ピクっと

- ピクトグラム作成を通じた状態遷移モデルの学習環境 -

伊藤 一成 [1,a)]

概要：筆者は，ピクトグラムコンテンツ作成環境「ピクトグラミング」を提案している．ピクトグラミングは，プログラミング学習環境の側面も有し，様々な文脈で利用されている．人型のピクトグラムを変形する命令，線画によるオブジェクト記述のための命令，セーフティーマーク記述のための命令を併用することで，短時間でピクトグラムのデザイン指針に準じたアニメーション効果を含む多様な作品を作成できる．同時にプログラミングの諸概念が学習できるのが特徴である．また，Python 言語，JavaScript 言語，ビジュアルブロック言語により記述できる派生環境もインターネット公開している．ピクトグラミングおよび派生環境は，手続き型のプログラミングパラダイムをベースとしている．ピクトグラミングの資産を活用しつつ，ピクトグラムの状態とその状態遷移を状態遷移図で表現するピクトグラムコンテンツ作成環境「Picuit（ピクっと）」を新たに開発した．本論文では，その実装方式を中心に論ずる．

キーワード： ピクトグラム，ピクトグラミング，状態遷移図，ビジュアル言語，同調的学習

Picuit

Pictogram content creation environment based on state transition diagram

KAZUNARI ITO[1,a)]

Abstract: We proposed pictogram content creation environment called “Pictogramming.” This application provides functions to transform the human-shaped pictogram, to draw objects by line drawing, and to set safety sign. The combination of three type of functions enables user to create various works based on design guideline in a short time and can learn some basic concepts of programming languages. We also released some derivative applications on the web, which can describe with Python, JavaScript and visual block. Pictogramming and its derivative applications are classified as a procedural programming language. This time we constructed “Picuit,” which visualizes pictogram’s states and their state transitions with state transition diagram while inheriting the characteristics of Pictogramming. This paper argues an implement method.

Keywords: Pictogram, Pictogramming, State transition diagram, Visual language, Syntonic learning

1. はじめに

筆者は人型ピクトグラムを用いたコンテンツ作成環境「Pictogramming（ピクトグラミング）」を提案し Web で公開している[1]．Pictogramming は，Pictogram（ピクトグラム）と Programming（プログラミング）を合わせた造語で，プログラミング学習環境の側面も有している．Pictogramming は，ピクトグラムを作るという目的指向で設計されており，独自の疑似言語によるピクトグラムの作成を通じてプログラミングの諸概念(順次処理, 並行処理, 繰返し, 条件分岐等)を体得できるように設計されている．さらに，Python 言語で記述できる Picthon（ピクソン）[2]や，JavaScript 言語で記述できる JavaScpict（ジャバスクピクト), ビジュアルブロックを組み合わせてプログラミングできる Block Pictogramming（ブロックピクトグラミング）も開発している[3]．これらのアプリケーション群をまとめて，ピクトグラミングシリーズと呼称しており，全てインターネット上で自由に利用できる[4].

ピクトグラミングシリーズは，それぞれ異なるプログラミング言語で，人型ピクトグラムの動作や各種図形の描画を細かい粒度で記述できる．一方で，プログラムが複雑になると全体の動作や挙動を俯瞰して理解しづらくなるという一面も有していた.

近年，Computational Thinking が注目され[5]， 計算論的な概念や思考の重要性が広く謳われるようになってきた.

その概念の一つに，ソフトウェアやプログラムの状態の遷移を図示する状態遷移図がある．設計やモデリングの分野で広く用いられており，情報教育の領域でもオートマトン，モデル化とシミュレーションや計測制御などで取り入れられている[6][7][8][9].

[1] 青山学院大学
Aoyama Gakuin University, Kanagawa 252-5258, Japan
[a)] kaz@si.aoyama.ac.jp

そこで，ピクトグラムの状態群とその状態遷移を状態遷移図で表現することで，ピクトグラムコンテンツ作成を通じて，状態遷移モデルについて学習できる環境「Picuit（ピクっと）」を新たに提案する．

以下 2 章でピクトグラミングおよび派生アプリケーションの概要を示す．3 章で，今回実装した Picuit について説明し，4 章で，先行研究との関連について述べつつ，考察する．5 章でまとめる．

