■学前教育理论

儿童编程工具发展历程、特征及对未来教育路向的影响

陈 翠[1]，凌晓俊[2]，郑渊全[3]

（1.苏州科技大学教育学院，江苏苏州 215000；2.吉林师范大学教育科学学院，吉林四平 136000，
3.台湾清华大学竹师教育学院，台湾新竹 300141）

摘 要：从1960年代LOGO的诞生开始，儿童编程经历了准备期、发展期、实施期、停滞期、复兴期五个时期的发展。总体特征是从复杂语法到简单语法、从抽象系统到可视化系统、从文本编程到非文本编程、从软件编程到物理编程。编程工具的不断发展和完善，让儿童学习编程成为可能。随着数字化社会的到来，编程教育将逐渐成为儿童教育中不可或缺的一部分，并将向下延伸到小学和幼儿园，其教育路向很可能朝学校化、普及化、低龄化发展。

关键词：儿童编程工具；发展历程；特征；教育路向
中图分类号：G623.6,G202,TP311.56 **文献标识码**：A **文章编号**：2095-770X(2020)09-0068-06
PDF获取：http://sxxqsfxy. ijournal. cn/ch/index. aspx **doi**：10. 11995/j. issn. 2095-770X. 2020. 09. 010

The Development Course and Characteristics of Programming Tools for Children and Its Influence on Future Education Direction

CHEN Cui[1], *LING Xiao-jun*[2], *CHENG Yuan-chuan*[3]

(1. *School of Education, Suzhou University of Science and Technology, Suzhou* 215000, *China;*
2. *School of Education Science, Jilin Normal University, Siping* 136000, *China;*
3. *School of Education, National Tsing Hua University, Hsinchu* 300141, *China)*

Abstract:Since the birth of the LOGO in the 1960s, children's programming has experienced five stages of development: preparation period, development period, implementation period, stagnation period and revival period. The overall features range from complex syntax to simple syntax, from abstract systems to visual systems, from text programming to non-text programming, from software programming to physical programming. The development and perfection of programming tools make it possible for children to learn programming. With the advent of the digital society, programming will gradually become an indispensable part of children's education. Programming education will be extended down to primary schools and kindergartens, and the path of programming education is likely to be school-based, popularized and developed at a younger age.

Key words:programming tools for children; development course; characteristics; education direction

21世纪正在被新技术和我们尚未想像的新科技所包围，人工智能、新能源科技、无人驾驶、MR混合现实以及各种"黑科技"，正在构成人们现在和未来生活的真实环境。儿童作为在数字世界中成长的一代"数字原住民"(digital natives)，经历着科技的迅速发展、各领域的迅速更新、就业形式的层出不穷，势必需要掌握新的数字技术和素养，才能为将来更好的生活做准备。许多国家都在推动儿童编程的学习，英国、美

收稿日期：2020-05-23；修回日期：2020-06-03
基金项目：吉林省教育厅"十三五"社会科学项目(JJKH20180796SK)
作者简介：陈翠，女，山东枣庄人，苏州科技大学教育学院讲师，博士，主要研究方向：儿童编程；凌晓俊，女，安徽阜阳人，吉林师范大学教育科学学院讲师，硕士，主要研究方向：幼儿科技；郑渊全，男，台湾台北人，博士，台湾清华大学竹师教育学院教授、博导，主要研究方向：教育科技，教育政策。

国、芬兰、以色列等二十多个欧美国家已将编程不同程度地纳入中小学和幼儿园课程[1][2]。2017年7月，我国国务院印发的《新一代人工智能发展规划》中，明确提出在中小学阶段设置人工智能相关课程，逐步推广编程教育。

编程教育正成为儿童教育中不可或缺的一部分，然而，国内许多儿童编程系统的开发者并不完全了解编程发展历程和编程工具的类型，致使编程系统的开发走上了人云亦云的弯路，或者教师无法根据儿童年龄、发展特征和教学目标选择合适的编程工具，导致儿童学习效率低下。典型的例子如近年来市场上大量研发和生产的按键类机器人，尽管可以作为入门级编程工具，但其缺陷非常明显，凸显在儿童编写的程序一旦出现错误就需要从头开始重新编写，无法对程序进行调试(debug)，影响学习环节和学习效果。还有一些编程工具过于重视娱乐功能，忽略了儿童实际编程能力的培养。因此，本文对儿童编程发展历程和特征进行梳理和总结，并进一步厘清儿童编程教育的未来路向，以期为儿童编程的研究、工具的开发和教育策略的制定提供理论上的依据。

