メディア表現教育を総合的に支援するためのプログラミング環境

杉浦　学[*1]
Email: manabu [at] yamanashi-eiwa.ac.jp

*1：山梨英和大学人間文化学部人間文化学科

◎Key Words プログラミング教育，フィジカルコンピューティング，メディア表現

1. はじめに

1990 年代の初頭から，コンピュータ・センサ・I/O モジュールなどの情報・電子技術を利用してアートを制作する「メディアアート」と呼ばれる分野が登場した．このメディアアートの表現手法の一つとして，鑑賞者の行動などの「フィジカル・インタラクション」を活用するものがある．例えば，作品の鑑賞者の身体的行為に反応する映像や音などを，コンピュータでリアルタイムに生成・表示するといったものである．また，現実世界の出来事と，コンピュータで作り出した仮想空間の映像や音を関連付けることで，現実世界と仮想空間の境界が曖昧に感じられる体験を鑑賞者に提供するといった手法がある．

本研究の目的は，こうしたフィジカル・インタラクションの仕組みを学習者が構築することにより，基礎的な表現設計と情報技術の両面を学習する「メディア表現教育」の実践を支援することである．フィジカル・インタラクションの仕組みを実現するためには，鑑賞者の行動などを測定するセンサの制御，センサからの入力値によって映像や音などを生成・変化させるためのプログラミングが必要となる．これに加え，センサなどの電子部品と回路，それらを制御するハードウェアに関する知識も必要となるため，初学者が短時間でこれらの全てを学習するのは困難である．そこで，映像表現を制作するための Processing によるプログラミングと，Arduino の制御プログラミングの双方に対応したビジュアルプログラミングエディタである「Sketch Blocks」を開発し，メディア表現教育におけるプログラミングの支援を試みる．

2. メディア表現教育の事例と必要なシステム

ここでは，本研究が支援対象に想定するフィジカル・インタラクションを取り入れたメディア表現教育の事例を整理し，作品の制作に必要なシステムの構成について述べる．

2.1 教育実践の事例

事例の 1 つ目として，有賀らの実践[(1)]がある．この事例では，テーブルにセンサと制御基板を設置し，テーブルに対する身体的行為（触る，たたく，なでる，息を吹きかける等）によって，PC に接続された天井のプロジェクタからテーブルに投影される映像が変化する作品（図 1）を制作する教育実践である．テーブルに対する行為と映像の関連性を設計するデザイン能力，センサ制御と映像表現を連動させるためのプログラミングを学習するカリキュラムとしている．

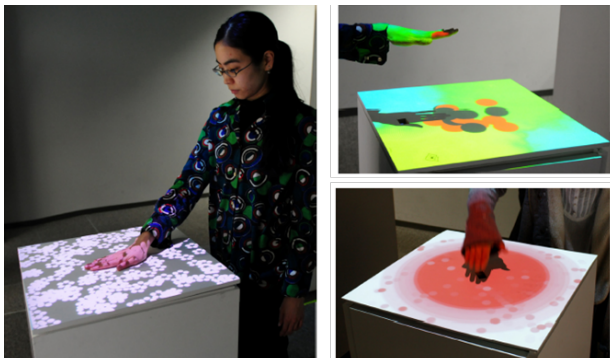

図 1　テーブルインタラクションの作品例[(1)]

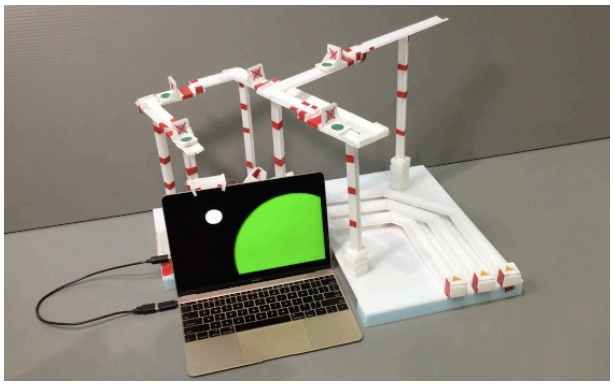

図 2　拡張現実ピタゴラ装置の作品例[(3)]

