Design and Implementation of "Puzzle Programming" -New Teaching Method for Programming Education-

Kazuhide YAMAMOTO*, Noriyuki ISHIGURO, Yusuke SAITO**, Hirohide HAGA***

(Received April 18, 2014)

Currently students in elementary, junior, and senior high school are encouraged to learn programming in their class of informatics. In this research we propose visual programming environment named "Puzzle Programming" to make studying programming with pleasure. Puzzle Programming is implemented using Unity game engine. All data for puzzle programming is managed by Yaml, human readable data serialization format. Prototype puzzle programming system was implemented on Apple iOS. Tablet PC iPad is used for playing "Puzzle programming." Experimental evaluation proved the effectiveness of "Puzzle Programming" for programming study.

Key words : Information Education, Programming Education, Serious Game, Puzzle Programming

キーワード : 情報教育，プログラミング教育，シリアスゲーム，パズルプログラミング

パズルを用いたビジュアルプログラミング教材の実装と評価

山本　一秀，石黒　紀悠，齋藤　雄輔，芳賀　博英

1. はじめに

近年，教育現場において，情報教育の一環としてのプログラミング教育の導入が進んでいる．2012年の 4 月より実施された文部科学省の新学習指導要領では，中学校の技術・家庭科の項目に「プログラムによる計測・制御」が追加され，これにより中学校においてプログラミングが必修化された[1]．また，2013 年 6 月に政府が発表した成長戦略素案では，「産業競争力の源泉となるハイレベルな IT 人材の育成・確保」の項目で，「来年度中に産学官連携による実践的 IT 人材を継続的に育成するための仕組みを構築し，義務教育段階からのプログラミング教育等の IT 教育を推進する．」とあり[2]，義務教育段階のプログラミング教育は，今後さらに政策上の重点を置かれるものと考えられる．

プログラミングが義務教育段階から必修化されることで，これまでは情報系の専門学校や大学での必修科目，あるいは理系の大学生の教養科目として，プログラミングに対し一定以上の興味があると考えられる層にのみ行われていたプログラミング教

* Graduate School of Science and Engineering, Doshisha University, Kyoto
Telephone:+81-774-65-6979, E-mail:kyamamoto@ishss10.doshisha.ac.jp
** Graduate School of Information Science, Nara Institute of Science and Technology, Nara
*** Department of Science and Engineering, Doshisha University, Kyoto
Telephone:+81-774-65-6978, E-mail:hhaga@mail.doshisha.ac.jp

育が, その対象をプログラミングについての興味や予備知識に乏しい層にまで広げることになる. そのため, プログラミングに興味の無い人の理解を助けるために, 本研究では C 言語のようなテキスト記述によるプログラミングではなく, パズルを用いたタブレット端末向けビジュアルプログラミング言語(Visual Programming Language, 以下 VPL と略す)教材「Puzzle Programming」を開発した. さらに, 作成したプログラムを用いてゲームをプレイできるようにし, ゲームのエンターテイメント性によるユーザの学習意欲の向上を計った.

「Puzzle Programming」の開発後には, 同志社国際中学校・高等学校にて「Puzzle Programming」を用いたプログラミング体験授業を実施し, 生徒に対してアンケートを取って「Puzzle Programming」の評価を行った.

2. ゲームデザイン

2.1 「Puzzle Programming」の設計方針

「Puzzle Programming」の開発目的は, プログラミング経験のない初学者のユーザが抵抗感なくプログラミングの学習を始め, 将来的にテキスト記述のプログラムの学習へとスムーズに移行できる学習環境をつくることである.「Puzzle Programming」はタブレット端末上での動作を想定して開発しており, ユーザがタブレット端末上でジグソーパズルを組み立てるように操作することでプログラムを記述する. パズルを用いることにより, ユーザはテキスト記述型言語のようにキーボードによるテキスト記述を行うことなく, 直感的な操作でのプログラム記述が可能となっている. また, プログラムをパズルで表現したことのもう 1 つの利点としては, プログラムの文法規則をパズルのピースの形状の物理的制約に置き換えたことにより, プログラムを記述する際のエラーを未然に防ぐことができる点が挙げられる.

「Puzzle Programming」では, 作成したプログラムを用いてキャラクターを操作するシリアスゲームを設けている. シリアスゲームは「教育をはじめとする社会の諸領域の問題解決のために利用されるゲーム」と定義され[3],「Puzzle Programming」における「迷路ゲーム」及び「ロボットゲーム」がこれに該当する. ゲームのエンターテイメント性により, プログラミングに興味の無いユーザの関心を集めること, 及びユーザの継続的な学習の意欲向上を目的とし, シリアスゲームを採用した.