一、儿童编程工具发展历程

近六十年来，为了将编程介绍给年龄较小的儿童，研究人员进行了不懈的努力，不断开发适合儿童编程的界面、系统和玩具。1960年代末，美国著名计算机科学家西蒙·派珀特博士(Seymour Papert)发明了第一款专为儿童设计的编程语言“LOGO”。自此，儿童编程正式成为一个领域并开始了其发展历程。根据Higginson的理论，儿童编程可以划分为五个不同的阶段：准备、发展、实施、停滞和复兴，这是国际上首次对儿童编程发展历程进行明确划分[3]。但Higginson仅仅提出五阶段理论，并未对每个阶段的发展历程和特征进行描述，下面是在Higginson五阶段理论基础上对儿童编程发展历程的详述和总结。

(一)准备期：LOGO语言开发，为儿童打开编程之门

1960年代中期，美国著名数学家、计算机科学家、麻省理工学院人工智能实验室创始人Papert带领其团队开发适合儿童的编程语言，经过近5年的努力，LOGO编程语言问世，这是学术界公认的第一款专门为儿童设计的编程语言。在LOGO出现之前，就已经存在许多编程语言，例如TRAN、PASCAL、BASIC、SMALLTALK和LISP[4]，这些语言复杂的语法对读写能力尚不熟练的儿童来说非常困难。Papert认为儿童不应该被读写能力和复杂的语法规则所限制，只要有适合儿童发展水平的媒介(media)，他们有能力编写电脑程序[5]。LOGO语言简化了语法，是一种与自然语言非常接近的编程语言，儿童只要使用一些生活中常见的词汇，如“FORWARD”、“BACKWARD”、“LEFT”、“RIGHT”等，就可以编写程序[6]。LOGO还改变了儿童与计算机的沟通方式，以往所有的编程语言都是人与计算机本身进行沟通，而在LOGO中，儿童直接对屏幕上的小海龟编程，而不是对处理器(processor)编程。如此，增加了儿童对编程的兴趣，儿童用LOGO语言告诉海龟向哪走，在对海龟发出命令的同时会假想自己是小海龟，这种比拟和想像使得儿童的编程学习更加具体化。

LOGO改写了儿童学习编程的方式，奠定了儿童编程的基本理念和基本原则，此后出现的儿童编程系统几乎都基于LOGO的理念进行设计[7]。但是LOGO也有局限性：一方面，LOGO并没有完全改变传统的编程方式，仍然是基于文本的计算机语言；另一方面，LOGO中涉及的许多概念，例如变量(variable)或递归(recursion)，对较小的儿童来说比较困难，因此LOGO较少用于6–7岁以下的儿童[8]。

(二)发展期：非文本编程系统开发，为学龄前儿童打开编程大门

为了让较小的儿童也能学习编程，1970年代中期，麻省理工学院人工智能实验室研究员Perlman与Papert合作，在LOGO环境的基础上创建了第一个非文本编程系统——TORTIS (Toddler's Own Recursive Turtle Interpreter System)。TORTIS使用物理模块进行编程，为3–6岁的幼儿打开了编程的大门。TORTIS系统中最著名的是Slot Machine，由三种颜色的插槽机架组成，可以插入三种卡片：数字卡、动作卡、条件卡。在每个机架的左侧是一个“Do it”按钮，按下按钮，将程序传递给计算机，计算机再控制海龟按照顺序执行机架中每张卡片的动作。研究表明，4岁的幼儿就可以用Slot Machine学习编程，他们能够理解系统在做什么，也能解释正在发生的事情，可以设计指定的程序[9]。虽然Slot Machine应用非常简单，但是造价昂贵，很少在研究和实验室之外使用[10]。不过，TORTIS最大的贡献在于创造性地提供了一种思路：编写程序不一定需要文本，也可以使用物理实体，这标志着儿童编程“非文本”时代的到来。