2. ピクトグラミングおよび派生アプリケーション

本章ではピクトグラミングおよび派生アプリケーションについて解説する．

2.1 実装方式と画面説明

ピクトグラミングは，Web アプリケーションであり，ブラウザ上で動作する．図 1 にスクリーンショットを示す．

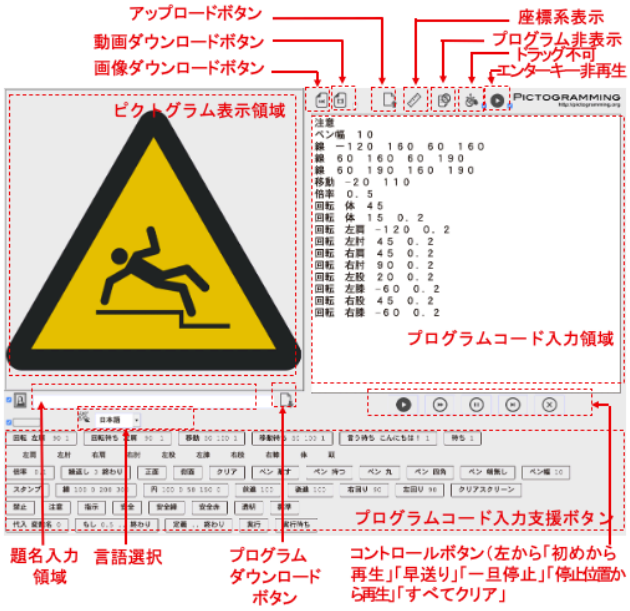
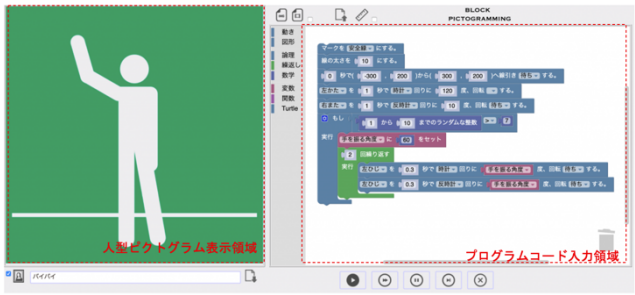


図 1　ピクトグラミングのスクリーンショット

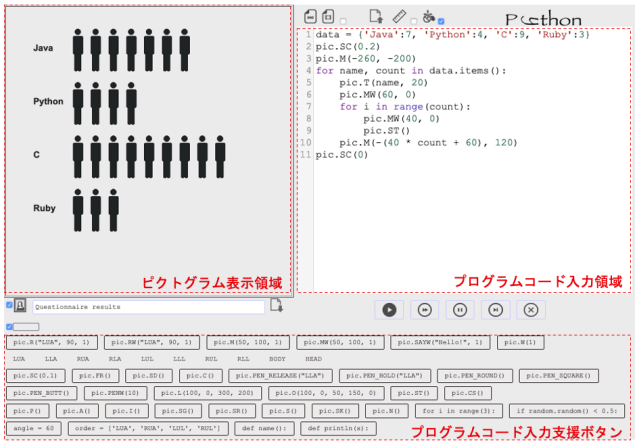
Fig.1　Screenshot of Pictogramming.

画面上部左側は，プログラムの実行結果を表示する”ピクトグラム表示領域”，上部右側はプログラムを入力する”プログラム記述領域”，下部にはプログラムの入力を支援するボタン群が配置されている”入力支援ボタン領域”がある．ピクトグラム表示パネルには，ISO3864 で定義されている形状の人型ピクトグラムが表示される．人型ピクトグラムは，体と頭を合わせた部分が 1 つと，上腕，前腕，上腿，下腿が左右それぞれ 1 つの計 9 種の部品が連結されている．各部位のサイズは ISO3864 で定義されているものを忠実に再現している．

Pictogramming は，既存の教育向けプログラミング言語[10]にあるように，プログラムの記述に教育利用を念頭にした独自記法を用いており，アルファベットだけではなく，日本語やひらがなによる記法もサポートしている．さらに，初等教育段階からのプログラミング教育への関心の高まりやプログラミングの苦手意識の高い学習者にも配慮し，視覚的なブロックを使って記述可能なピクトグラミングの派生環境である Block Pictogramming（ブロックピクトグラミング）も公開している．スクリーンショットを図 2(a)に示す．一方で，ピクトグラミングの特徴を継承した汎用プログラミング言語の学習環境を構築する意義は高いと考えられ，Python 言語で記述できる Picthon（ピクソン）や，JavaScript 言語で記述できる JavaScpict(ジャバスクピクト）も公開している．ピクソンのスクリーンショットを図 2(b)に示す．



(a) ブロックピクトグラミング



(b) ピクソン

図 2 各種派生アプリケーションのスクリーンショット

Fig.2　Screenshot of derivative applications.