2.2 操作方法

Fig. 1 の「Puzzle Programming」の画面は, 本システムのメイン画面である. 画面右側上部のピースを選択するセレクタ, 左側のピースを置くことが可能なセル（マス目）, 及び左側上部のプルダウンメニューにより構成されている.

「Puzzle Programming」には, ログを出力する「プログラミング」, シリアスゲームである「迷路ゲーム」,「ロボットゲーム」の 3 種類のゲームタイプがあり, ユーザはどのゲームタイプでプログラミングを行うのかを画面上部のゲームタイプのメニューから指定することができる. セレクタには, 様々な形や色をしたパズルのピースが「関数, 条件, 変数」の 3 種類にカテゴリ分けされて配置されている. Fig. 2 に示すように, 関数のカテゴリには, それぞれのゲームタイプ（プログラミング, 迷路ゲーム, ロボットゲーム）に対応した API(Application Program Interface)が割り当てられた「関数ピース」と, それに対応した「引数ピース」が用意されている. 条件のカテゴリには, 条件判定処理を行う「If ピース」と, 繰り返し処理を行う「While ピース」が用意されている. 変数のカテゴリには, 変数を利用する際に用いる「変数代入ピース」と変数や数字を用いて算術演算を行う「計算ピース」が用意されている. これについては 2.3 節の「ピースのマッチング効果」にて述べる.

パズルによって組まれたプログラムは, Fig. 1 中のプルダウンメニューの実行ボタンを押すことで実行が開始される. ゲームタイプが「プログラミング」の場合, ログ出力画面（Fig. 3）が表示され, 組まれたプログラムのログを逐次に出力する. また, ゲームタイプの「迷路ゲーム」,「ロボットゲーム」の出力については, 2.5 節の「シリアスゲームとの

連携」にて述べる．

Fig. 1. Top screen of puzzle programming system.

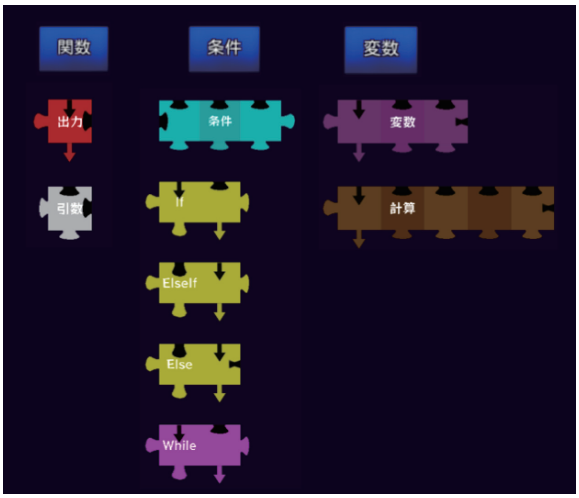

Fig. 2. Pieces included in “Function” category.

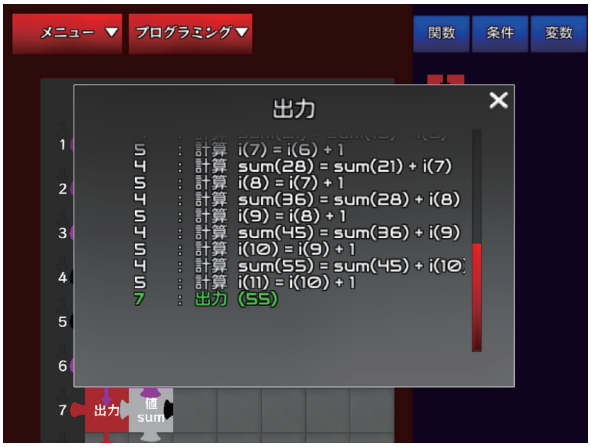

Fig. 3. Display of execution log.

2.3 ピースのマッチング効果

プログラミングをパズルで表現したことの利点は，ユーザにとってなじみやすいことだけでなく，パズルの形状によってプログラミングの文法規則上の制約を表現できることも挙げられる．Fig. 4 のように，条件判定処理を行う「If ピース」と「条件ピース」は互いにマッチするようデザインされており，隣り合って配置することができる．このように，ピースの形状がマッチするということは，「文法規則に添っている」ことを意味している． 一方，ピース同士がマッチしなければ，当然隣同士にピースを置くことができない．隣同士にピースを置くことができないということは，「文法規則に反している」ということを意味している．このように，パズルの形状に注意しながらプログラミングを行うことにより，ユーザはエラーを回避しながらプログラミングを行うことができる．これにより，初学者でもエラーによって作業を滞らせることなく学習を進めることが可能となっている．

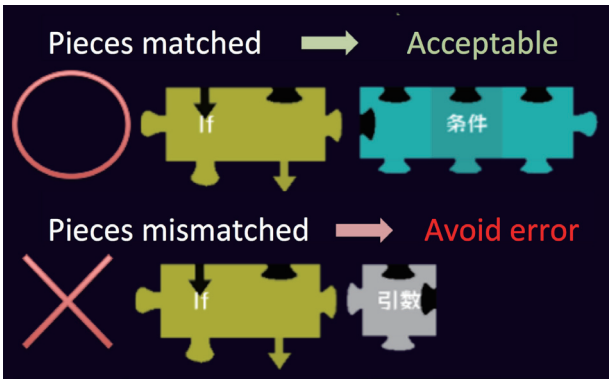

Fig. 4. Display of syntax structure by pieces matching.