事例の 2 つ目として，迎山[(2)]や原田[(3)]らによる実践がある．この実践では「拡張現実ピタゴラ装置」と呼ばれる，仮想世界と現実世界のルーブ・ゴールドバーグ・マシンを組み合わせた装置（図 2）の制作をテーマとしている．コンピュータの周囲に自作のレールを配置し，レールを転がるビー玉が仮想世界の PC の画面を通り抜けるといった表現や，仮想世界から現実世界にビー玉が出現するような表現を制作する．ビー玉の動きと PC で表示するアニメーションが連携する表現のデザイン，レールの造形に加えて，ビー玉の動きを操るためのサーボの制御，コンピュータによるアニメーションプログラミングを総合的に学ぶカリキュラムである．

これらの教育実践の事例では「情報技術をアートやデザインの手段として活用する能力を身につける」（主に表現やデザインを専攻する学習者に対する教育）と「情報技術をデザインやアートという応用分野を通じて学ぶ」（技術そのものを魅力的な応用分野を例に教育する）という 2 つの側面がある．つまり，情報技術を専門とする学習者だ

けでなく，デザインやアートの分野に興味がある学習者に対しても，情報技術に対する学習意欲を高められる教育実践であるといえる．こうした教育実践を様々な教育機関で実施できるような工夫が求められる．現在では，このような実践は大学での授業を中心に行われているが，教育のための支援環境を用意できれば，中学や高校などでも教育の実施可能性を高めることができるだろう．

2.2　フィジカル・インタラクションのためのシステム

事例として述べた作品を制作するためには，センサ，LED，サーボ，モータなどの電子部品に加えて，それを制御するためのマイクロコントローラが必要となる．また，映像の出力を行う PC については，センサからの入力値に応じて PC の画面出力を変化させ，LED，サーボ，モータなどを動作させるプログラムを記述する必要がある．制作する作品の仕様により，利用する電子部品の詳細は異なるものの，基本的には図 3 に示すような構成のシステムを構築する必要がある．

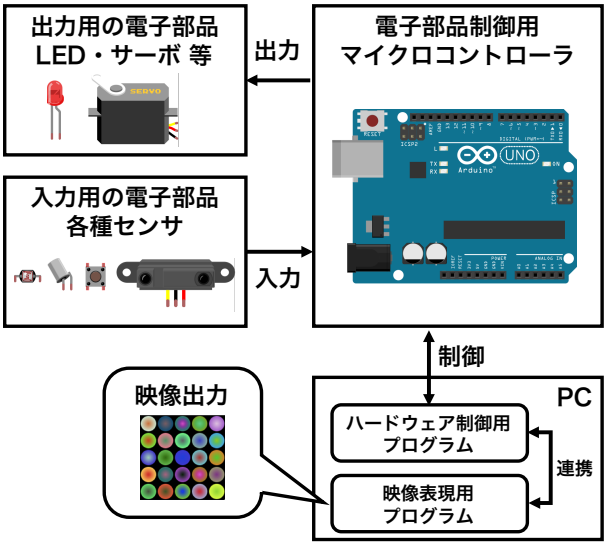

図 3　作品制作に必要となるシステム

PC を使った映像表現については，様々なソフトウェアが利用可能だが，初学者に対する配慮から，既存の実践においては Processing[4] が利用されている．マイクロコントローラに相当するハードウェアに関しては，フィジカルコンピューティングという名称が一般化してきており，Arduino[5] や Gainer[6] をはじめとした電子工作用のマイクロコントローラが安価に流通するようになった．迎山や原田らによる実践では Arduino や Gainer が利用されている．有賀らの実践では，ブレッドボードによる電子回路の作成が不要なオリジナルのマイクロコントローラが利用されているが，Arduino で代替可能である．

3.　既存のブロックプログラミングエディタ

図 3 に示したようなシステムを，初学者が構築することを想定した場合，学習すべき事柄は多い．センサからの入力値によって映像や音などを生成・変化させるためのプログラミングに加えて，LED やサーボなどの出力制御，さらにセンサをはじめとした電子部品や回路設計などに関する知識も必要となり，学習を支援するための仕組みが必要である．図 3 に示したようなシステムの全体を考えた場合，電子回路を含めたハードウェアの制作に関する支援も必要となるが，本稿ではマイクロコントローラによる制御と映像表現のプログラミングに焦点を絞って議論する．