これらの派生アプリケーションは，いずれも図 1 で示したピクトグラミングとほぼ同一のユーザインタフェースを備えている．いずれも，書かれたプログラムの実行も含めて，全てブラウザ上でされるため，学習者の PC の環境整備が困難である教育機関でも利用可能である．

2.2 プログラム記述仕様

人型ピクトグラムの変形及び動きの変化を表現できるアニメーションの機能を実装しており，回転と移動の 2 種類の命令を使い分け，さらに命令の逐次実行と並列実行を組み合わせることで，多様な動作を実現可能とする．

また，変数定義，繰返し，条件分岐や関数定義などの制御命令が用意されている．人型ピクトグラムが左手を振るプログラム例を図 3 に示す．

(a) ブロックピクトグラミング

```
回転 左肩 -120 1
回転待ち 右股 10 1
移動 100 -20 1.2
確率 0.3
  繰返し 2
    回転待ち 左肘 -60 0.3
    回転待ち 左肘 60 0.3
  終わり
終わり
```

(b) ピクトグラミング（日本語）

```
pic.r("LS", -120, 1)
pic.rw("RC", 10, 1)
pic.m(100, -20, 1.2)
if random.randrange(1, 10, 1) < 3:
    for _ in range(2):
        pic.rw("LE", -60, 0.3)
        pic.rw("LE", 60, 0.3)
```

(c) ピクソン

図 3 ピクトグラミング及び
各種派生アプリケーションのプログラム例

Fig.3 Example programs of Pictogramming and derivative applications.

図 3(a)のブロックピクトグラミングを例に，プログラム例の命令と動作について説明する．図 3(a)中の<1>で示されたブロックは，身体部位に対する回転の命令である．関節点，回転方向・角度，回転に要する秒数を設定する．

部位を指定する際，例えば人型ピクトグラムの左腕が右側に表示されると，自身の右腕と対応づけてしまい，混乱が予想される．そこで，人型ピクトグラムは，鏡に映っている自分に相当するという想定としている．人型ピクトグラム表示パネルに表示される人型ピクトグラムの各部位と位置の関係を図 4 に示す．鏡あるいは仮想的な鏡の前で，人が映った自分を見ながら動きを学習することは，日常行為だけではなく，スポーツや作業手順の動き習得などの分野でも幅広く行われている．ピクトグラムデザインにおける「人型ピクトグラムと自身との同一視」を重視する観点からも親和性が高く，この方式を採用している．

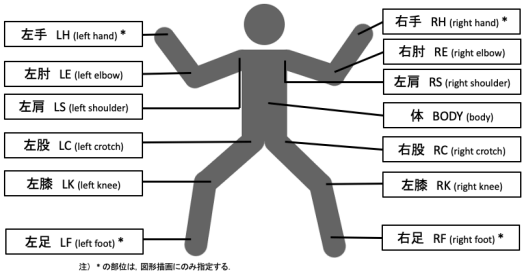


図 4 人型ピクトグラムの部位指定

Fig.4 Specified labels for the parts of the human pictogram

連結された次のブロックを同時に実行するか，実行が終了するまで，次のブロックを実行しないかを「待ち」の有無で選択できる．よって，<1>と<2>のブロックは同時に実行され，<3>のブロックは，<2>のブロックの実行完了後に実行される．このように「待ち」の有無だけで，順次処理と並行処理が表現でき，これにより多様な動作を実現できる．<3>のブロックは体全体を平行移動する命令である．<4>のブロックは，このブロックに囲まれた内側の命令を一定の確率で実行する．これは分岐命令に相当する．<5>のブロックは，このブロックに囲まれた内側の命令を一定回繰り返す．つまり<4>から<7>のブロックは，左右に左ひじを回転することで 2 回手を振る動作を 30%の確率で実行することを記述している．

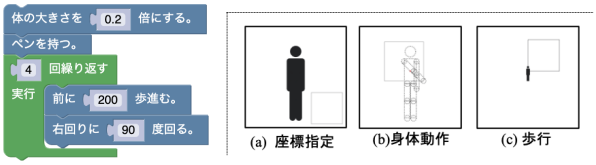
図 3(a)のブロックピクトグラミングと同一の出力をするピクトグラミング（日本語）のプログラム例を図 3(b)に示す．命令名に続いて複数の引数をスペース区切りで記述する形式となっている．同様に，図 3(c)にピクソンの例を示す．pic の識別子で指定されるオブジェクトがあらかじめ生成されており，そのオブジェクトに対するメソッド呼出しの書式で，ブロックピクトグラミングやピクトグラミングと同じ命令群を記述できる．人型ピクトグラムを複製する命令も存在するので，図 3(c)に表示されているような，人型ピクトグラムを構成素とするインフォグラフィックスも作成できる．

プログラミングの入門講座では，図形描画がテーマの実践も多い[11]. 比較的単純な命令セットで扱え，またコードと対応する出力が細かい粒度で視覚化されるのが理由である．単純図形の作画は，ピクトグラム作成においても不可欠な工程である．図 1 のスクリーンショットの例で示されているのは，「段差注意」のピクトグラムであるが，ここで段差のある床を線画で描画している．ピクトグラミングシリーズでは，認知視点の多重化の重要性を踏まえ，認知視点の異なる 3 種類の描画方式を実装している[12]．ブロックピクトグラミングを用いて正方形の描画を，3 種類の方

式によって行なった例を図 5 に示す.