2.4 既存の VPL との比較

「Puzzle Programming」は，ユーザが将来的に一般的なテキスト記述のプログラミング言語を学ぶことを考慮した上でデザインした．

既存の VPL として「Scratch」[4]（Fig. 5）を例にあげる．「Scratch」においては，For 文（Fig. 5 の上側）にあたる繰り返し処理のブロック（Fig. 5 の下側）には，「何回までループを繰り返すか」を指定するための「条件」があらかじめ付随している仕様

となっている．

```
for (int i = 0; i < v.length; i++) {
    v[i]++;
}
```

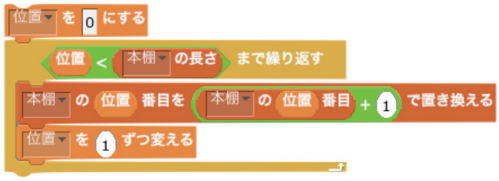

Fig. 5. Sample of “Scratch” visual programming environment.

一方「Puzzle Programming」は，Fig. 6 に示すように，繰り返し処理を行う While ピースに対して，While ピースとは異なる「条件ピース」を用意し，それら二つのピース同士を組み合わせることで繰り返し処理を可能としている．したがって，ピース同士を組み合わせるという行為から，視覚的に二つのピースの関係性を伝え，「While ピースには，必ず条件ピースが必要」といったように，本来のプログラミングの文法規則への連想を強めることができる．これにより，ユーザが将来的にテキスト記述のプログラミング言語に移行した際，繰り返し処理には必ず「条件」が伴うことを意識することを狙いとしている．

Fig. 6. “while” statement in puzzle programming environment.

If ピースに関しても，While ピースと同様に「条件ピース」を用いる．また，関数である出力ピースに関しても，引数をあらかじめ付随させたピースの作りにするのではなく，関数ピースとは異なる「引数ピース」を設けることにより，よりテキスト記述のプログラミング言語に添った仕様をとっている．

2.5 シリアスゲームとの連携

「Puzzle Programming」では，作成したプログラムを実行してログ出力画面を表示するだけでなく，作成したプログラムによってゲームタイプ「迷路ゲーム」，「ロボットゲーム」においてキャラクターを動かすことができる．作成したプログラムによってゲームをプレイすることで，ユーザの学習意欲を促進することを狙いとしている．

以下，「迷路ゲーム」と「ロボットゲーム」それぞれのゲーム内容について述べる．

2.5.1 迷路ゲーム

迷路ゲームは，キャラクターをスタート地点からゴール地点まで，障害物を避けながら導くことでステージをクリア（Fig. 7）できるゲーム内容となっている．キャラクターが移動中に障害物への衝突，もしくはエリア外に侵入すると，ゲーム画面上に失敗を意味する「Fail」の文字（Fig. 8）が表示される．

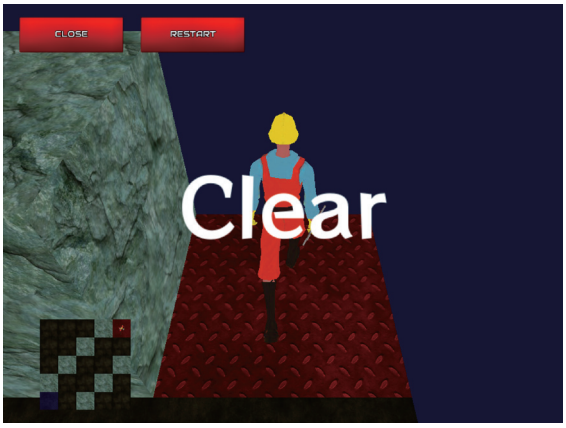

Fig. 7. Screenshot of “maze” game.

Fig. 8. Screenshot of “maze” game.