(三)实施期：玩具公司参与编程开发，让儿童编程实施成为可能

1980年代以前，由于计算机并未普及，很少学校有条件让儿童学习编程。1980年代以后，一批玩具公司参与到编程工具的开发中，出现了一批造价相对较低的编程工具，让儿童编程在教育界的实施成为可能。

最有代表性的是乐高公司的“可编程积木”(Programmable Bricks)。为了让LOGO和TORTIS的理念广泛应用于儿童编程，1980年代，丹麦乐高公司和美国麻省理工学院的媒体实验室(MIT Media Lab)合作，进行LEGO/ Logo项目即“可编程积木”的开发[11]。LEGO/ Logo是一个基于计算机的学习平台，它将乐高积木与LOGO编程语言结合在一起。儿童用乐高积木建造自己的机器人，并使用LOGO对其编程[5]。几年以后，LEGO/Logo项目发展成为我们熟知的乐高头脑风暴系列机器人套组(Mindstorms robotics kits)。“可编程积木”的推出，让许多学校实施儿童编程教育成为可能，代表了80年代儿童编程领域的巅峰。

与此同时，美国游戏公司Milton Bradley发布的可编程电动汽车Big Trak[12]，以及英国Valiant Technologies公司推出的基于LOGO理念设计的Roamer机器人，让儿童透过操控机器人顶端的按键进行编程，首次将编程与计算机完全分离。儿童不需要通过任何计算机就能实现简单的程序设计，这为儿童编程工具的开发提供了崭新的方向。但其缺陷也很明显，一是其编程语言不够完整，缺乏一些重要的指令，例如分支指令(branching instructions)和流程控制指令(control flow instructions)等；二是无法调试或修改程序，一旦发生错误，只能从头开始设置指令。

(四)停滞期：反对儿童编程声音高涨，儿童编程几乎停滞

1980年代后期到1990年代末，儿童编程经历了一段时间的“消沉”。由于对多数人来说，编程只是一种狭隘的技术性活动，是一些有数学天赋的成年人掌握的一项复杂的技能，不是每个孩子都要成为程序员，因此，儿童没必要学习这样复杂的知识。儿童联盟(Alliance for Childhood)甚至强烈反对儿童使用计算机和科技产品，认为计算机和科技玩具对儿童创造力、想像力、注意力、智力和社会情感有不良影响[13]。因此在这长达10年的时间里，几乎没有创新性的儿童编程语言或编程环境诞生。唯一值得一提的是，1990年代中期，AlgoBlocks编程系统的出现让“有形编程”(tangible programming)的概念进入学术领域[14]。AlgoBlock发展了Big Trak和Roamer等玩具的物理编程的特性，将按键变成木块，儿童只需排列木块即可进行编程。但直到2000年，AlgoBlock有形编程的理念才得到世界各地实验室的重视。

(五)复兴期：编程工具蓬勃发展，编程教育得到重视

进入2000年以后，随着计算机、智能手机、平板电脑以及人工智能的相继发展，儿童编程工具持续创新，编程教育得到重视。世界各地的实验室和游戏公司开发了各式各样的编程系统，这些系统一般可以分为两类，一类是软件编程系统，一类是物理编程系统。

1.软件编程系统

软件编程系统中较有代表性的包括ToonTalk、Hopscotch、Kodable、WeDo2.0、Scratch、ScratchJr、Code Spells、Lightbo、Kinderlogo、Daisy the Dinosaur以及以Scratch语言为基础的Tynker等。其中，最受欢迎的是Scratch。Scratch于2006年首次发布，由Papert的学生Resnick领导的麻省理工学院媒体实验室的“终身幼儿园小组”开发[15]。Scratch提供了一个基于LOGO的友好界面，允许用户使用编程的基本概念如排序、重复循环和变量来编写游戏、故事和视频[16]。儿童通过拖曳预先设定好的积木式程序模块堆叠出指令，并且设置或控制角色及背景的动作和变化，从而完成程序设计[17]。随着Scratch的开发，儿童编程工具设计原则开始确立：

(1)低地板(Low floor)：工具必须足够直观，让新用户逐渐适应并有一定程度的信心；

(2)高天花板(High ceilings)：工具必须允许有经验的用户创建越来越复杂的程序；

(3)宽阔的墙壁(Wide walls)：该工具必须能够承载广泛的项目，能够让用户利用个人经验和流行文化的元素，设计和开发独特的程序，以代表他们自己的兴趣和背景[18]。