(a) 座標指定

(b) 身体動作



(c) 歩行　　出力

図 5 ブロックピクトグラミングによる図形描画の例

Fig.5 An example of figure drawing using Block Pictogramming.

図 5 の「(a)座標指定」は，座標を指定した線分の組み合わせで正方形を描画する例である．これは汎用プログラミング言語で図形を描画する場合の一般的な方式である．図 5 の「(b)身体動作」は，本アプリケーション特有の描画方式で，体の部位を指定してその部位の動作の履歴を線画で表現する．この例では，左手の動きで正方形を描画している．図 5 の「(c)歩行」は，人型ピクトグラム自体の平行移動の履歴で描画する方式で，主に教育用プログラミング言語で用いられるタートルグラフィックスと同等である．

日常的な人間の動作や行為に関連させて図形描画することにより，人型ピクトグラムや他の参与者，他の参与者が創作する人型ピクトグラムへの自己同一化を促す．

ピクトグラムには，禁止，注意，指示，安全の 4 項目に関するピクトグラムデザインのガイドラインが ISO3864 で策定されている．ピクトグラミングではこれに準拠し，通常のモードに加え，禁止，注意，指示，安全用のピクトグラムを作るために単一の命令・ブロックでこれら 6 種のセーフティサインを設定できる．一覧を図 6 に示す.

図 6 セーフティサインの例

Fig.6 An examples of safety sign

ピクトグラミング及び派生アプリケーションでは,「ピクトグラム表示領域」上での様々なマウス操作や，画面下部の「入力支援ボタン領域」に配置されたプログラムコード入力支援ボタンにより，対応した命令を自動的に「プログラムコード入力領域」に追加できる．そのため静止画のピクトグラムは，マウス操作主体で作成できる．表 1 にマウス操作と対応するアクションの一覧を示す.

表 1 対応するマウス操作と処理一覧

Table 1 List of mouse operations and corresponding processes.

ドラッグ開始時のマウスの座標	マウス操作	処理
人型ピクトグラムの表示領域内	左クリック＋ドラッグ	人型ピクトグラムの対応する部位を回転
	右クリック＋ドラッグ	人型ピクトグラムの平行移動
人型ピクトグラムの表示領域外	左クリック＋ドラッグ	線分の描画
	右クリック＋ドラッグ	楕円の描画
–	長押し	人型ピクトグラムの向いている方向の変更（正面と側面）
人型ピクトグラムの描画可能点付近	クリックのみ	描画可能点におけるペンの持ち・放し

一方，アニメーション効果を持つピクトグラムを作成する場合は，プログラムコード入力領域に入力された命令・ブロック列に対して，変数の値をキーボードで書き換える必要がある．さらに，繰返し，条件分岐，関数，変数等の使いこなすことで，さらに複雑な挙動のピクトグラムを作成できるようになる．このように段階的に学習を進めていくスモールステップの工夫を取り入れている.

3. Picuit（ピクっと）の提案

本章では,提案する状態遷移モデルの学習環境 Picuit（ピクっと）について説明する.

3.1 画面説明

Picuit のスクリーンショットを図 7 に示す.

主に，ピクトグラムの状態の作成や実行時の表示領域である「(a) ピクトグラム表示領域」と，状態群の遷移関係を定義する「(b) 状態遷移リスト表示領域」から構成されている．「(a) ピクトグラム表示領域」は，ピクトグラミングシリーズ共通のコンポーネントである．「(b) 状態遷移リスト表示領域」には,「遷移前状態」,「遷移後状態」,「遷移前と遷移後の関係」の三つ組（以下トリプルという）が複数表示される．トリプルの例を図 8 に示す.

