Code Weaver: ミクストリアリティを利用したタンジブルなプログラミング学習ツールの研究（2）

坂本恋[†1]　大島登志一[†2]

概要：本研究は，児童が「プログラミング的思考」を身につけられるよう支援することを目的とする．コンピュータに不慣れでも気軽に使用可能なように，プログラムコードを物理的な実体として扱えるものとした．ユーザは，机上にカードを並べることでプログラムを組み，実行結果を確認し，それを修正することができる．本稿では，プログラミングの学習モデルを整理したうえでプロトタイプシステムの展示実験の考察を行い，それを踏まえたシステムの改良について論じる．

Code Weaver: A Programming Learning Tool with Tangible User Interface (2)

REN SAKAMOTO[†1]　TOSHIKAZU OHSHIMA[†2]

Abstract: This study proposes a new programming learning tool with a tangible user interface. We adopt a tangible user interface using mixed reality in order to alleviate the difficulties encountered by beginners. Users can control the movement of a tiny virtual spider which draws various shapes with its thread as it moves. It provides users with elemental concepts of programming including sequential, iteration and branch structure. In this paper, we propose a model of problem-solving ability that can be learned by programming and discuss the requirements of programming learning tools.

1. はじめに

2020 年度から実施される小学校学習指導要領[1]により，全国の小学校でプログラミング教育が必修となる．文部科学省の「小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）」[2]では，特定のコーディングを覚えることがプログラミング教育の目的なのではなく，「情報技術を手段として使いこなしながら，論理的・創造的に思考して課題を発見･解決し,新たな価値を創造する」資質・能力としての「プログラミング的思考」を育むことが目的であるとし，その必要性について説明している．

そうした，プログラミング教育への社会的要請に対し，子供でも学びやすいプログラミング学習ツールがいくつも開発されている．GUI を通して画面上のプログラムを操作することで，キーボードによるソースコード入力の必要性をなくしたもの[3]が現在の主流である．

また，タンジブルユーザインタフェース（TUI）を用いたツール[4]も提案されている．TUI を用いたものは，手で直接操作するという身体性の特長を活かして直観的な操作が可能であることから，少ない前提知識で使い始めることができ，試行錯誤も円滑に行える利点が期待されている．

本研究では，主に児童を対象ユーザとし，タンジブルなプログラミング学習ツールの開発を行なっている[5]．このツールでは，ユーザはバーチャルなクモ形キャラクタの動きをプログラムして図形を描くコンテンツに取り組む．コンピュータに不慣れでも気軽に使用可能なように，プログラムコードを物理的な実体として扱えるものとした．ユーザは，机上にカードを並べることでプログラムを組み，実行結果を確認し，その場でそれを修正することができる．

本稿ではまず，プログラミング学習ツールの開発における指針として「プログラミング的思考」のモデルを整理する．続いて，これまでに開発してきたプロトタイプシステムの展示実験も振り返り，本システムの要件を改めて整理する．最後に，それらの考察を踏まえて，体験と実装の２つの観点から，今後の改良の方向性について論じる．

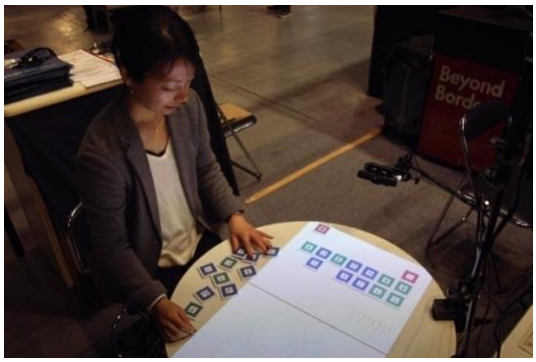

図 1　Code Weaver の体験の様子

†1 立命館大学 大学院 映像研究科
Graduate School of Image Arts, Ritsumeikan University
†2 立命館大学 映像学部
College of Image Arts and Sciences, Ritsumeikan University

