紙媒体の命令オブジェクトを用いた
タンジブルなプログラミング環境

2015年3月

201313118
彡田　一貴

筑波大学情報学群
知識情報・図書館学類

目 次

図 目 次

第1章　はじめに

本章では，研究の背景と目的，および本論文の構成について述べる．

1.1　背景

プログラミングは人間の意図した処理を行うようにコンピュータに指示を与える行為である．一般にはプログラムを書くことを職業とするプログラマがソフトウェア開発に用いるものであるが，近年では論理的思考の養成やコンピュータへの理解を深めることを目的に，情報教育の一環として広く扱われるようになった．文部科学省が発表した小中高校における学習指導要領の改善についての答申においてもプログラミング教育の導入とその必要性について述べられている [1].

そのため，専門的な知識がない初学者においても学びやすい，教えやすいプログラミング学習環境の整備が必要とされる．初学者に向けたプログラミング学習環境のアプローチとして，Graphical user interface(GUI) を用いて操作を簡略化したものが多い．例えば，Resnick らの Scratch[2] はマウスによるドラッグ&ドロップによる命令の組み合わせによってプログラミングができる環境を提案した．一方で早期の入門的なプログラミング学習においては GUI を用いた手法よりも Tangible user interface(TUI) を用いた手法の方が学習効果が高いという結果が Horn らによって明らかとなった [3]．TUI を用いたプログラミング環境の研究の例として，Horn らは木製のブロックを命令オブジェクトとし，これを組み合わせることにより積み木遊びのような感覚でプログラミングが可能な環境を提案した [4].

しかし，TUI を用いる手法においては大量のセンサや特殊なハードウェア構成を必要とするため，環境整備が複雑で高コストになる．教育や学習を容易にするために TUI を用いる環境の提案が多くなされているが，その環境整備が容易ではないというのが新しいプログラミング環境提案における現状である．

1.2　提案と目的

本研究では，紙媒体の命令オブジェクトを用いることにより，手軽かつ安価に環境整備が可能な TUI を用いたプログラミング環境を示す．本手法の特徴は印刷することによって作成可能な紙媒体の命令オブジェクトを web カメラによって撮影，認識することによって，紙を並べることによるプログラミングを実現する点にある．

また，プログラミングとして必要な基本的なシンタックスを備えた上で，タンジブルならではの命令が組み込めることや，紙に対する書き込み等のインタラクションが可能な環境を目指した．本手法の環境を実現し，環境整備が容易であることと，初学者に対し一定の学習効果があることを示すことが本研究の目的である．

1.3　本論文の構成

第 1 章以降の本論文の構成は次のとおりである．第 2 章では，関連研究を示すことにより本研究の位置づけを明らかにする．第 3 章では，作成したプログラミング環境の概要，構成，使用方法について述べる．第 4 章では，作成したプログラミング環境でプログラミングを行うためのシンタックスやをプログラム例，学習例を述べる．第 5 章では，本手法に実装されている特徴的な機能や，プログラミング環境デザインについて述べる．第 6 章では，作成したプログラミング環境を用いた評価実験の手順と結果について述べる．第 7 章では，本手法の適用範囲，問題点，応用について議論する．第 8 章では，本研究の結論を述べる．

第2章　関連研究

本章では，本研究の関連研究および位置付けを述べる．はじめに，プログラミング支援を行う研究を述べる．次に TUI を用いることにより，操作方法を簡略化したプログラミング環境研究を述べる．最後に紙を用いた創造ツール研究について述べる．

2.1　プログラミング支援

今日までに，様々な手法を用いてプログラミング学習の支援が試みられている．HMMMML[5] では，プログラミングに対して苦手意識を持つ学生の学習モチベーションを高めることを目的に，複雑な構文を一切取り除き，音声合成や web を利用した検索，MIDI 音の出力といった処理を 1 命令で実行可能にした新言語をデザインした．また HMMMML2[6] や HMMMML3[7] は，プログラミングにおけるソースコードのスペルミスや宣言抜けなどを許容し，半端なコードであっても解釈を行い，コンパイルエラーが極力起きない環境を示した．Nigari[8] はプログラムを自動的に可視化することで，プログラムの流れの直感的理解を促した．加藤ら [9] はプログラミングに時間軸の概念を取り入れ，過去のソースコードの状態と実行結果を視覚的に表示することでエラーやコードの入力ミスを少なくする手法を示した．また佐藤ら [10] は統合開発環境 Eclipse 上で連動する，GUI プログラム操作に対するプログラム実行部分の可視化システムを示した．Scratch[2] はマウスによる操作を主とし，画面上に表示される命令ブロックのドラッグ&ドロップにより簡易的なプログラミングができる環境である．VISCUIT[11] ではメガネと呼ばれる命令オブジェクトに対して絵や数字を描き込み，その組み合わせによって様々な命令を作ることができる．

これらの研究は文字入力またはマウスによる GUI 操作を主としたプログラミングにおける学習支援を試みた研究である．本研究では実世界における命令オブジェクトを操作に用いたプログラミング環境において，少ない命令で描画や音の発生，変数の管理が行える言語をデザインした．また，プログラム実行順序の可視化や命令オブジェクトによるプログラムをソースコードに変換する機能により，学習を支援する．

2.2　TUI を用いたプログラミング環境

プログラミング学習の導入を容易にするために，TUI を操作手法として取り入れている環境はこれまでにいくつか提案されている．E-Block[12] や T-Maze[13] は中に LED やセンサ，アクチュエータが入ったブロックを並べることで，LED 制御やセンシングを取り入れたプログラムが組める環境を示した．tactusLogic[14] や Tern[4] は木製の命令ブロックを組み合わせることでプログラミングが可能な環境を示した．Turtan[15] はプラスチック製のブロックをテーブルトップインタフェース上に配置し，操作することで様々なビジュアライジングが可能な環境を示した．

これらの環境では，プログラムに使用するオブジェクトの認識にセンサや特別なマーカを用いている．そのためオブジェクトそのものが高価で複雑な場合が多い．一方本研究が示す環境は，紙を命令オブジェクトとして採用することにより低コストかつ容易に学習環境の整備が可能である．

2.3 紙を用いた創造ツール

通常複雑な装置を用いる開発や学習において，紙を用いることで環境を低コスト化する様々な試みがこれまでになされてきた．Sketching in Circuits[16] では，導電性のあるテープを用いることにより電子回路を紙の上に実現する手法を示した．Robert[17] は本にシール状の命令オブジェクトを貼付けることによって物語の展開をプログラムする環境である．

これらの研究は紙を創造的環境の一部に使用することによって，環境の低コスト化や整備の容易さに貢献出来ることを示している．

第3章　Sheets:紙媒体の命令オブジェクトを用いたプログラミング環境

本章では，紙媒体の命令オブジェクトを用いたプログラミング環境である Sheets について述べる．Sheets の概要を述べた後に，システム構成，実装について述べる．

3.1　概要

Sheets は，紙媒体の命令オブジェクトを並べて組み合わせることで，プログラミングを行うことができる初学者向けの学習環境である．命令オブジェクトとして紙媒体を使用することで，プログラミング学習の過程に紙に対する手書き入力や自由なスケッチを行うといったアナログな動作を組み込むことを実現した．Sheets は学習支援として，命令実行順序の可視化機能や，現存するプログラミング言語への変換機能をもつ．Sheets によって作られるプログラムは基本的に逐次実行によって動作し，図形やイラストの描画，音の発生，図形やイラストの移動，繰り返し，条件分岐，変数，実世界イベントなどの構文が用意されている．