2.5.2 ロボットゲーム

ロボットゲーム（Fig. 9）は，迷路ゲームと同様，「Puzzle Programming」上で組まれたプログラムの実行により，ロボットが左右に回転または前進などの動作を行う．ロボットの行く手を阻む障害物が周囲に置かれていないため，「迷路ゲーム」より比較的自由に動作させることができる．Fig. 10 に示すように，関数ピースの横に引数ピースを置くことで，前進か後進かどうか，回転するスピード，秒数などのパラメータの指定が可能となる．ロボットが可能な運動としては，円運動，四角運動，ギザギザ運動，8 の字運動などがあげられ，無限ループ（While ピース）を用いることで，同じ動作を繰り返すこともできる．

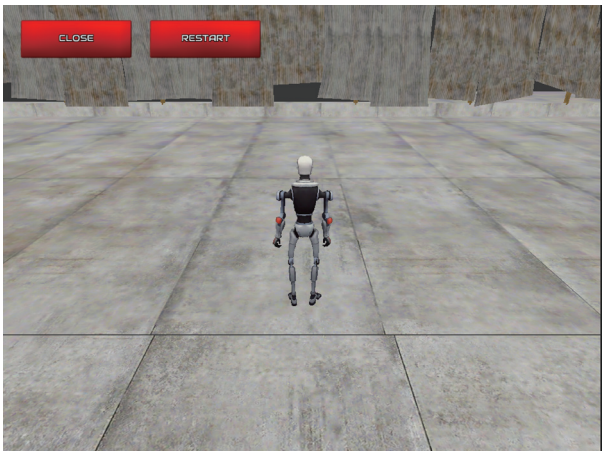

Fig. 9. Screenshot of “robot” game.

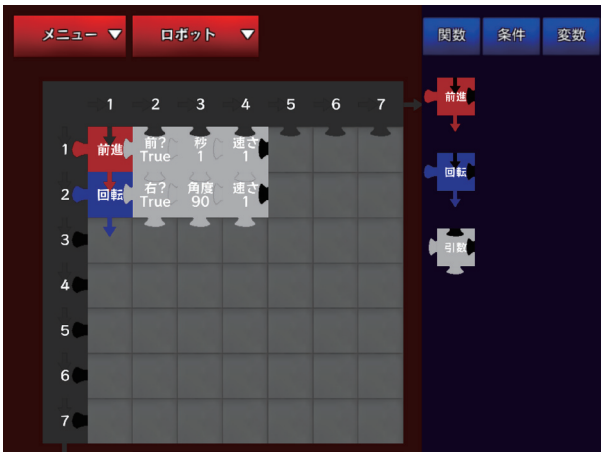

Fig. 10. Programming robot game in puzzle programming environment.

3. 開発

3.1 開発環境

「Puzzle Programming」を開発するにあたって，ゲーム開発エンジン「Unity」[5,6]（ユニティ）」を用いた．Unity とは，米 Unity Technologies が提供している本格的なインタラクティブ 3D アプリケーションの開発を容易にするゲーム開発エンジンである．Unity は 3D の開発を得意としているが，カメラの機能を 2D 専用のものに切り替えることで 2D のアプリケーションの開発も可能である．今回の「Puzzle Programming」はこの方法により，2D のアプリケーションとして実装した．

Unity の特徴としては，Wii，Xbox360，PlayStation 3 などのコンソールゲーム機器向けから，iOS，Android，Windows，Mac などの様々なプラットフォームに対応しているクロスプラットフォームである点や，開発における初期コストが他の開発エンジンに比べてリーズナブルな点が挙げられる．

3.2 システム構成

Fig. 11 に「Puzzle Programming」の機能構成図を示す．「Puzzle Programming」では，主にパズルピースとそのピースを配置するセルから構成されている．パズルピースにはピースとプログラム処理の情報，セルはそのピースを保有する形となる．本章で

は，各構成要素に分けて述べる．

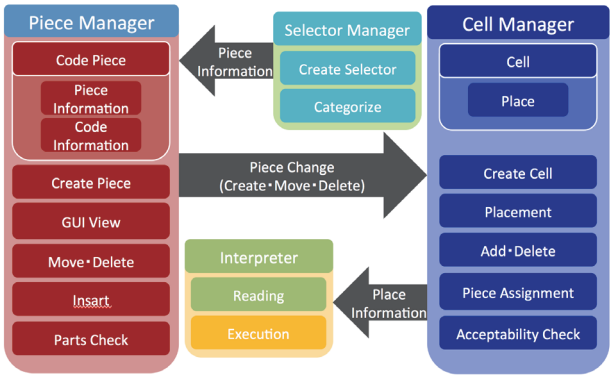

Fig. 11. Function components of puzzle programming.

3.2.1 Yaml

本研究では，データ管理に「Yaml」[7]と呼ばれる文字列型のデータ形式を用いる．Yaml はXMLやC，Perl と言ったプログラミング言語からきているとされ，マークアップ言語よりデータ重視を目的としたデータ形式である．Yaml によるリスト，ハッシュ，コメントアウトの表記例を Table 1 に示す．

Table 1. Examples of Yaml description.