2. 児童のためのプログラミングツール

2.1 ユーザの取り組むプログラミングタスク

小学校児童を主な対象ユーザとしたプログラミングツールについては，それぞれに工夫されたプログラムコンテンツやユーザインタフェースの提案がなされてきた．

プログラムの流れをそのアウトプットと分かりやすく関連づけ，ユーザが思考しやすい表現で提示する設計は，多くの研究に共通している．例えば Papert によって子ども向けに開発されたプログラミング言語 Logo[6]は，「前進」「回転」といった命令によって，画面上のキャラクタやロボットの動きをプログラムすることができる．ユーザの身体をキャラクタの身体と同調的に思考することができ，プログラムの流れを子どもでも思考しやすい[7]ため，さまざまなプログラミング学習ツールの実行結果に同様の表現が採用されている．

2.2 グラフィカルプログラミング言語

一般的なソフトウェア開発で主に用いられる，タイピングによってコーディングを行うテキスト型プログラミング言語は，熟練者によれば速く汎用性も高い．一方で，キーボードによるタイピングは，それ自体がスキルを要するタスクであるため，特に子どものようなタイピングに慣れていないユーザにとっては大きな負荷となる．Scratch[3]や Viscuit[8]は，マウスやタッチパネルなどの GUI を用いてグラフィカルにプログラミングを行うことで，キーボード入力を極力なくしている．また，iPad アプリ Swift Playgrounds[9]では，テキスト型のプログラミングを前提としつつ，入力予測機能である QuickType によって，タップ操作によるキー入力を支援する取り組みもなされている．

2.3 タンジブルプログラミング言語

画面上での操作ではなく，現実の物体をユーザの手で直接操作するタンジブルユーザインタフェース（TUI）を用いたプログラミング学習環境の研究も数多くなされている．TUI を用いたプログラミング学習環境は，マウスによるドラッグ&ドロップや各種コマンドなどの，コンピュータの恣意的な操作ルールをほとんど学ぶ必要がなく，現実空間の操作ルールをそのまま適用することができる点で，GUI より直観的な操作が可能であるといえる．また，プログラミングを行なっているユーザインタフェースが個人に占有されず，他の学習者や教師との共有空間に開かれることにより，学習状況の相互観察やコミュニケーションを促す効果も注目される．

TUI を用いたプログラミング学習環境の研究には，鈴木らの AlgoBlock[4]がある．これは，命令が割り当てられた複数の物理的なブロックを相互に接続することによりプログラミングを行うプログラミング言語である．TurTan[10]では，命令の割り当てられたオブジェクトをテーブル上に配置することによって Logo のようなグラフィクスをリアルタイムに描画する．また，Horn らの研究では，木製のオブジェクトを組み合わせてロボットの挙動をプログラムする Tern[11]を用いて，電子的部品を必要としないオブジェクトを用いた学習環境が提案されている．

プログラミングが必修として全ての子どもに学ばれるためには，手で直接操作するという身体性の特長を活かして少ない前提知識で，試行錯誤も円滑におこなえることが望ましい．また，学習過程でのコミュニケーションを増強するために，複数人で物理的な場を共有できるような学習ツールとすることが肝要である．

3. Code Weaver のアプローチ

3.1 「プログラミング的思考」のモデル

システムのデザインについて論じる前に，学習者が身に着けるべき「プログラミング的思考」のモデルを整理する．文部科学省の「小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）[2]」によれば，「プログラミング的思考」とは「自分が意図する一連の活動を実現するために，どのような動きの組み合わせが必要か，どのように改善していけばより意図した活動に近づくのかということを論理的に考えていく力の一つ」であるとされている．

著者らは，この説明を一般的なプログラミングの過程と対応させて解釈し，次のような過程で構成される活動として「プログラミング的思考」のモデル化を行なった．学習者は，解決すべき課題を定めた後，次のような過程で手順を具体化していく．

(1) 手順全体の大まかな流れを設計する
(2) 手順を小さなまとまりごとに構築する
(3) 小さなまとまりを組み合わせて手順全体を構築する
(4) 手順を実行し評価・修正する

これら 4 つの過程を循環しながら手順の完成度を上げていくこととなる．「プログラミング的思考」のモデルを図 2 に示す．このモデルの各過程は，Code Weaver では次のように関連づけられる．

まずユーザは，例示された図形もしくは描きたい図形を考え決め，以下の過程で手順を組み上げていく．

(1) 図形を描画するにあたって特に繰り返し現れる特徴に着目しながらプログラム全体の構造を考える．
(2) 特徴ごとに描画処理を構築する．
(3) 特徴ごとの描画処理を組み合わせてプログラム全体を構築する．
(4) プログラムを実行して描かれた図形を確認し，修正すべき内容を決める．