3.2　システム構成

実装した環境の全体図を図 3.1 に，システム構成を図 3.2 に示す．本環境は，プログラムを作るために使用する紙媒体の命令オブジェクト，並べられた命令オブジェクトを認識するための web カメラ，命令オブジェクトの意味を解釈し実行結果の表示を行うためのコンピュータ及びソフトウェアからなる．

以下の節では使用したハードウェア及び実装したソフトウェアの詳細を述べる．

図 3.1: システムの全体像

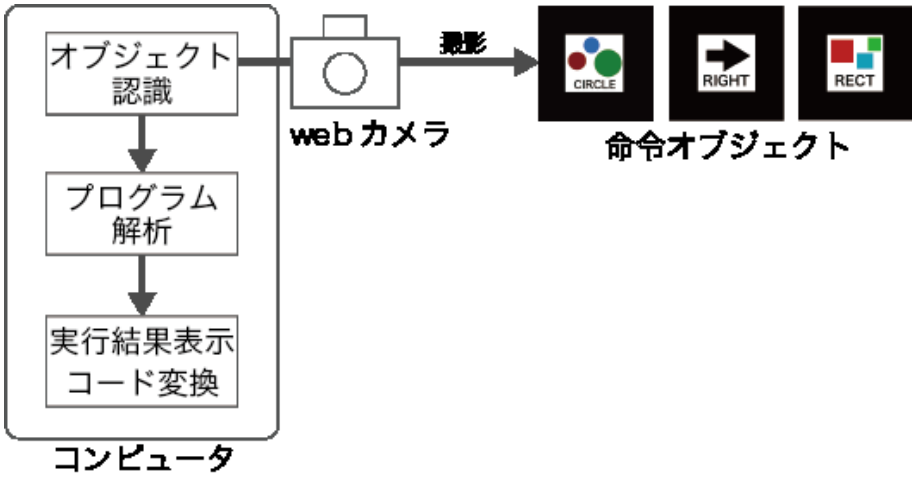


図 3.2: システム構成

3.3 ハードウェア

3.3.1 命令オブジェクト

本システムにおいて用いる命令オブジェクトは紙によって作られる．命令オブジェクトは図 3.3 に示すように，縦 64mm 横 40mm であり，中に一辺 38mm 正方形の AR マーカーが描かれている．図 3.4 に実際に使用する命令オブジェトの例を示す．マーカーの下部には命令の説明が記述されている．本システムでは，この命令オブジェクトを組み合わせることによりプログラムを作り上げることができる．

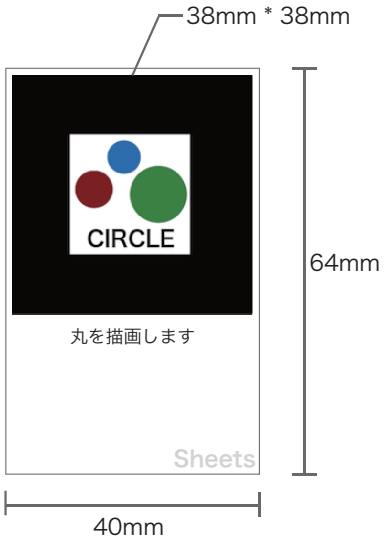


図 3.3: 命令オブジェクトの概要

図 3.4: 命令オブジェクトの例

3.3.2 web カメラ

web カメラは USB によってコンピュータと接続し，実世界の映像をソフトウェアに取り込む．本システムにおいて web カメラは，並べられた命令オブジェクトを撮影するために使用される．本研究における被験者実験やシステム展示においては，撮影を容易にするために IPEVO 社のスタンド型の web カメラ ZiggiHD[1] を用いた．なお，一般的なノート型コンピュータ等にあらかじめ搭載されている web カメラを用いても動作可能である．

3.3.3 コンピュータ

web カメラからの映像を解析して並べられた命令オブジェクトを認識する処理，及びユーザによって作られたプログラムの実行結果を出力する処理を行うコンピュータとして，Apple 社の MacBook Pro[2](CPU:Intel Core i7 2.9GHz, RAM:8GB) を使用した．

[1]ZiggiHD http://www.latex-cmd.com/struct/footnote.html
[2]Macbook Pro http://www.apple.com/jp/macbook-pro/

3.4　ソフトウェア

ソフトウェアを ActionScript3.0 言語を用いて作成した．これは，命令オブジェクトの認識，解釈，実行結果の描画，及び命令オブジェクトによるプログラムをソースコードに変換する処理によって構成される．ソフトウェアの実行画面を図 3.5 に示す．

図 3.5: ソフトウェア実行画面

3.4.1　命令オブジェクトの認識

命令オブジェクトには AR マーカーが描かれているため，これを認識することでどの命令オブジェクトが置かれているかを判断することができる．AR マーカーとは，正方形の黒枠で囲われた中に何かしらのパターンが描かれた画像である．パターン画像を数値化して，パターンファイルとしてあらかじめソフトウェアに登録することにより，web カメラによって撮影される画像の中に登録されたパターンが存在するかどうかを判断することができる．本システムにおいて用いた AR マーカーは解像度を 16*16 の 256 分割とした．

ソフトウェアにおいて登録されるパターン画像は，任意の ID とカメラ空間上における位置座標を持つため，これらの情報を使うことで，実世界でどのような命令オブジェクトがどのように並べられているかを把握することができる．AR マーカーの認識には ActionScript3.0 の AR ライブラリである FLARToolkit[3] を用いた．

また，本手法では，マーカーに対する塗り潰しを行うことで命令が変化することがある．これはあらかじめ塗り潰された結果のマーカーパターンの画像を登録することで実現した．

[3]FLARToolkit http://www.libspark.org/wiki/saqoosha/FLARToolKit

3.4.2　命令オブジェクトの解釈

FLARToolklit によって，web カメラで撮影された領域にどのような命令オブジェクトがどのような位置にあるかを認識した．用意されている一つ一つの命令オブジェクトに対し，その命令内容通りの処理を行うクラスが定義されており (図 3.6)，命令オブジェクトが認識されると対応するクラスのインスタンスが生成される．命令に対応したインスタンスは配列に格納される．認識された命令オブジェクトはユーザ視点から向かって左から順に実行されるため，置かれた位置の x 座標を基準に昇順ソートされる (図 3.7)．実行が行われる際には，配列の先頭から順番に対応した処理を行うメソッドが呼び出される．

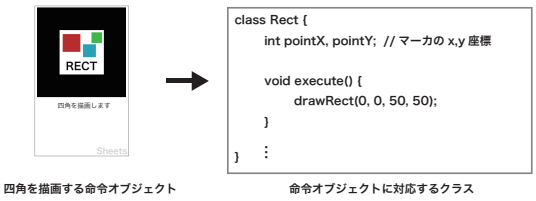


図 3.6: 命令オブジェクトと処理クラスの対応例

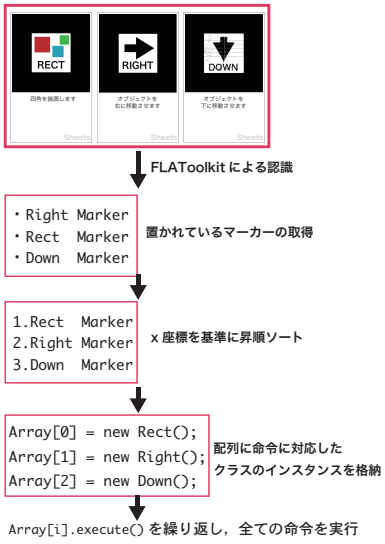


図 3.7: 命令オブジェクトが実行されるまでの処理

3.4.3　実行結果の描画

命令オブジェクトによって作られたプログラムの実行結果は図 3.5 に示した実行結果画面部に表示される．描画命令に対応した結果は，四角や丸などの図形が実際に描画される．変数命令に対応した結果は，実行結果画面下部に値として表示される．命令はユーザ視点から向かって左から順に逐次実行されるため，描画命令は置く順番によって実行結果の見た目が異なることがある．図 3.8 は実行順序によって表示結果の見た目が変化する例である．