支援の必要性に関していえば，有賀らの実践の結果として，1 割程度の学生はプログラミングに否定的な徒労感を覚えると報告されている．さらに，映像表現のプログラミングにおいては，サンプルコードの模倣と再生産のステップから進んで，オリジナルの映像表現を行おうとプログラミングに挑戦した作品については，プログラムの実装が中途半端な形にとどまった結果，教員からの作品全体としての評価が低くなるという問題点が指摘されている．このことからも，映像を制作するためのプログラミングとマイクロコントローラの制御プログラミングを統合的に支援する必要があることが分かる．

マイクロコントローラとして一般的な Arduino を対象とし，初学者でも扱いやすいように配慮されたブロックプログラミングエディタは既に多く開発されている．こうしたエディタを利用することにより，Arduino 言語の文法習得が不要となり，文法エラーが発生しないという利点が得られる．表 1 に筆者が調べた，日本語表記のブロックによるプログラミングが可能で，一般的な PC で実行できるエディタの一覧とそれぞれの機能を示す．

表 1　既存エディタの機能比較

名称	映像制作機能	言語移行機能
ArduBlock[7]	×	○
BlocklyDuino[8]	×	○
mBlock[9]	△	△
S4A[10]	○	×

○：あり，△：限定的にあり，×：なし

本研究で支援対象として想定する作品の制作を考えた場合，既存の支援環境で着目すべきは，Arduino の制御プログラミングだけでなく，アニメーションなどの映像制作に関するプログラミングも含めた支援が可能かという点である．多くの既存エディタは Arduino の制御プログラミングのみを対象としており，本研究が対象とするメディア表現の作品制作には不十分である．これについては表 1 中の「映像制作機能」の項目に示した．

発展的な学習を視野に入れた場合，ブロックで記述したプログラムが，Arduino 言語をはじめとしたテキスト言語にどのように変換できるかを理解しやすい環境であることが望ましいと考えられる．表 1 中の「言語移行機能」の項目に，この機能の有無を示した．この機能により，基礎的な学習はブロックエディタを活用して行い，プログラミングに慣れてきた段階で徐々にテキスト言語の記述によるプログラミングに移行するといったカリキュラムも実施可能となる．なお，表 1 中の mBlock は通常モードと Arduino モードを区別しており，通常モードではブロックで記述した Arduino 言語のコードは表示されないが，映像制作はサポートできる．一方，Arduino モードではブロックで記述した結果に対応する Arduino 言語が表示できるが，映像制作に関するブロックは利用できなくなる

ため，両方の機能を一度に使うことができない．このことから表 1 中では△を記載してある．

4. Sketch Blocks

4.1 概要

表 1 に示したように，映像制作機能と言語移行機能の両方を持ち，メディア表現作品のプログラミングを統合的に支援できるブロックプログラミングエディタは存在しない．そこで，両方の機能を持つ「Sketch Blocks」を開発した．Arduino IDE のツールとして動作する ArduBlock を実装の基盤とし，Processing Development Environment (PDE) の外部ツールとして動作するように変更した．さらに，Processing による映像制作のための各種ブロックの追加，PDE 上で Arduino の制御を行うためのブロックや設定機能を追加した（図 4）．