此后的儿童编程系统基本都遵循这个原则进行设计。

2.物理编程系统

物理编程系统主要有麻省理工学院媒体实验室开发的“Curlybot”[19]、塔夫茨大学开发的“KIBO”[20]，以及其他实验室开发的“有形编程砖系统”(Tangible Programming Bricks system)[21]、“Electronic Blocks”[22]、“GameBlocks”、Beebot、Blue-Bot、Cubetto和Pixie等。其中，最有代表性的是KIBO，KIBO专为4至7岁的幼儿设计，由两个部分组成：木制编程块和机器人主体。木制编程块是KIBO的程序语言，包括动作、重复循环、条件、事件等指令。机器人主体具有嵌入式扫描仪，用户透过扫描编程块上的条形码让机器人执行程序[23]。KIBO成功解决了物理编程工具普遍存在的“太简单”和“省略一些重要编程概念”的问题[24]，让4-7岁的儿童能够真正学习编程概念[25]。

二、儿童编程工具发展特征

整体来说，儿童编程工具朝“文本简化”或“非文本化”、具体化、可视化、手动操作化等方向发展，编程工具难度不断简化，避免繁琐的语法和代码，有效降低了儿童学习编程的年龄和门槛。具体来说，儿童编

程工具的发展具有以下四个特征。

（一）从复杂语法到简单语法

在LOGO出现之前，在美国学校里较广泛使用的是专为初学者设计的BASIC语言，其复杂的语法规则让许多儿童望而却步，LOGO简化了编程语法，使用生活中常见的词汇让儿童学习编程。近期的图形化可视编程语言Scratch，进一步简化了语法，儿童只需要掌握基本的“WHILE”、“REPEAT”、“IF”等语法就可以编写程序。再到ScratchJr几乎不需要使用语法，ScratchJr编程语言由“编程块”（programming blocks）组成，儿童可以通过拖动和排列块序列来创建程序，编程块的设计确保儿童不会在ScratchJr中产生语法错误。语法的简化让儿童更能在简单、易用和有趣的数字环境中学习编程和计算思维。

（二）从抽象系统到可视化系统

早期编程语言以文本化编程为主，其优势是拥有成熟的语法和系统架构、规范的编码规则与灵活的操作性，但缺陷是过多的抽象概念规则。可视化编程系统则有效解决了这个问题。可视化编程系统向用户直接呈现可视图像（visual images）和图标（icons），抽象的文字和符号被图片取代，儿童只需拖动和连接计算机屏幕上的图标就可进行编程[26]。目前，可视化编程环境可分为两类，一类是以Scratch为代表的图形嵌套文本的模式，当今大部分针对儿童编程的语言都是这种形式，例如著名的编程语言ToonTalk和非常有影响力的Alice、ROBOLAB、Play-i、CodeSpells、TangibleK、Hopscotch、Google Blockly、Tynker等，这类编程系统是具备基本阅读能力的儿童学习编程的入门首选；另一类是以ScratchJr为代表的完全图形模式，此类编程环境还包括Lego的WeDo 2.0，Wonder Workshop推出的Dash & Dot，专为iPad设计的编程游戏Kodable等，图形内用符号或数字来表示指令，儿童不需具备基本的阅读能力。

（三）从文本编程到非文本编程

基于文本的编程语言是最传统也是应用最广泛的编程语言，常用的语言包括LOGO以及基于Ruby语言的Hackety Hack等。但是对于读写能力和抽象概念规则能力较弱的儿童来说，文本编程语言中细碎的语法规则很容易分散其注意力，导致无法理解程序结构，基于文本的计算机语言是儿童早期学习编程的严重障碍。因此，非文本的编程应运而生。非文本编程包括可视化编程、物理实物编程和不插电编程（unplugged programming），这些编程方式不需要儿童具备文本阅读能力，只需要认识基本的符号和数字即可。非文本化的编程有效降低了儿童学习编程的阅读门槛和年龄门槛。