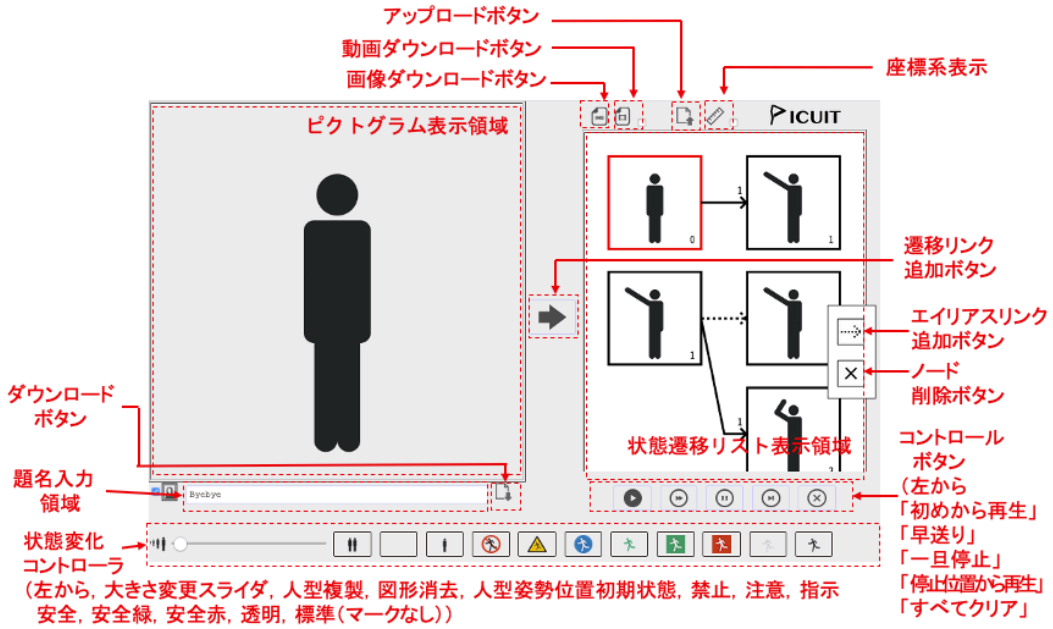


図 7 Picuit のスクリーンショット

Fig.7 Screenshot of Picuit

図 8 状態遷移の表示形式

Fig.8 Display format of state transition

遷移前状態と遷移後状態は，その状態におけるピクトグラム表示領域をサムネイル化した画像で表現される．内部的には，遷移前状態から遷移後状態へのマウス操作による一連の変化を記憶しているが，あくまで遷移前状態と遷移後状態の画像を示している．そのため，アニメーションの場合，遷移前状態から遷移後状態にどのように変化するかは，実際には再生させないと確認できない仕様となっている．この点は今後の検討課題とする．

画像の右下には，作成された順に連番(ノード番号という)が付与される．遷移前と遷移後の関係は，遷移前状態を始点，遷移後状態を終点とする矢印で示す．矢印は，遷移リンク（図 8 左）とエイリアスリンク（図 8 右）の 2 種類が存在する．遷移リンクのトリプルは，遷移前の表示状態と遷移後の表示状態へ，指定された秒数かけて変化する．矢印は実線で描かれ，秒数は矢印のラベルに表示される．一方，エイリアスリンクはピクトグラムの状態が遷移前状態のノードになった場合，遷移後状態のノードに直ちに移行することを意味する．遷移前状態と遷移後状態でピクトグラム自体の姿勢が異なっていることもあり得るが，この際ピクトグラム自体の表示は変化しない．矢印は点線で描かれ，遷移リンクの場合と異なり，秒数は表示されない．

初期状態では，「(a) ピクトグラム表示領域」に直立状態のノード番号 0 のノード（開始ノードという）のみが表示される．

3.2 状態遷移リスト作成方法

Picuit では，ピクトグラムの作成は，トリプルの集合を作成することに帰着される．トリプルの作成手順は，遷移リンクとエイリアスリンクで異なり，それぞれ次のとおりである．

3.2.1 遷移リンクのトリプル作成

遷移リンクのトリプルは次に示す(1)から(3)の手順で作成する．

(1) 遷移前状態のノードの選択

遷移前状態に指定するノードは，作成済みのいずれかのノードをクリックすることで選択できる．選択されたノードは赤色の枠で囲われ，またその状態が「(a)ピクトグラム表示領域」に表示される．

(2) 「ピクトグラム表示領域」の状態の変更

「(a)ピクトグラム表示領域」に表示されているピクトグラムの状態を，先に表 1 で示した各種マウス操作を適用し変化させる．それ以外の状態の変化（人型ピクトグラムの大きさの変更，複製，図形消去，姿勢の初期化，セーフティーマークの設定等）については，図 7 の画面下部の「状態変化コントローラ」にあるスライダーの操作及びボタン押下により変更する．

(3) 遷移後状態の登録

図 7 の画面中央の「遷移リンク登録ボタン」にある登録

ボタンを押下すると遷移前状態に指定したノードと，登録ボタン押下時点の「(a)ピクトグラム表示領域」に表示されているピクトグラムの状態を遷移後状態のノードとするトリプルが「(b) 状態遷移リスト表示領域」に追加表示される．遷移に要する秒数は，トリプルのリンク表示の矢印上に表示される．初期は 0 秒かけて状態が遷移する設定になっているが，秒数表示部分をクリックすると入力ダイアログが表示され，変更できる．

3.2.2 エイリアスリンクのトリプル作成

エイリアスリンクのトリプルは，次に示す(1)から(2)の手順で作成する．

(1) 遷移前状態ノードの選択

3．2．1 項「遷移リンクのトリプル作成」(1) 遷移前状態のノードの選択と同じ方法で選択する．

(2) 遷移後状態の選択

「(b) 状態遷移リスト表示領域」上に表示されるいずれかのノードを右クリックすると，エイリアスリンクボタンを含むパネルが現れる．エイリアスリンクボタンを押下するとそのノードが遷移後状態のノードとなり，遷移前状態に指定したノードとのエイリアスリンクのトリプルが新たに生成される．生成されたトリプルは，「(b) 状態遷移リスト表示領域」に追加表示される．