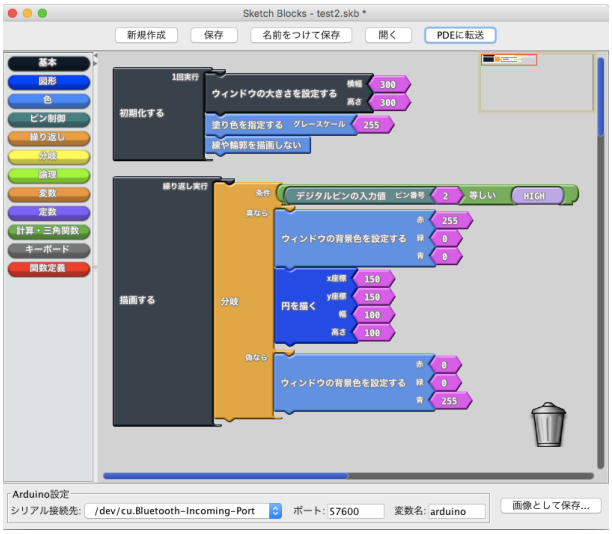

図 4　Sketch Blocks の実行画面

4.2 主な機能

機能 1：ブロックによるコード記述

OpenBlocks フレームワーク[11]によるブロックエディタを使って，Arudino を制御するためのプログラムと，Processing による映像表現のためのプログラムを同一エディタ上で記述することができる（図 5）．

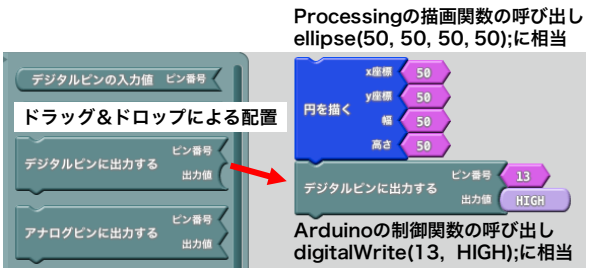

図 5　ブロックによるプログラムの記述

各種のブロックは画面の左側に配置されたパレットに収納されており，必要なブロックをマウスで選択し，右側のスペースに配置して組み合わせることで，プログラムを記述する．図 5 に示したプログラムは，「円を描く」という Processing の描画関数の呼び出しと「デジタルピンに出力する」という Arduino の制御関数の呼び出しを連続して行う場合の例である．

機能 2：Processing への変換・PDE への転送

ブロックエディタで作成したコードは任意のタイミングで Processing のコードに自動変換し，PDE 上のエディタ画面に転送することができる．図 6 に示したのは，Arduino の 2 番ピンに接続されたタクトスイッチを押すと，PC に表示された Processing ウィンドウの背景色が青から赤に変更されるプログラムである．図 6 のブロックを自動変換して PDE に転送したコードを図 7 に示す．ブロックで記述していないライブラリのインポートのコード，ピンの入力設定などは変換時に自動的に付与される．プログラムのコンパイルと実行は PDE で行う．

図 6　タクトスイッチと描画を連携させるプログラム

```
import processing.serial.*;
import cc.arduino.*;

Arduino arduino = new Arduino(this, "/dev/tty.usbmodem1421", 57600);

void setup() {
  size(300, 300);
  arduino.pinMode(2, Arduino.INPUT);
}

void draw() {
  if (arduino.digitalRead(2) == Arduino.HIGH) {
    background(255, 0, 0);
  } else {
    background(0, 0, 255);
  }
}
```

図 7　図 6 のブロックを変換・PDE に転送した結果

機能 3：編集差分のハイライト表示

ブロックエディタで編集した差分の箇所が，テキスト言語に変換されたコードのどの部分に対応しているのかをハイライトして表示する．図 6 のコードの一部に「円を描く」というブロックを追加し，PDE にコードを転送した様子を図 8 に示す．差分（増分）のコードである「ellipse(50, 50, 50, 50);」の行がハイライト表示され，ブロックによるコードの記述がどのようにテキスト言語に変更を与えたかが理解できるように工夫した．また，関数の引数（ellipse 関数の場合は座標や大きさなど）の数値など，ブロックの一部を変更した場合も該当する部分がハイライトされるため，テキストで関数を記述する際の引数の役割も理解できることが期待できる．プログラミング慣れてきた学習者は Sketch Blocks を利用せずに，PDE 上でテキストによるプログラミングを行うことに移行しやすくすることをねらった．

PDE上での表示

```
10
11 void draw() {
12   if (arduino.digitalRead(2) == Arduino.HIGH) {
13     background(255, 0, 0);
14     ellipse(50, 50, 50, 50);
15   } else {
16     background(0, 0, 255);
17   }
18 }
19
```

図 7　編集差分のハイライト表示

4.3　実装例

図 3 に示したシステムの基本要素を再現した電子回路を用意した. Arduino とブレッドボードによる電子回路の配線図を図 8 に示す.