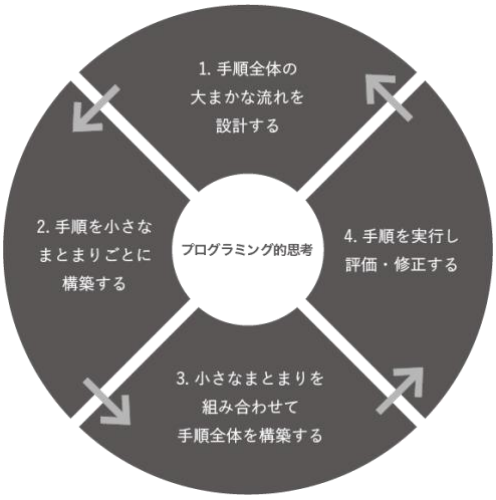

図 2　「プログラミング的思考」のモデル

3.2 デモ展示の考察

本研究では，2019 年 3 月にフランスのラヴァル市で開催された Laval Virtual 2019 にて，多くの一般ユーザを対象とした運用実験を行なった．その際，特にユーザの年齢を尋ねることにより，本システムの対象年齢について考察を行なった．

図 3　Laval Virtual 2019 での展示風景

【取り組む内容と理解度について】

開発当初より対象年齢を 10 歳以上と想定していたが，それはおおよそ妥当であることがわかった．10 歳未満の子どもにとっては「プログラミング的思考」よりも，むしろ幾何学的な概念，特に角度の単位などの知識が前提とされていることが本システムを体験する上での障壁となっていた．すなわち，対象とする学齢の知識レベルに合わせてコンテンツを設計する必要があるということである．本システムのコンテンツにおいては，ユーザの使用する命令に角度の単位を用いず，角度があらかじめ決められた「(直角に)右へ曲がる」などの命令を用意しておくことで，10 歳未満の子どもでもこのシステムを使えるように改良する方向性が考えられる．

また，主に 13 歳程度以上の兄姉や保護者がユーザと一緒に取り組み，思考の過程を共有したり助言したりすることで，同じ年齢のユーザと比較して理解が早まる様子が見受けられた．

【ユーザインタフェースについて】

プログラムを行うためのカード上に印刷された，プログラムの命令を表すピクトグラムは，言葉による説明がなくとも容易に理解されることがわかった．しかし「FORWARD（上向き矢印のピクトグラム）」を右向きに配置する体験者は，日本での展示の際にはほとんど見られなかったのに対し，今回の展示では右向き（まれに下向き）に配置する体験者が多く見受けられた．また，「RIGHT/LEFT（回転する矢印のピクトグラム）」も 90 度回転させた向きで配置する体験者も多く見られた．この結果より，カードの向きに関しては，文化的な違いによる解釈の相違の傾向なども考察した上で，配置についての許容範囲を広くする必要があることがわかった．図 4 にピクトグラムを印刷したカードを示す．

次に，デモ展示での考察も踏まえながら，システムデザインの現状と改良の方向性について述べる．

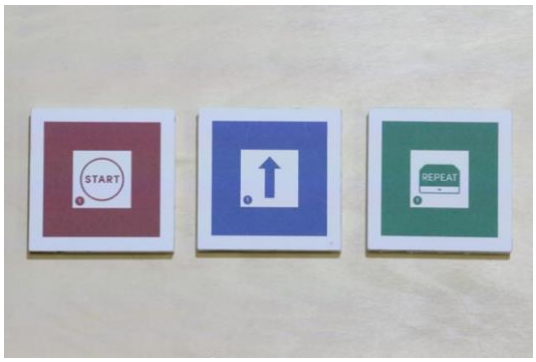

図 4　命令を表すピクトグラムを印刷したカード

4. システムデザイン

4.1 プログラミングコンテンツ

本研究では小学校児童を主な対象ユーザとし，「プログラミング的思考」を身につけられるよう支援することを目的とする．そこで今回は，「プログラミング的思考」を構成するコンピュータの処理を次の三つと定義した．すなわち，一つ一つ順番に命令を実行していく「逐次処理」，条件に応じて実行する処理を分岐させる「条件分岐」，条件のもとで同じ処理を繰り返す「反復処理」である．これらの処理を活用しながら，ユーザは「プログラミング的思考」を身に着けるプログラミングコンテンツに取り組む．そのため，本研究の扱うプログラムはノイマン型コンピュータによる逐次実行型のプログラムを想定している．すなわち，開始と終了が定義されていて，開始から終了にいたるまで逐次実行されるものである．本研究では，プログラムの開始と終了を，それぞれ開始命令と終了命令として割り当てる．開始命令に任意の命令を追加していくことでプログラミン