Items	Description examples
List	[apple, orange, grape]
Hash	key : value name : Yusuke Saito
Comment out	# comment

表記例からもわかるように，テキストで書かれているため可読性が高く，複雑でないデータの管理において Yaml は有用であると考える．本研究では Unity で Yaml データを扱うため，独自のパーサを Unity 用に開発し，使用している．

3.2.2 パズルピース

前述のとおり，「Puzzle Programming」ではパズルのピースが大きな役割を持っている．画面上では Fig. 12 の左図のように見えるが，実際は右図のように凹凸のパーツごとにわけて配置している．このピースの構成情報は Yaml によって管理されており，Fig. 12 のピースのテキストデータを List 1 に示す．

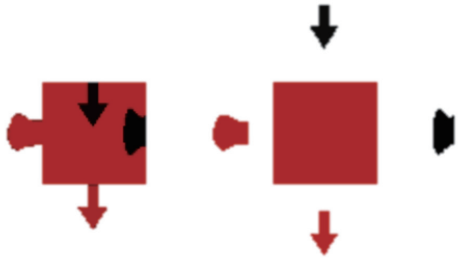

Fig. 12. Decomposition of pieces.

List 1. Yaml description of a puzzle piece (Fig. 12)

```
id        :4                          # ID
condition :[false, false, true, true]
                                      # Concave or convex
type      :[3, 1, 3, 2]               # Figure type
color     :[180, 0, 0, 255]           # Color
```

上記の例のキー「condition」と「type」には［上，右，下，左］の順で各パーツの形と凹凸を指定する．また，新しくテキストデータを作成することで，様々な形や色のピースを作成することができる．

このパズルピースにコード情報を付与したものを本研究ではコードピースと呼ぶ．実際に画面上に配置されるものはすべて Fig. 13 のようなコードピースとなる，コードピースはパズルピース同様 Yaml で管理されており，Fig. 13 を例としたテキストデータを List 2 に示す．

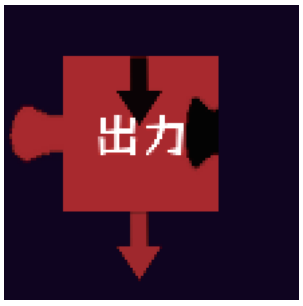

Fig. 13. Composition result.

List 2. Yaml description of a code piece (Fig. 13)

```
name      :出力                  # Code piece name
piece     :[4]                   # Piece ID
class     :DebugCode             # Class name
method    :Debug                 # Function name
p_name    :[値]                  # Argument name
p_type    :[System.Single]       # Argument type
p_def     :[1.0]    # Argument default value
return_type:        # Return value type
category  :0        # Category number
```

コードピースを実行する際は，List 2 中のキー「class」,「method」,「p_type」を用いて動的に関数を呼び出している．またキー「category」は，ピース選択画面の関数,条件,変数の種類にそれぞれ0, 1，2 が割り振られている．

3.2.3 インタプリタ

Fig. 14 に「Puzzle Programming」のインタプリタ処理手順のフローチャートを示す.

「Puzzle Programming」のインタプリタでは，処理の現在位置を元に,その位置にあるセルおよび配置されたコードピースを参照しながら実行していく．また，エラー処理に関しては，基本的にパズルの制限によってなされている．しかし「Puzzle Programming」では変数宣言を絶対としていることにより,宣言されていない変数は使用不可能なため,宣言されていない変数の使用時のみエラー処理をインタプリタ内で行っている.

3.2.4 シリアスゲームとの連携

2 章で述べたように,「Puzzle Programming」は 2 つのシリアスゲームと連携している．Unity では画面単位で開発が行われ,作られた 1 画面をシーンと呼び,これらのシーンの組み合わせや遷移によってゲームを開発していく.「迷路ゲーム」と「ロボットゲーム」は，それぞれシーンを別々に作成して開発を行った．シーンを分けて開発することで，ゲーム単位でのデバッグが容易であり，「Puzzle Programming」のシーンにすべてのゲームを含めるのはメモリ上無駄が多いと判断したからである.