3.4.4　ソースコードへの変換

ユーザが命令オブジェクトを並べて作るプログラムは，実際に存在するプログラミング言語のソースコードにリアルタイムに変換して表示される．変換される言語は，描画やサウンドの制御が容易に行える Processing 言語 [4] を用いた．

[4]Processing https://processing.org/

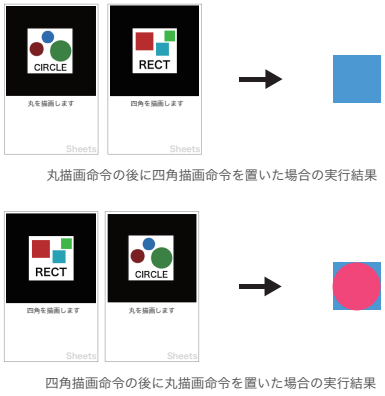


図 3.8: 命令オブジェクトの順序と実行結果例

命令オブジェクト対応するクラス定義内に，その命令相当の処理を行う Processing のソースコードを文字列として保持することにより，命令オブジェクトによるプログラムに対応するソースコードの出力を実現している．例えば，四角を描画する命令オブジェクトに対応するクラスでは，Processing にて四角を描画する命令である rect(x,y,w,h); という文字列を保持している．並べられたそれぞれの命令オブジェクトに対応するクラスに書かれている Processing のソースコードを繋ぎ合わせることで，擬似的なソースコード変換を行っている．

3.4.5 プログラムの実行

ユーザが並べた命令オブジェクトの実行は，リアルタイム形式もしくはステップ形式で実行される．実行形式の切り替えは図 3.5 に示した実行モード切り替えボタンによって行われる．

リアルタイム形式では，命令オブジェクトを置き，認識された瞬間に命令オブジェクトの解釈を行い，実行結果の描画やソースコードへの変換が行われる．ステップ形式では，作りたいプログラムに必要な全ての命令オブジェクトを並べたあとに画面をクリックすることにより，先頭から順に 1 秒程度の時間をかけながら実行される．ステップ形式においては，命令オブジェクトが実行されるタイミングでハイライトされるため，プログラムの実行順序を視覚的に確認することができる．

3.5 使用手順

本システムを用いてTUIなプログラミング学習を行う際の使用手順は以下の通りである.

1. 命令オブジェクトを印刷[5]し，切り取りを行う.
2. ソフトウェアを立ち上げ[6]，使用するwebカメラを選択する.
3. 命令オブジェクトを並べてプログラムを作り上げる.
4. 実行形式を選択し，プログラムの実行を行う.
5. 実行結果やソースコードを確認し，命令オブジェクトの組み合わせを変更するなど，プログラミングを繰り返す.

[5]Sheetsで用いる命令オブジェクトはwebで公開している
http://www.iplab.cs.tsukuba.ac.jp/~tada/Sheets/sheets_command.png

[6]Sheetsで用いるソフトウェアはwebで動作する
http://www.iplab.cs.tsukuba.ac.jp/~tada/Sheets/

第4章　シンタックスとプログラミング学習例

本章では，Sheets において実装されている命令とその使用例について述べる．また，複数の命令を組み合わせた学習例を示す．

4.1　Sheets の命令群

Sheets は基本的に図形の描画とその操作を組み合わせることがプログラミングを行う．そのため，各種図形を描画する命令である「描画系命令」，任意の図形を移動させる「移動系命令」，サウンドを発生させる「音系命令」，変数を操作する「変数系命令」，繰り返しや条件分岐といった基本的なシンタックスを記述するための「構文系命令」，実世界のイベントを扱う「イベント系命令」が用意されている．

以下の節では各命令の内容と使用方法について述べる．

4.1.1　描画系命令

図 4.1: 描画を行う命令オブジェクト

「丸描画命令」と「四角描画命令」は，命令が実行されるとそれぞれ丸と四角の描画を行う．図形の色は認識される度にランダムで変わる．「イラスト描画命令」は，ユーザが命令オブジェクト上に描く任意の図を描画することができる．図 4.1 に示されているイラスト描画命令オブジェクトの破線内に書き込んだ絵を切り抜いたものが，描画内容として現れる．任意のイラストを描画している様子を図 4.2 に示す．

図 4.2: 任意のイラストを描画するイラスト描画命令の例

4.1.2 移動系命令

移動系命令は，「右移動命令」,「下移動命令」,「ランダム移動命令」の 3 種類が用意されている．各命令を実行するための命令オブジェクトを図 4.3 に示す．

図 4.3: 移動を行う命令オブジェクト

「右移動命令」と「下移動命令」は，命令が実行されると任意の図形の移動を行う.「右移動命令」では図形の x 座標が増加し,「下移動命令」では図形の y 座標が増加する.「ランダム移動命令」は，命令が実行されると任意のオブジェクトを x 座標 y 座標共にランダムな位置に移動させる.

移動系命令の対象となる図形は，直前に実行された描画系命令によって描画されたものである．例えば,「四角描画命令」のあとに「右移動命令」を並べると，四角形が描画され，それが右に移動するが,「四角描画命令」「丸描画命令」「右移動命令」のように並べると，四角形，丸が順に描画された後に最後に描画された図形である丸が右に移動する.

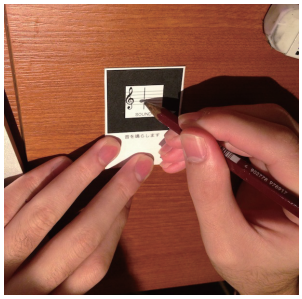
4.1.3 音系命令

音系命令は，任意の単音を再生する「単音再生命令」が用意されている．命令を実行するための命令オブジェクトは図 4.4 に示すように 1 つであるが，命令オブジェクトに記されている五線譜に音の種類を表す音符を書き込むことで任意の単音を再生することができる．

単音再生命令

図 4.4: 単音再生を行う命令オブジェクト

単音は 1 オクターブ分用意されている．命令オブジェクトへの書き込みによって再生される単音を変更する様子と，単音の例を図 4.5 に示す．


命令オブジェクトへの書き込み

ソの音を再生

レの音を再生

ミの音を再生

図 4.5: 命令オブジェクトに書き込みを行い，再生する音を変化させる例

4.1.4　変数系命令

Sheets では 3 つの変数が用意されている．条件分岐などで値として使用する「変数 A」,「変数 B」,「変数 C」，各変数の値を増減させる「変数 A 増減命令」,「変数 B 増減命令」,「変数 C 増減命令」を用いて変数を操作する．各命令を実行する命令オブジェクトを図 4.6 に示す．