（四）从软件编程到物理编程

软件编程环境虽然更新迭代并且不断完善，但屏幕和键盘依然是较小儿童学习编程的障碍。近年来，为了让3-6岁的儿童也能学习编程，物理编程环境得到教育界的重视，使用物理对象来表示各种编程元素、命令结构和控制流程结构，儿童只需将这些物理对象进行连接即可形成完整的程序。最常见的是使用有形的物理块代表程序指令，由实际的机器人或者其他物体来执行程序，如Beebot、Big-Track、Pixie、Curlybot、LittleBits、KIBO等。基于现实的互动是有形编程中最重要的元素，整个编程过程中无需使用计算机和屏幕，一是解决了许多教室没有足够电脑的问题，二是可以管理屏幕时间（screen time），调节滥用和过度使用屏幕的问题[27]。因此，物理编程工具非常适合年幼的儿童，并且使合作学习成为可能[6]。

综上，为了让儿童有效学习编程，需要进行工具的准确选择，教育者要能够灵活地根据特定情况选择最合适的工具。例如，对于较小的儿童来说，物理编程系统更有利，而较大的儿童，可视化编程系统更适合。在课堂环境中，物理编程界面更有利于全班性活动——学生可以坐在一个圆圈中并合作使用物理编程工具，这些活动可以在不同的地点进行，而不需要学生聚集在大型显示屏或投影屏幕周围[28]。教育者也可以在工具的选择上为学生提供分层支架，学生可以从物理系统开始，随着年龄和认知能力的增长，再转换到具有增强功能和复杂性的可视化系统。此外，有些编程概念较为复杂，比如“重复循环”、“条件分支”等，为了帮助儿童理解这些概念，教育者也可以结合课堂活动进行不插电的编程游戏，让儿童的学习更加具体化。

三、儿童编程的未来教育路向

由于编程工具的不断完善，儿童进行编程学习已经不存在技术上的问题，未来，随着数字化社会的不断推进、人工智能在各领域的应用，具备基本编程能力将可能成为各行业的基本要求。编程工具的发展和完善使得编程教育具备了向下延伸到小学和幼儿园的可能，因此，儿童编程的未来教育路向有向学校化、普及化、低龄化发展的趋势。

（一）学校化

2014年9月，英国将计算机列入国家课程框架，其中，关键阶段1（Key Stage 1）要求5-7岁的儿童需要了解什么是算法（algorithms），并能够创建和调试（debug）简单的程序[29]。英国的这一举措在欧洲掀起了一股课程改革的热潮，目前，包括法国、芬兰、以色列、

西班牙等20个欧洲国家已将编程纳入义务教育课程中[30]。美国也是较早制定儿童编程教育政策的国家之一，2016年，奥巴马政府推出“全民计算机科学计划”（Computer Science for All），将计划纳入每一个教育层面，旨在使从幼儿园到高中的所有美国学生学习计算机科学，并为各州提供40亿美元的资金，通过培训教师、发展高品质教学资源和建立有效的区域合作伙伴关系来提升K-12计算机科学教育[2]。

我国在2017印发的《新一代人工智能发展规划》中，提出在中小学阶段逐步推广编程教育。目前，一些中小学已经设置了编程课程，未来，编程课程很可能和语、数、外一样重要，成为中小学必修课程之一。

（二）普及化

随着适合儿童的编程工具的开发和各种免费网站的投入使用，儿童编程将变得越来越普及，学习编程再也不需要高昂的花费，运用网上的免费资源，儿童就可以在简单的指导下学习编程。美国和欧洲已经提供了非常多的编程学习网站。例如，美国的Code.org（http://code.org/）和Code Academy（http://www.codecademy.com/）等国际组织正在通过交互式在线课程为儿童提供基本编程概念的学习[16]。欧洲也建立了类似于美国的Code.org网站，如“Digital Single Market”（https://ec.europa.eu/digital-single-market/en）或者“all you need is {C<3DE}”（http://www.ally-ouneediscode.eu），为学习者提供广泛的编程学习资源。另外，ScratchJr（https://www.scratchjr.org），Scratch（https://scratch.mit.edu），CodeCombat（https://code-combat.com）、Code Monster（http://www.crunchzilla.com/code-monster）均提供免费的学习资源，儿童可以在网站学习编程、创作程序，并在网上社区分享自己的作品。这些免费学习资源的出现将使儿童编程学习的普及化成为可能。