3.3 作成例

作成されたトリプルの集合は，連結有向グラフを形成する．開始ノードからリンクに沿って状態を遷移していく．遷移するリンクが存在しない状態に到達したところで実行は終了となる．グラフ中のループ構造は無限回の繰返し処理に相当する．また，特定の遷移前状態とするリンクが複数存在する場合，等確率でいずれか一つのリンクを選択して状態を遷移する．つまり分岐処理に相当する．

図 9 に，6 つのトリプルから構成される例を示す．

実行開始時，遷移前状態がノード番号 0 であるトリプル 0-1 が適用される．ここで，「トリプル a-b」とは，遷移前状態がノード番号 a，遷移後状態がノード番号 b のトリプルをあらわす．ノード番号 0 とノード番号 1 の状態の違いである，左腕をあげる動作が 1 秒かけてなされ，ノード番号 1 に状態が遷移する．続けて，ノード番号 1 が遷移前状態であるトリプル 1-2 とトリプル 1-3 のいずれかが等確率で選択される．いずれもエイリアスリンクなので，出力は変化しない．トリプル 1-2 が選択された場合は，ノード番号 2 が遷移前状態であるトリプルは存在しないため，終了となる．一方，トリプル 1-3 が選択された場合には，トリプル 3-4，トリプル 4-5 が順次適用されるため，左上腕を左右に 1 回振る動作を行う．ノード番号 5 に遷移した瞬間，トリプル 5-3 の適用によりノード番号 3 に遷移し，同様に繰り返す．つまり図 9 は，50%の確率で手を振り続ける状態遷移リストである．

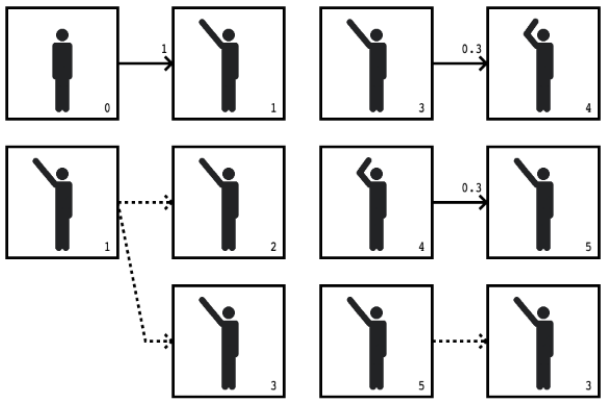


図 9 状態遷移リストの例

Fig.9 Example of state transition list

図形描画や分岐の状態遷移リストの例を図 10 に示す．

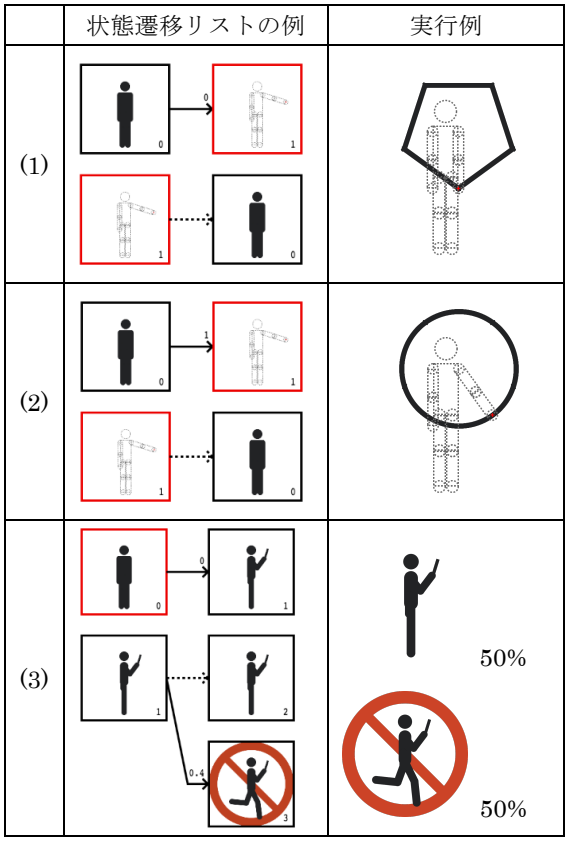


図 10 図形描画や分岐の例

Fig.10 Example of figure drawing and branching

Picuit では，図形描画については表 1 に示したマウス操作で可能な，図 5 の(a)座標指定と(b)身体動作による描画が可能である．

図 10 の(1)，(2)は，ペンを持った右手の動きにより図形を描画する例である．いずれもトリプル 0-1 は，人型ピクトグラムを透明にして右手を 72 度反時計回りに回転して