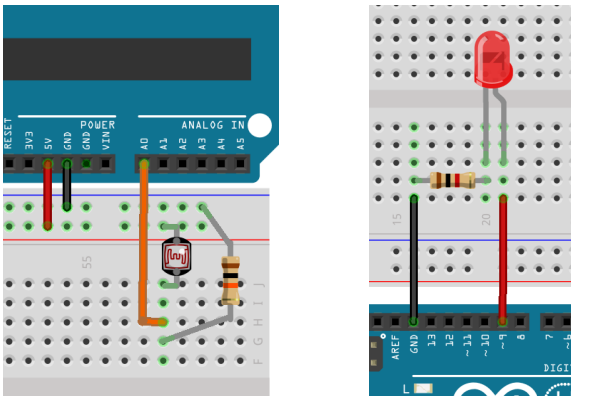

図 8　入力回路と出力回路

フォトレジスタで周囲の照度を測定（図 3 の入力用の電子部品の制御に相当）し，その結果によって PC の画面上に表示された円の大きさ（図 3 の PC による映像出力に相当）を表示するアニメーションを表示させる. また, LED の明るさ（図 3 の出力用の電子部品の制御に相当）とフォトレジスタが読み取った値を連携させ，周囲が暗くなると LED も暗く点灯するようにする.

このプログラムを記述するブロックを図 9 に示す. 図 9 に示したブロックを Processing のコードに自動変換し，PDE に転送した結果を図 10 に示す. Sketch Blocks によって，図 3 に示したシステムの基本要素の制御に必要なコードが記述できる.

図 9　入出力制御と画面描画のプログラム

```
1  import processing.serial.*;
2  import cc.arduino.*;
3
4  int input;
5
6  Arduino arduino = new Arduino(this, "/dev/tty.usbmodem1421", 57600);
7
8  void setup(){
9      size(300, 300);
10     arduino.pinMode(9, Arduino.OUTPUT);
11 }
12
13 void draw(){
14     background(0);
15     input = arduino.analogRead(0);
16     ellipse(150, 150, input / 3, input / 3);
17     arduino.analogWrite(9 , input / 5);
18 }
```

図 10　PDE に転送された Processing のコード

5.　まとめ

本稿では，フィジカル・インタラクションを活用したメディア表現に関する教育を対象とし，学習者が作品を制作するために必要なプログラミングを統合的に支援可能な Sketch Blocks の開発について述べた.

今後は Sketch Blocks を使った実験授業を行い，支援の効果について詳細な検証を実施する予定である. また，入出力回路の設計と作成などのハードウェアに関する支援について検討を行う予定である.

謝辞

本研究は JSPS 科研費 JP26750089 の助成を受けた.

参考文献

(1) 有賀妙子, 森公一 : “フィジカル・インタラクションを使ったメディア造形基礎教育におけるプログラミング学習の実践”, 情報処理学会論文誌, Vol. 52, No. 12, pp. 3096-3105 (2011).
(2) 迎山和司 : “拡張現実ピタゴラ装置 : 自然なインタフェースのための授業”, 情報科学芸術大学院大学紀要, Vol. 4, pp. 5-10 (2012).
(3) 原田泰 : “拡張現実ピタゴラ装置 2016”, https://youtu.be/OcVDcWUf7R0 (2017 年 6 月閲覧)
(4) Processing : http://processing.org (2017 年 6 月閲覧)
(5) Arduino : http://www.arduino.cc (2017 年 6 月閲覧)
(6) Gainer : http://gainer.cc (2017 年 6 月閲覧)
(7) ArduBlock : http://blog.ardublock.com (2017 年 6 月閲覧)
(8) BlocklyDuino : https://code.makewitharduino.com (2017 年 6 月閲覧)
(9) mBlock : http://www.mblock.cc (2017 年 6 月閲覧)
(10) S4A : http://s4a.cat (2017 年 6 月閲覧)
(11) http://web.mit.edu/mitstep/openblocks.html (2017 年 6 月閲覧)