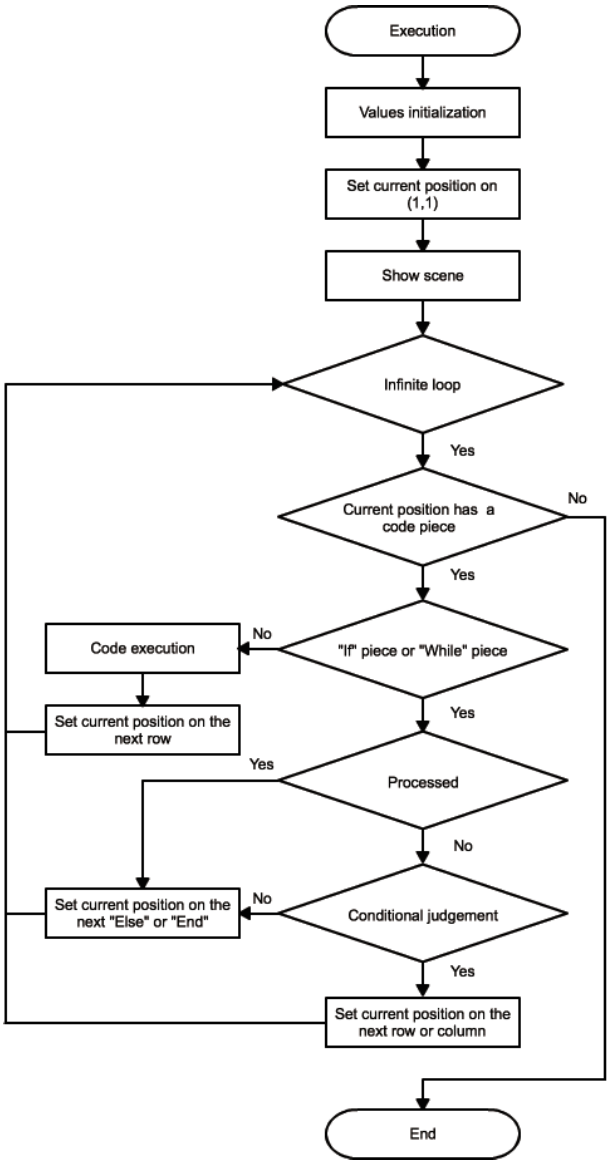

Fig. 14. Flowchart of puzzle programming interpreter.

作成されたシーンは,画面メニューのゲームタイプ変更時に「Puzzle Programming」のシーンに取り込まれる．また再度変更した場合は，変更前に表示しているシーンを削除し,変更後のシーンを取り込む.

ゲームタイプのデータも Yaml で管理されており,迷路ゲームを例にしたテキストデータを List 3 に示

す．List 3 中のキー「no_use_code」ではこのゲームタイプで使用しないコードピース名を指定する．これにより，画面右のピース選択画面から表示されなくなる．また迷路ゲームのようにステージ変更がある場合は，キー「stage_button」以降の情報を記述する必要がある．

List 3. Yaml description of game type “Maze”

```
name:迷路                    # Game name
no_use_code:[RobotMove, etc.]
                             # No use pieces
load_scenes: Maze            # Scene name
stage_button:true            # Show stage button
stage_names:[1, 2, 3, 4]  # Stage name
class: MasterManagerM
                            # Stage change class
method: ResetStage        # Stage change method
p_type: System.Int32      # Argument
```

4. 評価実験

本章では，「Puzzle Programming」を用いて行ったプログラミング体験授業(以下では本実験とする)について述べる．

4.1 実験環境

本実験の実験環境は下記に示す通りである．

- 実験日　2014 年 1 月 30 日
- 実験場所　同志社国際中学校・高等学校
- 被験者　高校 2 年生：27 名
  高校 3 年生：33 名
  ※　性別・文理混合
- 実験時間　2, 3 年生ともに 45 分×2 コマ
- 使用機材　iPad：13 台
- 使用ソフト　Puzzle Programming

4.2 実験の流れ

本実験の流れは下記に示す通りである．

1. 事前アンケート
2. 操作方法の説明
3. プログラミングの 3 大要素の説明
4. 共通課題
5. 応用課題
6. 事後アンケート

共通課題では，プログラミングの 3 大要素である「逐次実行」,「条件分岐」,「繰返し」が理解できているかを問う問題を出題した．ゲームタイプ「プログラミング」,「迷路ゲーム」で行う問題を，それぞれ 2 題ずつ出題した．

本実験は 2，3 人 1 組の班単位で行い，2 年生，3 年生ともに 11 班であった．

応用課題では，共通問題が解けた班を対象とし，あらかじめ作成しておいたプログラム例を生徒に見せ，その実行結果を予想する問題を出題した．

なお，2 年生の実験を先に行った際に，共通課題中の計算問題において，プログラミング以前の数学の段階でつまずく生徒がほとんどであったため，3 年生の実験ではその問題を省き，ゲームタイプ「ロボットゲーム」において「ロボットを正方形状に歩かせ続ける」という追加課題を出題した．

4.3 結果

4.3.1 事前アンケート

事前アンケートでは，本実験を受ける前の被験者のプログラミングやアプリケーション作成に対する意欲や経験を尋ねた．List 4 に事前アンケートの質問項目（抜粋）を，Table 2 に各質問の回答結果を示す．