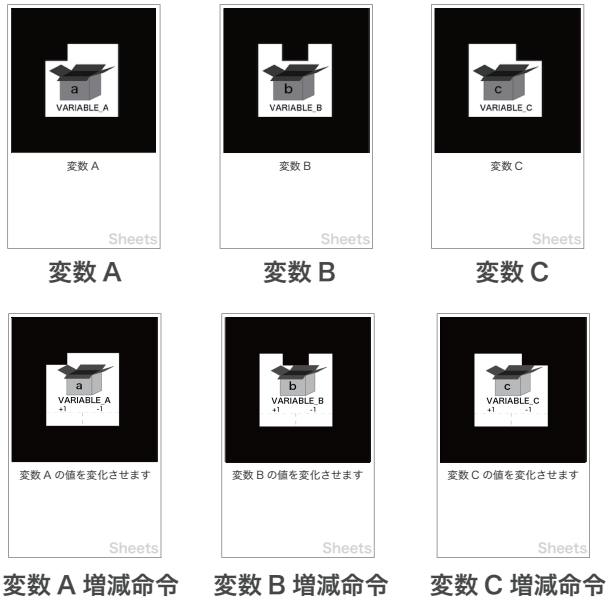


図 4.6: 変数系処理を行う命令オブジェクト

「変数 A」,「変数 B」,「変数 C」は，命令が実行されるとそれぞれ現在の値が実行結果画面に反映される．「変数 A 増減命令」,「変数 B 増減命令」,「変数 C 増減命令」は，命令が実行されるとそれぞれの値をインクリメントもしくはデクリメントすることができる．増減の切り替えは，命令オブジェクトの所定の部分を塗り潰すことによって行うことができる．zikkou「変数 A 増減命令」の命令オブジェクトに対して書き込みを行い，値の増減を行う例を図 4.7 に示す．

図 4.7: 変数増減命令の例

4.1.5　構文系命令

Sheets はプログラミング言語としての基本的なシンタックスとして，逐次実行を行うほかに，繰り返し構文と条件分岐構文を記述することができる．各構文を記述するための命令オブジェクトを図 4.8 に示す．

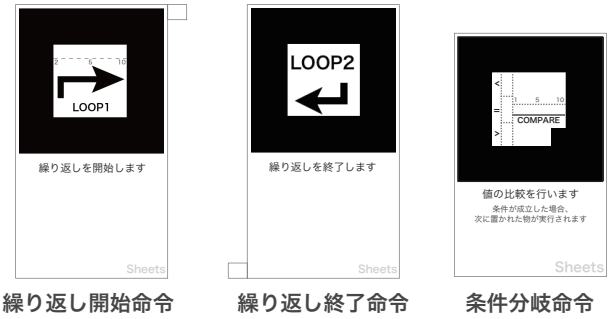


図 4.8: 繰り返しおよび条件分岐を行う命令オブジェクト

繰り返し構文

繰り返し構文は,「繰り返し開始命令」と「繰り返し終了命令」を用いて記述を行う．繰り返したい命令群をこの 2 つの命令で挟むことにより，挟まれた命令群が指定回数繰り返し実行される．繰り返しの回数は,「繰り返し開始命令」の命令オブジェクト上部の塗り潰し具合によって決まる．塗り潰しが行われなかった場合は 2 回，最大まで塗り潰しが行われた場合は 10 回となり，2-10 の間で繰り返し回数を指定できる．塗り潰しによる繰り返し回数の指定と，繰り返し文を使ったプログラムの例を図 4.9 に示す．

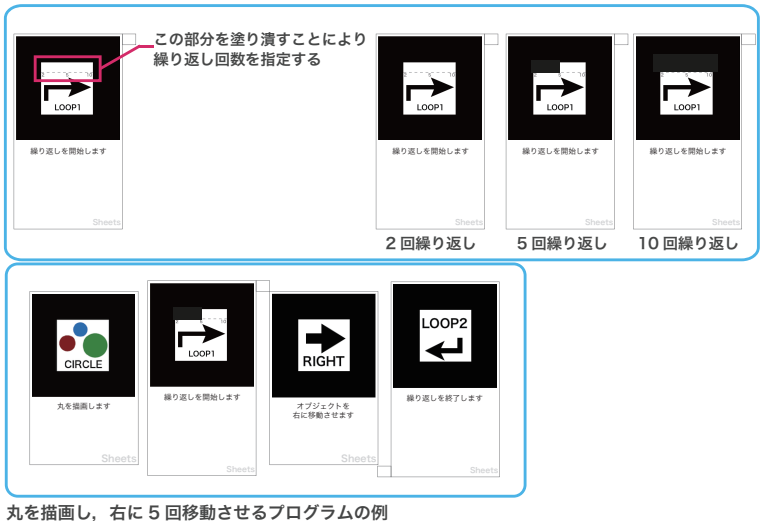


図 4.9: 繰り返し回数指定と繰り返しを使ったプログラム例

条件分岐構文

条件分岐構文は，変数と組み合わせて値の比較を行うことができる．比較できる内容は，値が大きいか，等しいか，小さいかであり，比較出来る値は 1-10 である．比較条件の決定には，「条件分岐命令」オブジェクトに対する書き込みを用いる．ユーザはまず，＞，＝，＜のいずれかに対応する部分を塗り潰す．次に比較対象となる値を，1-10 の範囲で塗り潰して決める．

条件が成立した際には，「条件分岐命令」の直後に置かれた命令が実行される．条件が不成立の場合には，「条件分岐命令」の直後に置かれた命令は実行されず，スキップされる．そのため，Sheets において条件分岐を行うプログラムを組み上げるには，まず「条件分岐命令」の前に比較対象となる「変数」を置き，条件が成立した際に実行される命令を「条件分岐命令」の後ろに置くことになる．「条件分岐命令」を使ったプログラムの例を図 4.10 に示す．

図 4.10: 繰り返し回数指定と繰り返しを使ったプログラム例

4.1.6　イベント系命令

Sheets は基本的に逐次実行型のプログラミング環境であるが，イベント駆動型の処理を組み込むことも可能である．イベント系命令として,「距離イベント」と「タッチイベント」の 2 種類が用意されている．これらのイベント系命令は，リアルタイム実行形式の際に使用可能である．

各命令を記述するための命令オブジェクトを図 4.11 に示す．

図 4.11: イベント系処理を行う命令オブジェクト

距離イベント

距離イベントは,「距離イベント開始点」オブジェクトと「距離イベント終了点」オブジェクト間の物理的な距離を測定し,「距離取得命令」によって値として使用できるものである．これにより，描画した図形を開始点-終了点オブジェクト間の距離に応じて移動させるといったプログラムが作成可能である (図 4.12).

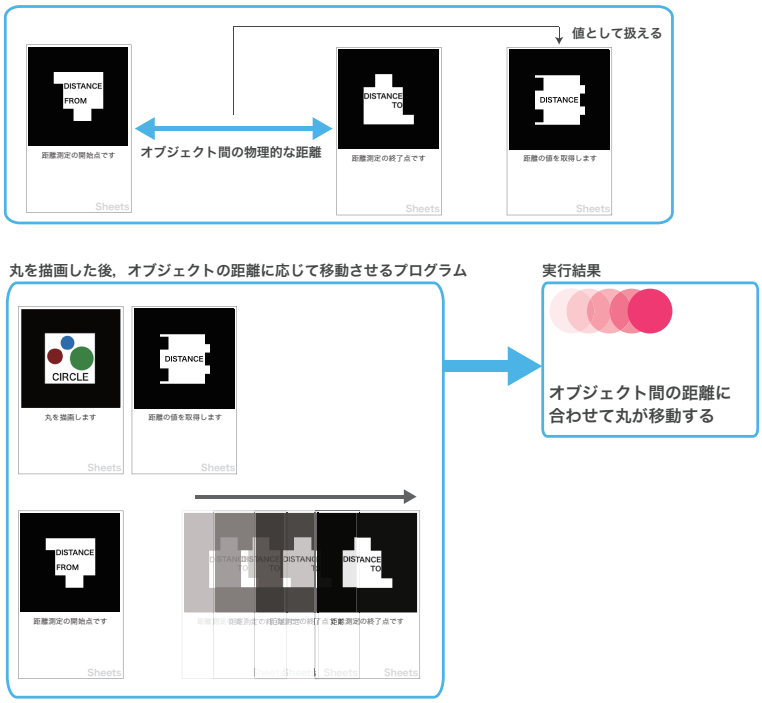


図 4.12: 距離イベント命令の使用例

タッチイベント

タッチイベントは,「タッチイベント命令」オブジェクトに対して指などで触れると,「タッチイベント命令」の次に置かれた命令が実行されるものである．これにより，命令オブジェクトにタッチすると描画内容が変わるといったプログラムが作成可能である (図 4.13).