グを行い，最後に終了命令を追加することでプログラムが実行される．

ユーザの取り組むプログラミングコンテンツは, Logo と同様にキャラクタを動かして図形を描かせるというものであるが，運動の軌跡が線として残るという特徴を活かし，このシステムではバーチャルなクモ形キャラクタが糸をひきながら図形を描いていくものとした．キャラクタの動きを制御する命令として「前進」「右（左）回転」などを，プログラムの流れを制御する命令として「開始」「終了」「繰返し」などをそれぞれ用意した．本システムを利用したプログラムの実行例を図 5 に示す．

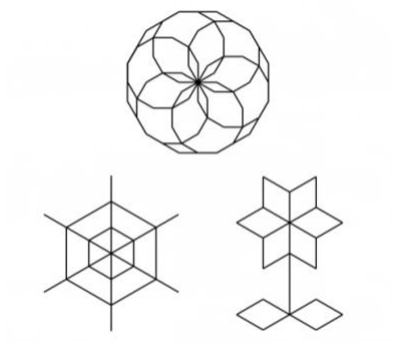

図 5　キャラクタによって描かれる図形の例

4.2 プログラムの構成要素

プログラムの「命令」は,「オペコード」と「オペランド」から構成される．オペコードは，命令の種類を表すものであり, オペランドは, 命令に関する数値を表すものである．

オペランドの内容はオペコードによって異なる. 例えば，キャラクタを前進させるオペコードには，距離を表すオペランドを組み合わせることができる．

本システムでは，以下のような種類の命令を使用する．

（1） 開始と終了

プログラムの開始と終了を，それぞれ「START」「GOAL」命令とする．

（2） 条件分岐

条件分岐を「IF」命令とする．真か偽の値をとるオペランドが組み合わされる．分岐の条件としては様々なものが考えられるため，プログラミングコンテンツに適した条件を検討しつつ，現在実装を進めている．

（3） 反復処理

反復処理の開始と終了を，それぞれ「REPEAT BEGIN」「REPEAT END」命令とする．REPEAT BEGIN には反復する回数の値をとるオペランドが組み合わされる．

（4） 前進

キャラクタの前進を「FORWARD」命令とする. 前進する距離の値をとるオペランドが組み合わされる．

（5） 右回転と左回転

キャラクタの右回転と左回転をそれぞれ「RIGHT」「LEFT」命令とする．マイナスとプラスがプログラムの本質ではないため，右回転と左回転とした．回転角の値をとるオペランドが組み合わされる．

（6） 関数

上記（1）から（5）までの任意の命令が組み合わされたまとまりを, 一つの命令として実行する命令を「FUNCTION」命令とする．プログラム全体を分割して小さなまとまりごとに構築するといった関数の考え方は,「プログラミング的思考」を身に着ける上で重要な役割を担うと著者らは考えたため,「FUNCTION」命令の実装を現在進めている．

4.3 ユーザインタフェースのデザイン

本研究では，プログラムコードを物理的な実体として扱えるタンジブルユーザインタフェースをデザインした．プログラムを構成する一つ一つの命令や数値が物理的なカードに割り当てられているため，ユーザはそれらのカードをテーブル上に並べることでプログラムを組み，実行結果を確認し，その場でそれを修正することができる．

システムは，まず始めに編集モードとなり,「START」カードをテーブルに配置することでプログラムの編集を開始することができる．続けて「前進」「回転」など，次に実行したい命令のカードを隣接して配置する. これを繰り返し，最後に「GOAL」カードを配置すると，システムは実行モードとなり,「START」カードから「GOAL」カードまでの間に配置されたカードの持つ命令を実行する．プログラムが実行されるとテーブル上でキャラクタが動き出し，その軌跡に沿って図形が描画される．システムのモード遷移図を図 6 に示す．