いる．一方，トリプル 1-0 はエイリアスリンクのため，右手を 72 度反時計回りに回転することを繰り返す処理となる.ピクトグラミングシリーズでは,時間 0 の遷移の場合，ペンの元の位置とその命令を実行したあとのペンの位置を両端とする線分を描画する.一方,時間が正の値のときは，ペンの移動の履歴どおりに描画する．よって，図 10 の(1)に示すようにトリプル 0-1 の秒数が 0 の場合は,多角形が，(2)に示すようにトリプル 0-1 の秒数が正の値の場合には，円が描画される.

図 10 の(3)は，禁止を表すサインの状態（歩きスマホ）とそうでない状態（直立状態でのスマホ）を分岐で表現した例である.

3.4 実行の手順

図 7 のコントロールボタン中の「はじめから再生ボタン」を押すと状態遷移グラフを走査して，図 2 に例示した形式のプログラムに内部で変換し，他のピクトグラミングシリーズと同様に「ピクトグラム表示領域」上で実行する．よって，図 9 や図 10 の(3)で示した例のように，単一ノードからの複数の遷移リンクのいずれかが選択される構造を含む場合は,変換されるプログラムが毎度異なることになる.この内部変換処理はピクトグラミングシリーズ共通である.

4. 関連研究・考察

これまでにプログラムの構造をグラフィカルに表現する手法の提案やアプリケーションが開発されてきた[13].

岡本らは状態遷移図にもとづく対話型アニメーション作成ツールを提案しており[14]，Java 言語や JavaScript 言語で書かれたコードへの変換機能も有している．状態遷移図をコード化する際には，深いネストなどの構造上の複雑さが懸念されるが,Java 言語の内部クラスや JavaScript の関数ポインタを使用することで，この問題を克服している.しかし,情報遷移図上で構成素の画像自体と文字列によるアクション名で提示されるため，状態遷移に伴う具体的な視覚的変化がわかりにくい．Picuit では，表示領域全体をサムネイル化しノードとして表示するので，遷移前と遷移後のサムネイル画像を差分比較することで具体的な視覚変化をイメージさせている．一方，画面領域のサイズ制約から，全てのトリプルをつなげた連結グラフの形式では提示せず，個々のトリプルを「状態遷移リスト表示領域」にリスト化して提示している．トリプル群の表示形式については，利用評価を通じて今後改良していく予定である．また UML のステートマシン図など，別の図式の表示機能も追加予定である.

KidSim[15]， Viscuit[16]など図形書き換え型のプログラミング言語では，書き換えルールを図形で示し，適用することで，アニメーションの作成を可能とする．書き換えルールを列挙すればよい手軽さが長所である．Viscuit では，連続的な平面上において複雑な動きの遷移を可能としている．自由度が高くなると書き換えルールの定義自体の厳密さが要求されるが，書き換え規則に曖昧性を導入することで，この問題を克服している[17]．Picuit においても図形書き換えルールのトリプル定義を可能とすれば同様の仕組みは実現できると考えられるが，ピクトグラムでは，出力される人型ピクトグラムの姿勢や線画によるオブジェクトの形状の厳密性が求められるため，今回は採用を見合わせた．今後の検討課題とする.

ピクトグラムは，高等等学校情報科新学習指導要領の「情報 I(2) コミュニケーションと情報デザイン」の部分で,情報を抽象化する方法として取りあげられている[18]．さらに指導要領では，「情報に関する科学的な見方・考え方」を「事象を，情報とその結び付きとして捉え，情報技術の適切かつ効果的な活用（プログラミング，モデル化とシミュレーションを行ったり情報デザインを適用したりすること等）により，新たな情報に再構成すること」であると整理されている.本提案のようなピクトグラム作成を通じて,情報デザイン領域に限定されず，プログラミング等の単元と有機的に関連すれば，情報技術の適切かつ効果的に活用により新たな情報に再構成する手段となり得ると考えられる.

小学校プログラミング教育の手引（第三版）では,「多様な教科・学年・単元等において取り入れることや，教育課程内において，各教科等とは別に取り入れることも可能であり，児童がプログラミングを体験しながら，コンピュータに意図した処理を行わせるために必要な論理的思考力を身に付けるための学習活動を行う必要があります.」とある[19]．小学校第 5 学年算数「プログラミングを通して，正多角形の意味を基に正多角形をかく場面」が学習指導要領で例示されている．タートルグラフィックスによる描画が主になっているが,図 10 の(1),(2)に例示したように本アプリケーションは，正多角形を様々なパラダイムで描画する体験を可能とする.