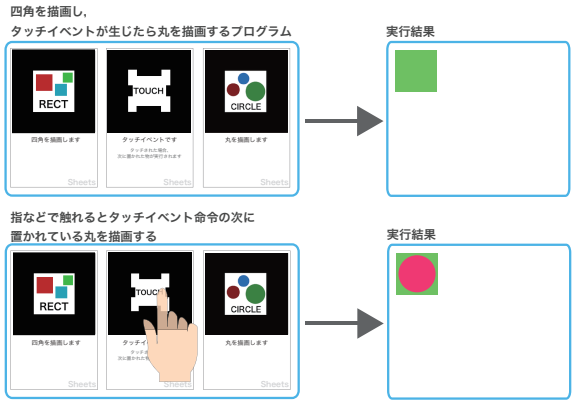


図 4.13: タッチイベント命令の使用例

4.2　プログラミング学習例

本章で述べた命令を組み合わせたプログラムを作ることで，逐次実行や繰り返し，条件分岐，変数といったプログラミングに必要な基本概念を学ぶことができる．例えば，描画と移動，繰り返しを組み合わせて,「丸が徐々に移動するアニメーション」を作ることができる．さらに変数と条件分岐を追加することで,「変数 A の値が 5 を超えたら丸を描画する」といった動きを作ることもできる (図 4.14)．図 4.14 のように，数枚の命令オブジェクトを並べることによって一般的なプログラミングのように繰り返しや条件分岐，変数を扱うことができる．

図 4.14: プログラミング学習例

さらにイベント命令を使うことで，タッチ入力があれば丸をランダムな位置に移動させるといった動きを作ることができる．

第5章　特徴的な機能と学習支援

本章では，Sheets に実装されている機能について述べる．Sheets は初学者にむけたプログラミング学習環境であるため，初学者の理解を援助するための仕組みをいくつか用意した．それぞれの仕組みがどのようなもので，なぜ実装され，学習の過程においてどのように役立つものであるかを以下の節で述べる．

5.1　実行順ハイライト機能

Sheets は，並べられた命令オブジェクトに対しハイライトを行い，どの部分を実行して出力結果に至ったかを可視化する．4 章で述べた通り，Sheets には逐次実行のほか，繰り返しや条件分岐が可能なため，命令の実行順序が変化することがある．一般的なプログラミングにおいても，if 文や loop,while 文による実行順序の分岐や変化は初学者にとって複雑であり，フローチャート等によって命令を書き下し，どのような実行順になるかをひとつずつ追うことで理解を試みることが多い．このような学習における躓きを撤廃するために，Sheets では実行時に命令オブジェクトをハイライトする手法を用いた．

この機能はステップ実行形式の際に動作する．ステップ実行形式では，1 つの命令につき 1 秒程度の時間をかけながらゆっくり実行するため，実行された命令と出力結果の対応を視覚的に確認することができる．ハイライトは，実行対象の命令オブジェクトを赤に着色することで行われる．繰り返し命令に入った場合には，繰り返し開始と終了命令を緑に着色し，繰り返し対象となる命令オブジェクト群を囲むことで，どの部分を繰り返し実行するのかを明確にする．簡単なプログラムを使ったハイライト例を図 5.1 に示す．

図左側の数字は実行順番を表す．

また，実際の使用時にハイライトが行われている例を図 5.2 に示す．Sheets は，命令のハイライトに加え，ソフトウェア上部に現在実行している命令内容の説明を提示することによって，どのような命令がどのような順で行われることで今の実行結果になったのかというプログラミングを理解する上で必要な読解力を身につけるための支援を行う．

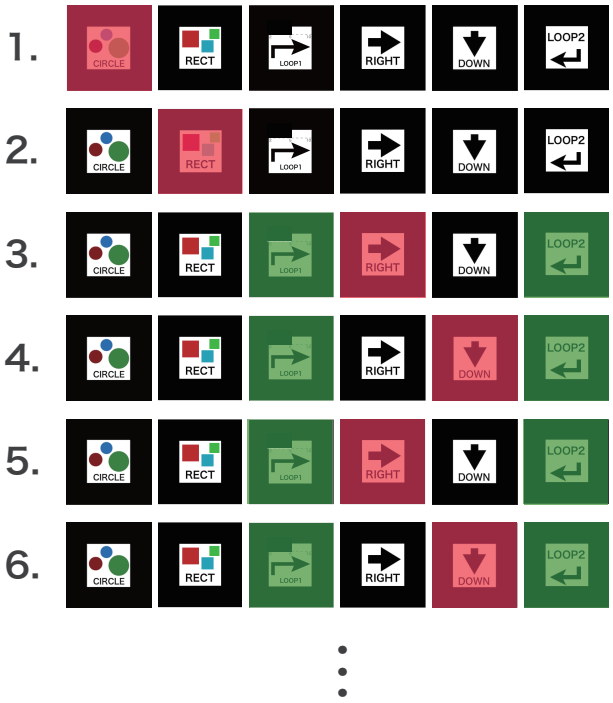


図 5.1: ハイライト例

図 5.2: ハイライト実行例

5.2　ソースコードへの変換機能

Sheetsは，紙媒体の命令オブジェクトを並べて作られたタンジブルなプログラムを，現存するプログラミング言語のソースコードへリアルタイムで変換する機能を持つ．ソースコードへの変換結果は図3.5示した実行画面右下部分に表示される．変換対象となる言語は，Sheetsにおいて作成可能な図形の描画や移動，サウンドなどの表現を短いコードで容易に記述できるものであるという観点から，Processing言語を採用した．

Sheetsのような初学者の学習導入に用いられるタンジブルなプログラミング環境は関連研究で述べた通り，これまでにもいくつかの提案がなされてきた．しかし，ほとんどの環境の学習到達目標が，タンジブルなプログラミングによってプログラミングの雰囲気や特徴を掴むまでにとどまっており，ユーザが次なるステップとして実際のプログラミングを扱うための学習の橋渡しが行われていない．そこで，Sheetsではタンジブルなプログラミングによって作られたプログラム相当のソースコードを提示することによって，描画などの制御や，繰り返し，条件分岐の記述方法などの理解援助を試みた．

出力されるソースコードはProcessing環境にそのままコピー&ペーストすることで実行可能であるため，繰り返しの回数や丸の半径といった簡単な部分の変更をきっかけに，実際のプログラミング言語に触れることができる．

