。

加えて、本研究では代数に関してゲームを作成していたが、今回の研究を土台にして他分野に関してのゲーム教材を作成することも今後の課題としていきたい。

参考文献

[1] BootStrap. Algebra. https://www.bootstrapworld.org/materials/algebra/.

[2] Code.org. Minecraft hour of code. http:://code.org/minecraft.

[3] Google. Blockly. https://developers.google.com/blockly/.

[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ワークショップ論文集, p. 15, 2019.