Q1 より，ほとんどの生徒にとって本実験が初めてのプログラミング経験であったことが分かった．

Q2 では，プログラミングに興味のある生徒はほとんどいないのでは，と予想していたが，予想に反して「少しある」の解答が多かった．義務教育でプログラミング教育を行うことを想定すると，多くの生徒が興味を持てるような環境作りが必要である．

Q3 においてプログラミングと関係のあるアプリ制作の意欲について尋ねたところ，2 年生，3 年生

ともに過半数以上がアプリケーション開発に意欲があり，プログラミングを始めるきっかけがあれば挑戦する生徒も出てくるのでは，と考えた．

List 4. Questions of prior survey

Q1. あなたはこれまでにプログラミングを行った経験はありますか？
1. ある　2. ない

Q2.　現時点で，プログラミングに対して興味はありますか？
1. とてもある　2. 少しある　3. あまりない
4. 全くない　5. 分からない

Q3.　あなたは，スマートフォンやタブレット端末(iPad 等)で動くアプリを自分で作ってみたいですか？
1. とても作ってみたい　2. 少し作ってみたい
3. あまり興味がない　　4. 全く興味がない
5. 分からない

Table 2. Answers of Q1~Q3.

		1	2	3	4	5	No answer
Q1	2nd	3	24	×	×	×	0
	3rd	1	31	×	×	×	1
Q2	2nd	0	9	6	3	9	0
	3rd	2	11	11	4	5	0
Q3	2nd	9	7	4	1	6	0
	3rd	1	13	8	5	1	5

4.3.2 出題した課題

Table 3 に，課題に対して正解できた班数を示す．なお，予備知識のない高校生が完全に自力で課題を解くことは困難であったため，途中で適宜助言を与えながら実験を行っており，助言を受けて答えが分かった場合でも正解扱いにしている．

2 年生においては，共通課題 2 が難しかったようで，ここですべての班がつまずいてしまった．プログラミングが難しいというよりも問題そのものの数学的な理解が難しかったようだった．一方で，「簡単な問題のうちは操作もわかりやすいけれど，難しい問題になるとどう操作したらいいのか分からない．」という意見もあった．

3 年生においては，共通課題 2 をなくして「ロボットゲーム」を用いた追加課題を導入したことで，どの班もスムーズに課題を進めることができた．応用問題においては，問題 2 が解答できた班とできなかった班に分かれたが，理解不足というよりは時間が足りなかった点が大きいと考えられる．

Table 3. The number of correct answers.

Questions	2nd grade	3rd grade
Basic 1	11	11
Basic 2	3	×
Basic 3	0	11
Basic 4	0	11
Additional	×	11
Advanced 1	0	11
Advanced 2	0	5
Advanced 3	0	1

4.3.3 事後アンケート

事後アンケートでは，「Puzzle Programming」の UI(User Interface)の評価や本実験を受けた後のプログラミングに対する理解度や意欲を尋ねた．List 5 に事後アンケートの質問項目（抜粋）を，Table 4 に各質問の回答結果を示す．

Q1 と Q2 では UI の評価を尋ねた．Q1 では，2 年生においては「どちらとも言えない」の割合が少し高いものの，全体的に高評価であった．Q2 においてもおおむね高評価であり，総じて UI の評価は高かった．また，自由記述欄には「パズルの形だから，自分でも作業できそうって思えました．」，「当てはまらない場合はパズルのピースが合体しないなどすごく分かりやすくてよかったです．」といった声が多く，パズルを用いた工夫が奏功した．

Q3 ではプログラミングの三大要素の理解度を，Q4 ではプログラミングの文法規則を尋ねた．2 年生においては共通課題 2 でかなり苦戦していたこ

List 5. Questions of posterior survey

Q1. 今日の体験授業で使用したアプリ「Puzzle Programming」の操作方法は分かりやすかったですか？
1. とても分かりやすい 2. 分かりやすい
3. どちらとも言えない 4. 分かりにくい
5. とても分かりにくい

Q2.「Puzzle Programming」の見た目（ボタンの配置や色，文字など）はどうでしたか？
1. とても良い 2. 良い 3. 普通
4. 悪い 5. とても悪い

Q3.「Puzzle Programming」を使用してみて，プログラミングの三大要素『逐次実行』『条件分岐』『繰り返し』は理解できましたか？
1. とても理解できた 2. 理解できた
3. どちらとも言えない
4. あまり理解できなかった
5. まったく理解できなかった

Q4.「Puzzle Programming」を使用してみて，「while のあとには条件が必要である」といったプログラミングの規則は理解できましたか？
1. とても理解できた 2. 理解できた
3. どちらとも言えない
4. あまり理解できなかった
5. まったく理解できなかった

Q5. 今回の体験授業を受けてみて，パズルを使わない実際のプログラミングを自分でもやってみたいと思いますか？
1. とてもやってみたい 2. やってみたい
3. どちらとも言えない 4. あまりやりたくない
5. まったくやりたくない

Q6. 今回の体験授業をきっかけに，プログラミングに対する興味・関心は増しましたか？
1.とても増した 2.増した
3.どちらとも言えない 4.関心がなくなった
5. まったくやりたくない

Q7. 今回の体験授業で解いた問題のレベルはどうでしたか？
1. とても簡単 2. 簡単 3. ちょうどよい
4. 難しい 5. とても難しい

Table 4. Answers of Q1~Q7.