5.3　紙媒体を使ったインタラクション

Sheetsは，紙を命令オブジェクトとして採用することで学習環境整備の容易さを実現した．しかし，準備が容易であるという理由のみで紙媒体を用いたわけではない．Sheetsでは紙を使ったならではのインタクションをプログラミングの過程に取り込んでいる．4章で述べた通り，命令オブジェクトへの書き込みによって，繰り返しの回数や条件分岐の内容を切り替えることができる．また，ユーザが描く自由なものをプログラムによって動かすこともできる．また，命令オブジェクトに描かれたARマーカーを認識に用いているため，マーカーの位置座標や認識の有無をwebカメラによる画像処理のみで行うことができた．そのため，特別なセンサ等を用いることなく，距離イベントやタッチイベントという実世界入力を取り込むことが可能となった．

プログラミングとは通常，どのような命令を実行させるにおいても，ユーザはキーボードによる命令入力を行うだけであり，動きに変化がない．また命令の入力から出力まで，全ての動作が画面の中で行われるデジタルなものである．この無機質な完全デジタル環境こそが，プログラミング初学者が感じる嫌悪感や難しさに繋がっているのではないかと考えたため，Sheetsでは先述したようなアナログな動作を多く取り入れた．Sheetsでは，プログラミングの過程において普段親しみのあるペンを使った書き込みなどの動作を取り入れ，プログラミングをより身近に感じられるように試みた．

第6章　評価実験

本章では，提案システムではあるSheetsを用いた評価実験について述べる．以下の節より，実験の概要，被験者の詳細，実験手順，結果，考察の順に述べる．

6.1　実験概要

実験対象となる被験者にSheetsによるプログラミングを体験してもらい，完遂時間やエラー率を計測すると共に実験前後にアンケート調査を行った．実験は，Sheetsを用いて提示された問題を解決するプログラムを組むものである．実験前のアンケートではプログラミングへの興味関心，経験について回答してもらい，実験後のアンケートではシステムに対する感想や既存のプログラミング環境との違いなどについての回答を依頼した．

実験の目的は，Sheetsのシステムとしての使いやすさや学習効果の測定，プログラミング経験者と未経験者で使用感や解答時間に違いがどの程度生まれるかの調査である．

6.2　被験者の詳細

被験者は１９歳から２２歳の合計８名 (女性２名) であり，全員が筑波大学の学生である．プログラミングを日常から使用する情報系の学生が４名，授業等で経験がある程度の学生が３名，プログラミング経験が全くない学生が１名であった．また，１名の学生がSheetsのような文字入力以外の方法によるプログラム環境を扱う経験があった．

6.3　実験手順

実験を開始する前に，まず被験者に対して研究の目的，実験方法などの事前説明を行った．次に，環境の使い方やシンタックス，各命令オブジェクトの内容，どのようなプログラムが組めるかといった，Sheetsというプログラミング環境の基本的な仕様について口頭説明を行った．その後，環境に慣れるための例題を３つ提示し，回答を依頼した．例題として扱われた内容は以下の通りである．例題では，回答に必要になる命令オブジェクトをあらかじめ被験者に与えた．

- 例題１：丸，四角を描画し，四角を右に移動させ，最後に音を再生するプログラム
- 例題２：丸を描画した後，左下に１０回移動させるプログラム
- 例題３：変数Ａを１０にするようなプログラム

例題を終えた後，以下に示す問題を出し，問題を解決するプログラムを作成するよう指示した．この際，回答に使用する命令オブジェクトは，被験者自身が全ての命令オブジェクトの中から自分で選択を行った．

- 問題１：丸を描画し，１０回右に移動させつつ，四角をランダムに発生させるプログラム
- 問題２：プログラム終了後に，変数Ａが１０，変数Ｂが５になっているプログラム

問題１は，Sheets における基本的な逐次実行の理解と，繰り返し構文が扱えるかどうかを問う問題である．問題２は，繰り返し構文と条件分岐構文を組み合わせたプログラムが組めるかどうかを問う問題である．

最後に，Processing 言語によって書かれたプログラムを見せ，そのプログラム内容の実行結果と同等の動きを行うものを Sheets によって作るという問題を提示した．問題として提示した Processing によって書かれたプログラムは以下の通りである．

code 6.1: ソースコード問題１

```
1 // 四角を１０回描画するプログラム
2 for (i = 0; i < 10; i++) {
3     rect(x,y,50,50);
4 }
```

code 6.2: ソースコード問題２

```
1 // 変数A が 5 より大きければ丸を描画するプログラム
2 for (i = 0; i < 10; i++) {
3     a++;
4     if ( a > 5 ) ellipse(x,y,50,50);
5 }
```

ソースコード問題１は，for 文を読み解くことができるかを問うものである．ソースコード問題２は，for 文と if 文の組み合わせを理解し，正しい条件文が作成出来るかを問うものである．

以上の問題に回答してもらった後，被験者が自由にプログラムを組む時間を取り，どのようなプログラムをどのように作成するかを観察した．

6.4 実験結果

6.4.1 回答時間とミス回数

実験結果を表 6.1 に示す．表 6.1 は，各被験者のプログラミング経験と，各問題における回答時間，回答ミス回数をまとめたものである．

表 6.1: 評価実験における各被験者の問題に対する回答時間 (sec) とミス回数 (回)

被験者番号	プログラミング経験	問題 1	問題 2	ソース 1	ソース 2
1	普段から使用	60	60	10	60
2	授業程度	60 ミス 1	110 ミス 1	80	120
3	未経験	120 ミス 1	350 ミス 1	180	120 ミス 1
4	普段から使用	60	60	20	40
5	普段から使用	105 ミス 1	60	20	50
6	授業程度	120	90	20	60
7	授業程度	180 ミス 1	240 ミス 2	20	50
8	普段から使用	60	60	15	50

被験者全員を対象とした場合，問題 1 の平均回答時間 (sec) は 92.63(SD=41.03)，問題 2 の平均回答時間は 128.75(SD=101.29)，ソースコード問題 1 の平均回答時間は 45.63(SD=54.85)，ソースコード問題 2 の平均回答時間は 68.75(SD=30.18) であった．

また，被験者をプログラミング経験度で分類した場合の各問題の平均回答時間 (sec) は表 6.2 に示す通りである．

表 6.2: 被験者のプログラミング経験度と問題に対する回答時間 (sec)

経験度	問題 1	問題 2	ソース 1	ソース 2
普段から使用	71.25 (SD=19.48)	60 (SD=0)	16.25 (SD=4.15)	50 (SD=7.07)
授業程度	120 (SD=48)	126.67 (SD=66.50)	40 (SD=28.28)	76.67 (SD=30.91)
未経験	120 (SD=0)	350 (SD=0)	180 (SD=0)	120 (SD=0)

表 6.2 に示す通り，プログラミングの経験度によって各問題の回答時間に大きな差が出た．プログラミング経験が豊富なほど，問題に対し早く回答を行った．

6.4.2 実験後のアンケート結果

実験後に被験者に対して行った口頭アンケートの内容とそれに対する主な回答を以下に示す.