（三）低龄化

新编程工具的出现让儿童早期编程学习成为可能。例如，ScratchJr通过使用拖放块和符号系统，让年仅5岁的儿童就可以编程[7]；而美国塔夫茨大学的研究表明，3岁的儿童就可以使用KIBO机器人创建语法正确的程序[31]。由此，新加坡政府发起了“Play-Maker Programme”计划，旨在让年幼的儿童接触科技，并为4-7岁的儿童提供科技产品（ScratchJr、KIBO、乐高WeDo2.0等），让他们以适合其发展的方式学习科技、编程等重要概念，并发展解决问题的能力[32]。因此，从科技社会的现实来看，科技是幼儿生活的一部分，编程是21世纪的新素养，是未来社会的一项重要技能，甚至是未来沟通和表达的重要方式，编程教育将朝低龄化的方向发展，儿童早期学习编程，不仅仅是对编程概念的认知，更重要的是计算思维的发展。

四、结语

从1960年代LOGO诞生以来，儿童编程工具经历了五个时期的发展，取得了三次跨越性的进步：第一次是编程语法的简化，为儿童打开了编程学习的大门；第二次是变文本编程语言为图形编程语言，让较小的儿童也能学习编程；第三次是物理编程系统的发明，让儿童早期学习编程成为可能。总体来说，儿童编程系统朝非文本化、具体化、可视化、手动操作化的方向发展，编程工具的难度不断简化，有效降低了儿童学习编程的门槛。由于编程工具的不断完善，儿童进行编程学习已经不存在技术上的问题，未来，儿童编程教育的路向很可能将朝向学校化、普及化、低龄化发展。为了让儿童有效学习编程，需要进行工具的准确选择，教育者和研究者要能够从易用性、儿童偏好、学习表现、协作能力等方面选择最合适儿童的编程工具。

［参考文献］

[1] European Schoolnet. Computing our Future: Computer Programming and Coding Priorities, School Curricula, and Initiatives across Europe [EB/OL], http://www.eun.org/resources/detail?publicationID=661.

[2] White House. Computer science for all [EB/OL], Retrieved from https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all.

[3] Higginson, W. From Children Programming to Kids Coding: Reflections on the Legacy of Seymour Papert and Half a Century of Digital Mathematics Education[J]. Digit Exp Math Educ 2017, 3: 71-76.

[4] Papert, S. Mindstorms: Children, computers, and powerful ideas[M]. New York: Basic Books,1980.

[5] Blikstein, P. Computationally enhanced toolkits for children: Historical review and a framework for future design [J]. Foundations and Trends in Human-Computer Interaction, 2015, 9(1): 1-68.

[6] McNerney, T. S. From turtles to tangible programming bricks: explorations in physical language design[J]. Personal and Ubiquitous Computing, 2004, 8(5): 326-337.

[7] Geist. E. Robots, programming and coding, Oh my![J]. Childhood Education, 2016, 92(4): 298-304.

[8] Wyeth, P., & Purchase, H. Designing technology for children: Moving from the computer into the physical world with Electronic Blocks [J]. Information Technology in Childhood Education Annual, 2002, 1: 219-244.

[9] Perlman, R. Using computer technology to provide a cre-

ative learning environment for preschool children [R]. Logo Memo 24, Cambridge, 1976, MA: MIT Artificial Intelligence Laboratory Publications 360.

[10] Bers, M. U., & Horn, M. S. Tangible programming in early childhood: Revisiting developmental assumptions through new technologies [A]. In I. R. Berson, & M. J. Berson (Eds.), High-tech tots: Childhood in a digital world?Greenwich [C]. CT: Information Age Publishing, 2009: 1-32.

[11] Cejka, E., Rogers, C., & Portsmore, M. (2006). Kindergarten robotics: Using robotics to motivate math, science, and engineering literacy in elementary school [J]. International Journal of Engineering Education, 22(4), 711-722.

[12] Horn, R. V. Marvelous toys and educational robots [J]. Phi Delta Kappan, 2005, 86(5): 408-409.