近年，スマートフォンやデジタルサイネージの普及により，アニメーションピクトグラムへの需要が高まっている[20][21]．図 10 の(3)に示したように，差異の部分を比較提示することで,伝えるべき情報をより明確化できうる[22].このような新しい形式のピクトグラムコンテンツを創造し普及させていく一助にもなりうるだろう.

5. まとめ

本稿では，状態遷移モデルの学習環境「Picuit」について解説した．今後は，授業利用や評価実験を通じて，本アプリケーションの有効性を検証し，改善すべき点を分析・評価する.

参考文献

[1] 伊藤一成. ピクトグラミング - 人型ピクトグラムを用いたプログラミング学習環境 - 情報処理学会論文誌 教育とコンピュータ, 2018, vol. 4, no. 2, pp.47-61.

[2] K. Ito. Picthon A Learning Environment of Python through Pictogram Content Creation, IEEE FIE (Frontiers In Education), 2020, pp. 1-6.

[3] K. Ito. Block Pictogramming -A Block-based Programming Learning Environment through Pictogram Content Creation-, IEEE Global Engineering Education Conference (EDUCON 2020), 2020, pp.1669-1673.

[4] https://pictogramming.org/

[5] Wing, J. M. Computational thinking. Communications of the ACM, 2006, 49(3), pp. 33-35.

[6] D. Isayama, M. Ishiyama, R. Relator, and K. Yamazaki. Computer Science Education for Primary and Lower Secondary School Students: Teaching the Concept of Automata. ACM Trans. Comput. 2017, Educ. 17, 1, Article 2, 28 pages.

[7] Bell, T., Alexander, J., Freeman, I., & Grimley, M. Computer science unplugged: School students doing real computing without computers. The New Zealand Journal of Applied Computing and Information Technology, 2009, vol. 13, no 1, pp. 20-29

[8] Susan H. Rodger, Jinghui Lim, and Stephen Reading. Increasing interaction and support in the formal languages and automata theory course. SIGCSE Bull.,2007, vol. 39, no. 3, pp. 58–62.

[9] Joshua J. Cogliati, Frances W. Goosey, Michael T. Grinder, Bradley A. Pascoe, Rockford J. ROSS, and Cheston J. Williams. 2005. Realizing the promise of visualization in the theory of computing. J. Educ. Resour. Comput. 2005, vol.5, no.2, 5–es.

[10] 兼宗進, 御手洗理英, 中谷多哉子, 福井眞吾, 久野靖. 学校教育用オブジェクト指向言語「ドリトル」の設計と実装. 情報処理学会論文誌プログラミング, 2001, vol. 42, no. 12, pp.78-90.

[11] 西田知博, 原田章, 中西通雄, 松浦敏雄. プログラミング入門教育における図形描画先行型コースウェアが学習に与える影響. 情報処理学会論文誌 教育とコンピュータ, 2017, vol. 3, no. 1, pp.26-35.

[12] K. Ito, Figure Drawing Method Based on Human Motion using Pictogramming, IEEE TALE (An International Conference on Engineering, Technology and Education), 2019.

[13] Glinert, Ephraim P., and Steven L. Tanimoto. "Pict: An interactive graphical programming environment." Computer 11,1984, pp. 7-25.

[14] 岡本秀輔, 鎌田賢, 中尾隆司. 状態遷移図にもとづく対話型アニメーション作成ツールの提案. 情報処理学会論文誌 プログラミング, 2005, vol. 46, no. 1, pp.19-27.

[15] D.C. Smith, A. Cypher, J. Spohrer. KidSim: programming agents without a programming language, Communications of the ACM, 1994, Col.37, No.7, pp.55-67.

[16] https://www.viscuit.com/

[17] Y. Harada and R. Potter. Fuzzy rewriting: soft program semantics for children, IEEE Symposium on Human Centric Computing Languages and Environments, 2003, Proceedings. 2003, pp. 39-46.

[18] 文部科学省 高等学校学習指導要領解説 情報編 http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/07/13/1407073_11.pdf （平成 30 年 7 月公開）

[19] 文部科学省 小学校プログラミング教育の手引（第 3 版） https://www.mext.go.jp/content/20200218-mxt_jogai02-100003171_002.pdf （令和 2 年 2 月公開）

[20] 北神慎司.動画形式の視覚シンボルの視覚的典型性に関する調査, 日本教育工学会論文誌, 2006, vol. 30, pp. 21-24.

[21] 石井幹大, 御家雄一, 伊藤一成. 人型ピクトグラムのアニメーション化とその理解度の分析と評価, 第 10 回データ工学と情報マネジメントに関するフォーラム DEIM2018, 2018

[22] 藤森誠, 伊藤一成, Martin J. Dürst, 橋田浩一. 差異表現に基づくピクトグラムの主題提示と認識度向上, 日本感性工学会論文誌, 2009, vol.8, no.3, pp.575-584.