- **システムを使ってみた感想について**
  - 自由度があって面白かった
  - ハイライトが分かりやすくて良い
  - 初学者はこれで十分かもしれないが慣れた人には物足りない
  - 難しいイメージだったプログラミングを分かりやすく身近に感じた
- **普段のプログラミングとの違いについて**
  - 移動や if 文の結果が 1 つ後ろの命令にしか反映されないことに違和感を感じた
  - 紙を並べるだけで描画などの命令ができるため楽で良い
  - どこから if 文が始まるのか分かりづらかった
  - 塗り潰しなどの動きがあり面白い
- **紙を用いた環境について**
  - 書き込むのが楽しい
  - 条件分岐文を塗り潰しで作るのが斬新で面白い
  - 鉛筆などアナログなものを使ったので身近に感じることができた
  - 一度書いたものを消す際に少し手間を感じた
  - 命令オブジェクトのデザインが視覚的に理解しやすいものであった
  - 自分が描いたものが動かせるのは楽しい
- **システムをスムーズに利用することができたかについて**
  - 変数の宣言がいらないのに戸惑いがあった
  - 条件分岐文の使い方を理解するまで時間がかかった
- **意見や要望**
  - switch-case 文がほしい
  - 実行結果はもっと大きく，激しいほうが楽しい
  - 音楽や映像などファイルが読み込めると楽しい
  - 条件文や移動命令の適応範囲を広くするために，括弧が使えるとよい

6.4.3　観察結果

繰り返しの回数や条件分岐を決めるための塗り潰し方法を間違える被験者が２人いた．問題２において，出題の意図としては繰り返し文の中に条件分岐を入れる方法 (図 6.1 回答 A) を回答してもらうものであったが，繰り返し文を２度使用することで条件分岐文を使わない回答 (図 6.1 回答 B) を行った被験者が３人いた．条件分岐において，本来は実行出来ない＞＝や＜＝を作ろうとした被験者が２人いた．

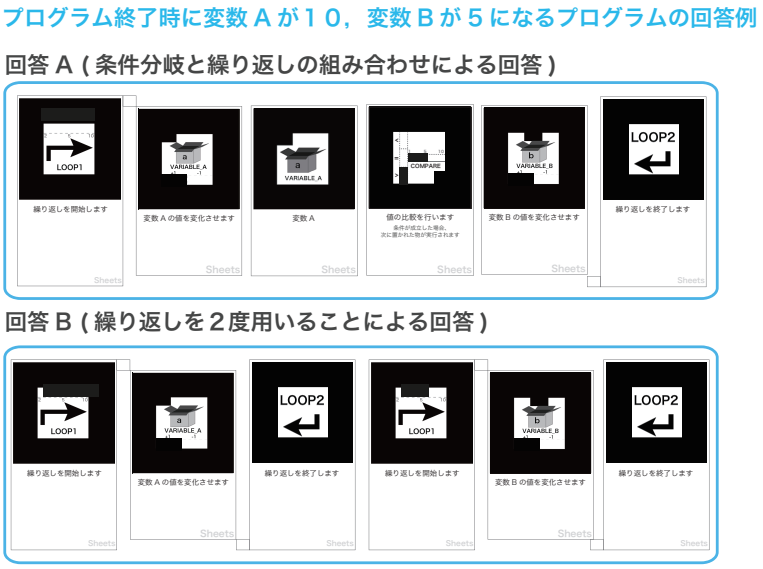


図 6.1: 問題２に対する回答結果例

評価実験を行っている様子を図 6.2 に示す．

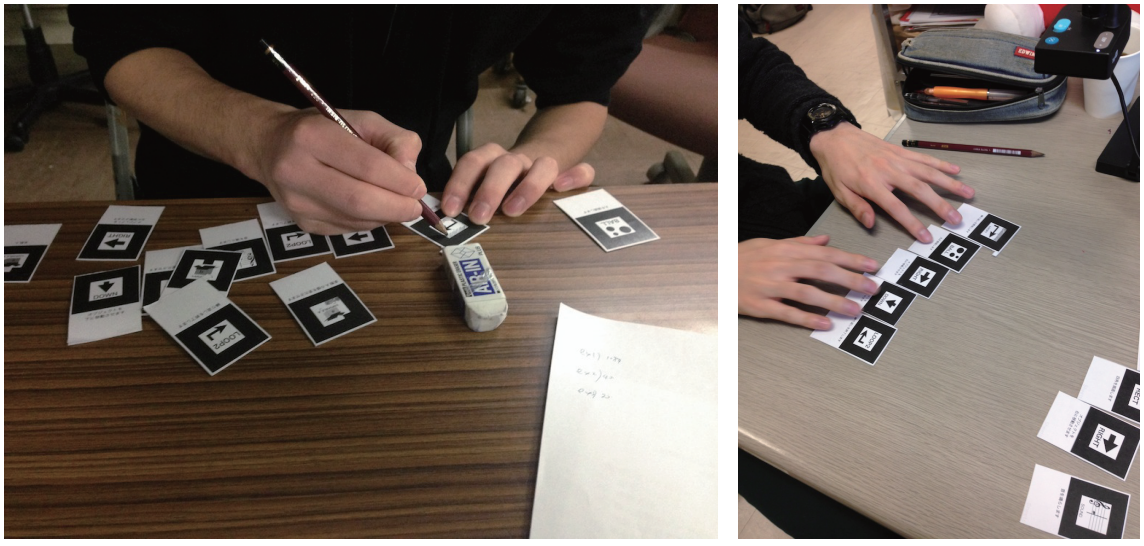


図 6.2: 評価実験の様子

6.5 考察

各問題に対する回答時間の結果を見ると，プログラミングの経験が豊富なほど素早い回答が得られ，ミス率も低いことが分かった．その差は提示される問題が複雑であるほど顕著であった．これは，Sheets のプログラミング言語としてのデザインが，実際のプログラミング言語における構文を単純にマーカー化したものであるためであると考えられる．繰り返し構文においては，繰り返し開始，終了オブジェクトで囲んだ部分が繰り返し対象になるという視覚的な直感に即したものであったが，条件分岐構文においては，一般的な if 文をただマーカーに置き換えるだけでは分かりやすさに繋がらないことが分かった．そのため，条件分岐の構文や方法をより抽象的にデザインし直す必要がある．

また，紙への書き込み動作やハイライト機能に対しては前向きな意見が多かった．特に塗り潰しやイラストを描くことがプログラミングへの親しみに繋がったとの意見が多くみられたため，紙ならではの動作をより環境に取り入れることで，未経験者がプログラミングを身近に感じるための手だてになるものだと考える．

第7章　議論

7.1　本手法の適用範囲

7.1.1　環境について

本論文では，紙媒体の命令オブジェクトと web カメラのみによる準備が容易なタンジブルプログラミング学習環境を示した．命令オブジェクトには AR マーカーが印刷されており，それを web カメラによって撮影することで命令の認識を行っているが，web カメラを用いる特性上，使用する環境によって認識率が変化する．被験者実験を行っている際においても，部屋の照明や被験者自身の影などによって，誤認識が起こることが場面が多々見られた．web カメラによって取り込んだ映像を解析してマーカーの認識を行うため，ある程度の明度やコントラストのコントロールはソフトウェア側で可能であるが，極端に薄暗い部屋もしくは明るい部屋での使用は現状のシステムでは困難である．

また，web カメラの視野角の限界により，一度に実行出来る命令オブジェクトの数が限られる．現状では 10 個前後が限界である．そのため，複雑すぎるプログラムを組み上げることが難しく，あくまでプログラミングのシンタックスに触れる程度に留まっている．これは web カメラ自身の移動や制御によって解決可能であるが，実現には特別な装置が必要になるため，整備が容易な環境という本システムのコンセプトに反することになる．少ない命令で豊かな表現を可能にする言語デザインを実現するか，命令オブジェクトの大きさ変更などの検討を今後行う必要がある．

7.1.2　対象者について

本環境はプログラミング未経験者を使用対象者と定めていたが，被験者実験の際の観察やアンケートからは，プログラミングの経験があるが挫折や難しさを感じた人にとって有益な環境であるという結果が読み取れた．これは実行順序がハイライトされてプログラムを視覚的に理解することができることや，ソースコードが提示されることが影響していると考える．初学者にとってはやはり既存の環境と同じくプログラミングの雰囲気を掴む程度のものに留まっており，確かな理解に繋がったかの測定がまだできていない．