プログラミング操作とプログラム実行結果の確認を思考の中断なく円滑にできるように，ユーザがカードを配置する手元にプロジェクションによって描画結果を表示することとした．図 7 にユーザインタフェースを示す．

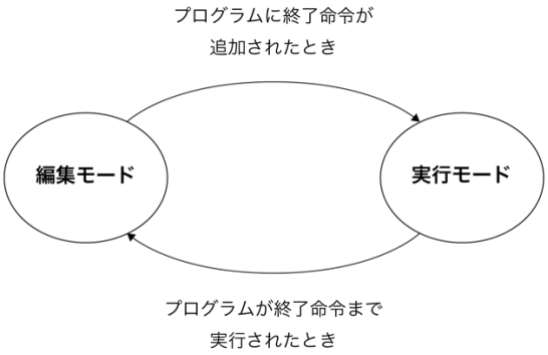

図 6　システムのモード遷移図

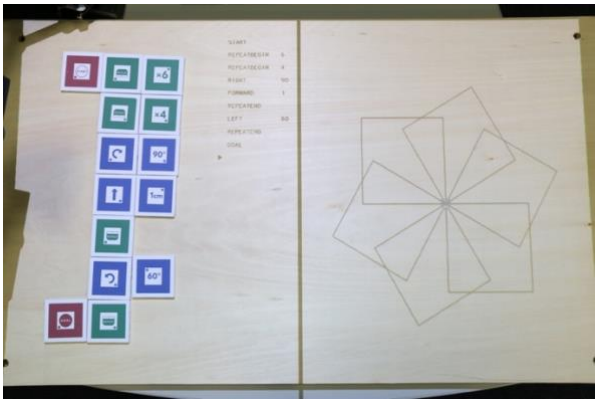

図 7 ユーザインタフェース

5. 実装

5.1 システム構成

図 8 に示すように，本システムは PC，カメラ，プロジェクタ，テーブル，マーカの描かれたカード群といった要素によって構成される．カメラとプロジェクタは PC に接続され，それぞれ三脚によってテーブルの周囲に固定されている．カメラはテーブル上のマーカを撮影し，プロジェクタはテーブルに映像を投影する．カメラが取得したマーカの写った画像は PC で処理され，マーカ同士の距離を計測するのに用いられる．マーカ同士の距離に応じて PC はプログラムを編集し，プログラムが実行されると，実行結果がプロジェクタによってテーブル上に表示される．

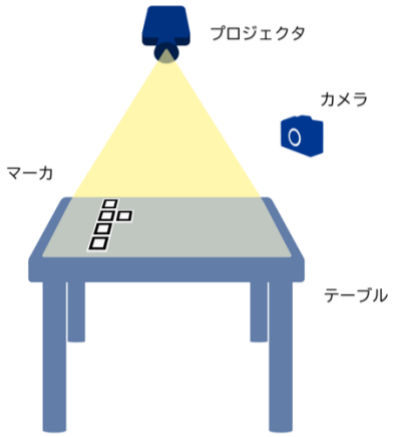

図 8　システムの構成

5.2 プロジェクションによる映像の提示

本システムでは，カードを配置するテーブルの下部にプロジェクタを設置し，そこから映像の投影を行うことで，カードによるプログラミング操作と映像による実行結果の確認を同時に可能にした．今回はキャラクタが図形を描く実行結果と，ユーザプログラムの状態を表す文字列を投影した．

マーカの位置情報取得にカメラによる画像認識を用いる関係上，マーカに人や物の影ができるとマーカが意図したとおりに認識されないことがある．テーブル上のマーカを置く領域に白色の映像を投影することによって，人や物の影をなくしマーカを鮮明に撮影することで，影による画像認識への影響を低減させている．

これまでに開発してきたプロトタイプシステムではプロジェクションをテーブルの上部から行なっていたが，プロジェクタを固定するためのアームによりシステム上部の空間に場所をとってしまうほか，照明環境によってはマーカの追跡が不安定になってしまうなどの課題があった．そのため，プロジェクタによる映像の投影とカメラによるマーカの追跡をテーブル下部より行うシステムを今回あらたに実装した．

プロジェクションにより，図形を描画するだけでなく，カードの周囲に情報を提示することもできる．ユーザがプログラミングを行なっている最中に，操作上のヒントや，エラーの原因となっている箇所を示すことなどを検討している．