[13] Campaign for a Commercial-Free Childhood, Alliance for Childhood, & Teachers Resisting Unhealthy Children's Entertainment [R]. Facing the screen dilemma: Young children, technology and early education, 2012. Boston, MA: Campaign for a Commercial-Free Childhood; New York, NY: Alliance for Childhood.

[14] Suzuki, H., & Kato, H. Interaction-level support for collaborative learning: AlgoBlock an open programming language [A]. In Proceedings Computer Support for Collaborative Learning CSCL'95 [C]. NJ: Lawrence Erlbaum Associates.1995: 349-355 Hillsdale.

[15] Horn, M. S., & Jacob, R. J. K. Designing tangible programming languages for classroom use [R]. In Proceedings of the 1st international conference on Tangible and embedded interaction, New York, 2007: 159-162.

[16] Portelance, D. J., Strawhacker, A., & Bers, M. U.Constructing the ScratchJr programming language in the early childhood classroom[J]. International Journal of Technology and Design Education, 2016, 26: 489-504.

[17] Bers, M. U. Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom[M]. New York, NY: Routledge press, 2018.

[18] Resnick, M., & Silverman, B. Some reflections on designing construction kits for kids [R]. In Proceedings of the 2005 Conference on Interaction Design and Children, New York: ACM, 2005: 117-122.

[19] Frei, P., Su, V., Mikhak, B., & Ishii, H. curlybot: Designing a new class of computational toys [A]. In T. Turner & G. Szwillus (Eds.), Proceedings of the SIGCHI Conference on Human Factors in Computing Systems [C]. New York: ACM Press, 2000: 129-136.

[20] Bers, M. U. Coding and computational thinking in early childhood: The impact of ScratchJr in Europe[J].European Journal of STEM Education, 2018, 3(3): 1-13.

[21] McNerney, T. S. From turtles to tangible programming bricks: explorations in physical language design[J]. Personal and Ubiquitous Computing, 2004, 8(5): 326-337.

[22] Smith, A. Using magnets in physical blocks that behave as programming objects[R]. In Proceedings First International Conference on Tangible and Embedded Interaction, TEI'07, Baton Rouge, LA: ACM Press, 2007: 147-150.

[23] Sullivan, A., & Bers, M. U. Dancing robots: Integrating art, music, and robotics in Singapore's early childhood centers [J]. International Journal of Technology and Design Education, 2018, 28(2): 325-346.

[24] Kwon, D. Y., Kim, H. S, Shim, J. K., & Lee, W. G. Algorithmic Bricks: A tangible robot programming tool for elementary school students [J]. IEEE Transactions on Education, 2012, 55(4): 474-479.

[25] Sullivan, A., & Bers, M. U. Robotics in the early childhood classroom: Learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade[J]. International Journal of Technology and Design Education, 2016, 26: 3-20.

[26] Sapounidis, T., Demetriadis, S. N., & Stamelos, I. Evaluating children performance with graphical and tangible robot programming tools [J].Personal and Ubiquitous Computing, 2014, 19: 225-237.

[27] Sullivan, A., Elkin, M., & Bers, M. U. KIBO Robot Demo: Engaging young children in programming and engineering [R]. In Proceedings of the 14th International Conference on Interaction Design and Children (IDC'15). ACM, Boston, USA, 2015.

[28] Horn, M. S., Crouser, R. J., & Bers, M. U. Tangible interaction and learning: The case for a hybrid approach [J]. Personal and Ubiquitous Computing, 2011, 16(4): 379-389.

[29] U.K. Department for Education. The national curriculum in England: framework document[OL], 2014. http://dera.ioe.ac.uk/21041/4/national_frame-work_with_draft_ks4_science.pdf.

[30] Bocconi, S., Chioccariello, A.,& Earp, J. The Nordic approach to introducing Computational Thinking and programming in compulsory education [EB/OL], https://doi.org/10.17471/54007

[31] Elkin, M., Sullivan, A., & Bers, M. U.Programming with the KIBO Robotics Kit in Preschool Classrooms [J]. Computers in the Schools, 2016, 33(3): 169-186.

[32] Sullivan, A., & Bers, M. U. Dancing robots: Integrating art, music, and robotics in Singapore's early childhood centers [J].International Journal of Technology and Design Education, 2018, 28(2): 325-346.

[责任编辑 任丽平]