また，ソースコード等を提示することによって理解を助けるという単純な仕組みではなく，初学者のプログラミングに対するモチベーションや親しみを提供できるような機能について検討する必要があると考える．

7.2　紙を用いたインタラクションについて

紙に対する書き込みなどによるインタラクションをプログラミングの過程に取り入れることが，利用者の学習に良い影響を与えることがあるということが被験者実験によって明らかになった．現状では，紙に対するインタラクションが命令を切り替えるための塗り潰しや，イラストを描くという範囲に留まっているが，さらに紙のアフォーダンスを応用した機能についても検討の余地があると考える．例えば，紙を折る，切る，重ねる，などの動作があげらる．

また，紙の特徴として，スキャナなどにより容易に電子化できることがあげられる．作った命令をスキャンし，一つの命令オブジェクトにまとめる関数化などの機能を作ることにより，プログラム内容を多様化，複雑化することが可能になる上，web カメラの視野角という環境における問題を解決することができる．

さらに，タンジブルなプログラミング環境ならではの使い方として，複数人による共同作業があげられる．複数のユーザが別々に作ったプログラムを合成させることができる機能や，複数人ならではの学習を促進する機能について今後検討の余地があると考える．

第8章　結論と今後の課題

本論文において，紙媒体の命令オブジェクトとwebカメラを使用することにより，安価かつ手軽であるタンジブルなプログラミング学習環境を提示し，その実装を行った．実装したシステムであるSheetsは，命令オブジェクトを並べることによってプログラミングが可能な環境である．環境の特徴として，命令の実行順序が可視化されること，Processing言語への変換を行うこと，実世界イベントや紙に対するインタラクションを学習の過程に取り入れたことがあげられる．

また，作成した学習環境を用いて被験者実験を行い，ユーザの環境に対する利用感や，学習における有益な機能，不完全な機能について明らかとなった．

今後の課題として，本手法の適用可能範囲の拡大，ならびに紙ならではの機能をより多く取り入れた学習環境デザインの向上があげられる．

謝辞

本研究を進めるにあたり，指導教官である時井真紀先生には多くのご指導及びご助言を頂きました．また，システム情報工学研究科，インタラクティブプログラミング研究室の田中二郎先生，志築文太郎先生，三末和男先生，高橋伸先生には，他学類の身ながら，日頃研究に関する大変たくさんのご指導を頂きました．特に田中二郎先生には，日々の研究活動や論文の執筆，学会参加へのサポート，研究生活における心構えまで，至る所で親身にご指導頂きました．心より感謝申し上げます．

時井研究室並びにインタラクティブプログラミング研究室の皆様には研究活動において様々なアドバイスを頂きました．一部の方には評価実験の被験者としてご協力いただきました．誠にありがとうございます．

また，知識情報・図書館学類の長谷川秀彦先生，宇陀則彦先生には，研究室配属や卒業研究の方針において非常に寛大な配慮をして頂きました．大変感謝しております．

最後に，自分の生活を支えてくださった両親，日常生活でお世話になった友人，本研究をご支援くださった皆様に心から感謝申し上げます．

参考文献

[1] 文部科学省 2008，幼稚園, 小学校, 中学校, 高等学校及び特別支援学校の学習指導要領等の改善 (答申)(online)，入手先 (`http://www.mext.go.jp/a_menu/shotou/new-cs/news/20080117.pdf`) (2014.12.20).

[2] Mitchel Resnick, John Maloney, et-al. *Scratch programming for all*, Communications of the ACM, Vol.52, No.11，pp.60-67，2009.

[3] Michael S. Horn, Erin Treacy Solovey, R. Jordan Crouser and Robert J. K. Jacob. *Comparing the use of tangible and graphical programming languages for informal science education*, In Proceedings of the 27th international conference on Human factors in computing system(CHI'09)，pp.975-984，2009.

[4] Michael S. Horn and Robert J. K. Jacob. *Designing tangible programming languages for classroom use*, TEI '07 Proceedings of the 1st international conference on Tangible and embedded interaction，pp.159-162，2007.

[5] 宮下芳明．プログラミングに対するモチベーションを向上させる新言語 *HMMMML* の開発, 第 51 回プログラミングシンポジウム論文集，pp.57-64，2010.

[6] 中橋雅弘，宮下芳明．*HMMMML2*：超好意的に解釈するコンパイラ, 情報処理学会夏のプログラミング・シンポジウム報告集，pp.107-110，2011.

[7] 中橋雅弘，宮下芳明．*HMMMML3*：他人を意識したモチベーション向上を考えたプログラミング環境, インタラクション 2011 論文集，pp.511-514，2011.

[8] 長慎也，甲斐宗徳，川合晶，日野孝昭，前島真一，筧捷彦． *Nigari-Java* 言語へも移行しやすい初学者向けプログラミング言語, コンピュータと教育研究会報告，pp.13-20，2003.

[9] 加藤邦拓，宮下芳明．時間とのインタラクションによるプログラミング支援, 情報処理学会研究報告 HCI，HCI-149，No.2，pp.1-6，2012.

[10] 佐藤竜也，志築文太郎，田中二郎．実行の可視化システムと連動した統合開発環境による *GUI* ベースプログラムの理解支援，第 15 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS2007) 論文集，pp.25-30，2007，

[11] Yasunori Harada and Richard Potter. *Fuzzy rewriting:soft program semantics for children*，In IEEE Symposium on Human Centric Computing Languages and Environments，Vol.1， No.1，pp. 39-46，2003.

[12] Danli Wang, Yang Zhang, Tianyuan Gu, Liang He and Hongan Wang. *E-Block: a tangible programming tool for children*, Proceedings of the 25th annual ACM symposium on User interface software and technology(UIST2012), pp.71-72, 2012

[13] Danli Wang, Cheng Zhang and Hongan Wang. *T-Maze: a tangible programming tool for children*, In Proceedings of the 10th International Conference on Interaction Design and Children(IDC'11), pp.127-135, 2011.

[14] Andrew Cyrus Smith, Heinrich Springhorn, Steven Bruce Mulligan, Ireyan Weber and Jackie Norris. *tactusLogic: Programming using physical objects*, IST-Africa Conference Proceedings, pp.1-9, 2011.

[15] Daniel Gallardo, Carles F. Julia‘ and Sergi Jorda‘. *TurTan: a Tangible programming language for creative exploration*, In Proceedings of the 3rd annual IEEE international workshop on horizontal human computer systems (TABLETOP '08), pp.412-420, 2008.

[16] Jie Qi and Leah Buechley. *Sketching in circuits: designing and building electronics on paper*, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems(CHI'14), pp.1713-1722, 2014.

[17] Michael S. Horn, Sarah AlSulaiman, and Jaime Koh. *Translating Roberto to Omar: Computational Literacy, Stickerbooks, and Cultural Forms* , In Proceedings of the 12th International Conference on Interaction Design and Children(IDC'13), pp.120-127, 2013.

付録

次ページ以降に 6 章の実験に関連する書類を示す.

実験前アンケート

・所属(学類，学年)：

・氏名：

・プログラミングへの興味・関心：
(弱い) 1 2 3 4 5 (強い)

・プログラミング経験度合い：
全くない ・ 授業で触れた程度 ・ 普段から使用している

・ 授業等で教わった年数：
年 (週 回，1回につき 時間程度)

・キーボードによる文字入力以外の環境(例えばマウスやペンタブレットなど)によるプログラミング学習環境の利用経験
ある ・ ない （ある場合，その環境名： ）