5.3 画像認識によるユーザプログラムの入力

本システムは，カードに描かれたマーカをカメラで撮影・画像処理し，マーカ間の距離を計測することでカード同士がつながっているかどうかを判断している．その計測に ARToolKit を利用した．

カードの下面にはマーカが描かれており，カメラでリアルタイムに撮影したマーカ画像をコンピュータで処理することで，それぞれのカードの位置を認識することができる．マーカ同士の距離が閾値より短ければ，カード同士が繋がったとみなし，コンピュータ側のユーザプログラムで命令の追加を行う．また長ければ，カード同士が離れたとみなし，命令の削除を行う．

6. むすび

本稿では，プログラミング学習ツールにおける開発指針としての「プログラミング的思考」のモデル化と，プロトタイプシステムの展示実験の振り返りを通じて，本システムの要件を改めて整理した．この考察を踏まえて今後は次のような方向性で研究を進めていく．

（1） プログラムの中で扱う命令については，「プログラミング的思考」のモデルに基づいて，ユーザがプログラム全体を小さなプログラムに分けて思考できるように，関数を定義・実行できる機能を実装する．それと併せて，条件分岐を学ぶことができるようなプログラミングコンテンツを検討する．

（2） プログラム操作に用いるタンジブルなカードのデザインについては，ピクトグラムに対する文化的な意味の違いを考慮しつつ，言語や年齢を問わず幅広い対象のユーザがプログラミングを学ぶことができるカードデザインを検討する．

（3） カードのトラッキングについては，現在 ARToolKit を使用しているが，より多くのカードを安定して同時に使用できるようにするため，別のトラッキング手法も検討する．

（4）プロジェクションによる映像提示については，図形を描画するだけでなくカードの周囲に情報を提示することにより，操作上のヒントや，エラーの原因となっている箇所を示すなどの機能を実装する．

（5）学習教材としての観点では，描こうとする図形に繰り返し現れる特徴を示したり，そのために関数機能の使用を促したりするようなチュートリアルを用意する．

こうした総合的な改善に取り組みながら，実際の学校現場での評価と検証を行っていく．

謝辞 本プロジェクトに協力されている立命館大学映像学部の大島研究室各位に感謝します．本研究の一部は JSPS 科研費 16K00288 の助成を受けたものです．

参考文献

1) 文部科学省．小学校学習指導要領．文部科学省告示第六十三号．平成 29 年 3 月 31 日．
2) 文部科学省．小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）．小学校段階における論理的思考力や創造性、問題解決能力等の育成とプログラミング教育に関する有識者会議．平成 28 年 6 月 16 日．
3) Maloney, J., Resnick, M., Rusk, N. and Eastmound, E.. The scratch programming language and environment. *ACM Transactions on Computing Education*, 2010, vol. 10, no.4, Article No. 16. doi>10.1145/1868358.1868363
4) Suzuki, H. and Kato, H.. Interaction-level support for collaborative learning: AlgoBlock—an open programming language. CSCL '95 The first international conference on Computer support for collaborative learning. 1995, pp. 349-355. doi>10.3115/222020.222828
5) 坂本恋，大島登志一. MR Code Weaver: 投影型ミクストリアリティによるタンジブルなプログラミング学習ツール，情報処理学会シンポジウムインタラクション 2018 論文集, 2018, pp. 895-899.
6) Papert, S.. On making a theorem for a child. *ACM '72 Proceedings of the ACM annual conference*, 1972, vol. 1, pp. 345–349.
7) シーモア・パパート『マインドストーム―子供，コンピューター，そして強力なアイデア』未来社, 1995.
8) Harada, Y. and Potter, R.. Fuzzy rewriting: soft program semantics for children. IEEE Symposium on Human Centric Computing Languages and Environments, 2003, Vol.1, no. 1, pp. 39-46.
9) “Swift Playgrounds”. www.apple.com/swift/playgrounds/, (参照 2019-8-2).
10) Gallardo D., Juliá C. and Jordá S.. TurTan: A tangible programming language for creative exploration. IEEE International Workshop on Horizontal Interactive Human Computer System, TABLETOP 2008, 2008, pp. 89-92.
11) Horn M., Solovey E., Crouser R. and Jacob R. Comparing the use of tangible and graphical programming languages for informal science education. CHI '09 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2009, pp. 975-984.