		1	2	3	4	5	No answer
Q1	2nd	2	11	10	3	0	1
	3rd	17	14	1	0	0	1
Q2	2nd	12	9	6	0	0	0
	3rd	10	17	5	0	0	1
Q3	2nd	1	13	8	5	0	0
	3rd	9	20	3	0	0	1
Q4	2nd	1	13	6	7	0	0
	3rd	10	17	4	1	0	1
Q5	2nd	1	7	9	9	1	0
	3rd	1	11	10	7	2	2
Q6	2nd	3	15	9	0	0	0
	3rd	8	20	3	1	0	1
Q7	2nd	0	0	0	22	5	0
	3rd	1	4	16	11	0	1

ともあり，難しく感じた生徒が多かったようである．3年生では，よりゲーム性の強い「ロボットゲーム」を導入したことで，理解度が高まったと考えられる．

Q5 では「Puzzle Programming」を足がかりに，本来のプログラミングに興味がわいたかを尋ねた．授業内で「実際のプログラミングは難しいけれど，『Puzzle Programming』は簡単です．」という説明をしてしまったため，本来のプログラミングに対して必要以上に苦手意識を与えてしまったと考えられる．また，実際のプログラミングとどのように対応しているのかを，もう少し分かりやすく説明する必要があったと考えられる．

Q6 においては，2 年生では 1，2 を合わせて全体の 66%，3 年生では 84%とどちらも 1，2 の割合が高く，被験者のプログラミングへの興味を高めること

に成功した.

Q7 では，2 年生においては全員が「4. 難しい」「5. とても難しい」のどちらかを答えており，2 年生には問題が難しすぎたと考えられる．3 年生は 2 年生と比べると「ちょうどよい」と答えた割合が高いが，33%の生徒が「4. 難しい」と答えており，高校生向けに問題レベルを再考する必要がある.

5. 考察と今後の課題

アプリケーションの UI の面では，事後アンケートでの評価はおおむね高く，教材としての使いやすさ，見やすさの観点においては一定の成果があったと言える．また，プログラミングにパズルを用いることによってユーザの興味を引きつけることにおいても，アンケートの結果から一定の成果があったといえるが，より踏み込んだプログラミングの技術やテクニック，およびプログラミングに不可欠な論理的思考力をユーザに身につけさせる工夫ができれば，教材としてより優れたものになると考えている.

また，本研究で行った実験では，被験者にとって難しすぎる問題を出すなど，1 つの完結した授業として実験を行う上で，授業としての完成度が低い，という反省点が挙げられる. 教材の開発を行う上では，どのように授業を行うのかをよく想定した上で，ユーザがより学びやすい環境作りを念頭に置く必要がある.

「Puzzle Programming」はプログラミング言語としての自由度が非常に低く，特に，再帰的な処理を行うことが現状ではできていない. あまり機能を拡張しすぎると初学者が取り組みにくくなる，という懸念はあるが，よくバランスを考えた上で機能の拡張を計ることも今後の課題である.

本研究を行うにあたり，同志社国際中学・高等学校のチャプレン・山本真司先生，および生徒の皆さんには，実験を行うにあたってご協力いただいた．ここに記して謝意を表する.

参考文献

1) 文部科学省，新学習指導要領・生きる力，http://www.mext.go.jp/a_menu/shotou/new-cs/ (参照 2014-02-06).
2) 産業競争力会議，成長戦略（素案），http://www.kantei.go.jp/jp/singi/keizaisaisei/skkkaigi/dai11/siryou1-1.pdf (参照 2014-02-06).
3) 藤本徹，シリアスゲームと次世代コンテンツ，http://anotherway.jp/seriousgamesjapan/archives/Fujimoto-SeriousGames.pdf (参照 2014-02-06).
4) MIT Media Lab Lifelong Kindergarten , Scratch , http://scratch.mit.edu (参照 2014-02-06).
5) Unity Technologies Japan, Unity, http://japan.unity3d.com/unity/ (参照 2014-02-06).
6) 高橋啓治郎，Unity 入門 高機能ゲームエンジンによるマルチプラットフォーム開発，(ソフトバンク クリエイティブ，東京，2011)，p.2.
7) Clark C. Evans , The Official YAML Web Site , http://www.yaml.org (参照 2014-02-06).