巧用 Scratch 作图培养小学生计算思维的实践探究

[摘要]文章从 Scratch 案例教学存在的一些问题出发，结合实例介绍了 Scratch 作图培养小学生计算思维的优势，探索在分析任务、程序实现、定义过程、个性创作等方面培养计算思维的方法与策略，为创作 Scratch 更复杂的作品奠定基础。

[关键词] Scratch 画笔 过程 编程 计算思维

Scratch 走进小学生课堂已有两三年，它是一种可视化的编程语言，可以制作动画故事、科学实验、趣味游戏等，深受学生的喜爱。对于小学生而言，Scratch 积木式的程序指令更容易学习、更好理解，有利于小学生掌握程序设计的基本方法，随着 Scratch 教学的深入开展，笔者在教学中经常会遇到这样一些现象：

现象一：跃跃欲试却无从下手

与电脑绘画、制作小报和设计演示文稿等相比，小学生一开始普遍对 Scratch 编程兴趣浓厚。尤其是当老师演示一些用 Scratch 设计的小游戏，小故事时，同学们都表现出跃跃欲试的样子。但是当真正开始编写程序时，很多学生却对着任务一筹莫展，不知如何入手。

现象二：反复调试却效果不佳

在设计制作复杂案例的时候，随着角色的增多，每个角色任务功能的增加，程序必定越来越复杂。学生在程序的调试过程中，对某些参数和指令反复修改，但却将程序越改越乱，久而久之出现畏难的心理，不愿继续自主探究。最后的作品无论从功能上还是细节处理上效果都打了折扣。

现象三：学会案例却创意不足

目前的 Scratch 教学一般都以案例作为载体，在老师的带领下，分析案例，编写指令、调试运行，学生基本都比较顺利地能完成任务。然而，每当让学生自主设计创作时，很多同学都难以跨越已有学习经验禁锢。

透过上述这些现象，分析其中的原因所在，深入思考 Scratch 教学的本质，笔者发现 Scratch 图形化的指令直观明了，学生可以很快掌握这些指令的作法，编写简单的小程序。但随着学习的深入，任务变得复杂，角色开始增多，每个角色的功能增加，用到的指令越来越多，程序也越来越复杂，这时就需要运用计算机科学的基础概念去求解问题和设计系统，利用递归思维和启发式推理来寻求解答，也就是在 Scratch 教学中要培养小学生的计算思维。

一、培养小学生计算思维的设想

在编程创作中，将程序划分为小部分并逐一解决，是一种非常重要的思维方式。Scratch2.0 的“更多模块”中的“新建功能块”指令的使用能很好的解决程序的模块化，可以提高程序在结构化和组织上的灵活性；它真正将 Scratch 带到编程思维的创作中。由于它的作用等同于过程函数，为了描述方便，以下都称之为过程函数。

但是想要学生在创作互动游戏等作品中直接理解并正确使用过程函数进行模块化编程，存在一定的困难。由于几何图形能简单简单直观的体现逻辑，所以以创作简易的几何图形为主题，带领学生进行对设计变量、函数、递归等概念的编程思维训练，会比较直观易懂。这种图形看上去复杂，其实一分解，就只有一个最基本的图形，经过程序迭代之后自动生成，图形的形成都具有一定的规律性。

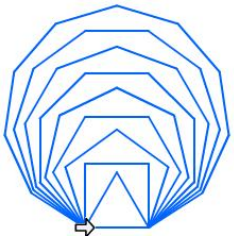


图 1 多样多边形

图 2 蜘蛛网

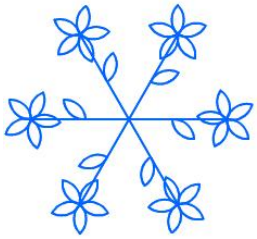


图 3 美丽的花团

Scratch 软件中自带有画笔模块，任何一个舞台中的角色都可以作为一只画笔，这支笔有两种重要的状态：抬笔和落笔。当角色当前状态下是落笔，那么角色移动时，它就会按照画笔的属性（颜色、大小、色度等）绘制出移动轨迹；反之，则不会留下痕迹。画笔的属性和 Scratch 中其他的指令还增加了 Scratch 作图的趣味性和互动性。

因此，笔者利用 Scratch2.0 作图对培养计算思维进行了实践研究，希望通过在 scratch2.0 中绘制各种创意图形的案例让学生体验问题的解决方法，为创作其他 scratch 作品打下编程的基础。

二、培养小学生计算思维的策略

计算思维是当前国际计算机界和教育界较为关注的一个重要的概念。根据计算思维包含的算法思维、评估、分解、抽象、概括等要素。结合 Scratch 教学中发现的问题和小学生的认知水平，笔者认为利用 Scratch 作图的创作过程中主要培养小学生分解、递归等算法思维、概括、推理等方面计算思维。

1. 分析任务：由外到内逐层分解

充分理解案例，学会分析任务是解决任何编程问题的第一步。理解之后我们可以做出一个大致的解决方案，然后将其划分为多个主要任务进行突破；几个主要的任务又可以各自作为大任务再分解成小任务进行解决。我们将这样的顺序称为由外到内逐层分解。在用 Scrctch 作图中，我们经常会用到这样的分析方法，这样的分析方法同样可以运用到其他类别的 Scratch 作品的创作中。

比如学生拿到《蜘蛛网》案例时（图 2），虽然直观感觉上对图形非常熟悉，也知道这个图形是一个特殊图形；但是由于缺乏计算思维的训练，学生在编程时仍旧表现出无从下手。这也正凸显了动手编程前任务分析的重要性。在程序实现之前，教师要引导学生对图形仔细观察，发现图形生成具有的规律，帮助学生通过一层一层的推理和分析。如蜘蛛网这个案例中，学生分析出了两种分解思路。

第一种（如图 4）：

A.这个蜘蛛网可以分解成 5 个六边形，这 5 个六边形的边长依次增大；

B.每一个六边形可以分解成 6 个大小一致的正三角形。

那么只要我们能画出正三角形，就能得到正六边形，将六边形增加边长，组成完整的蜘蛛网。

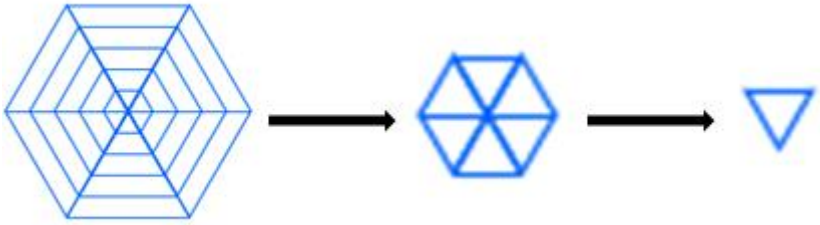


图 4 分析思路一

第二种（如图 5）：

A.这个蜘蛛网可以分解成六个大小一致的正三角形；

B.每个大的正三角形由 5 个边长依次增大的小正三角形组成。

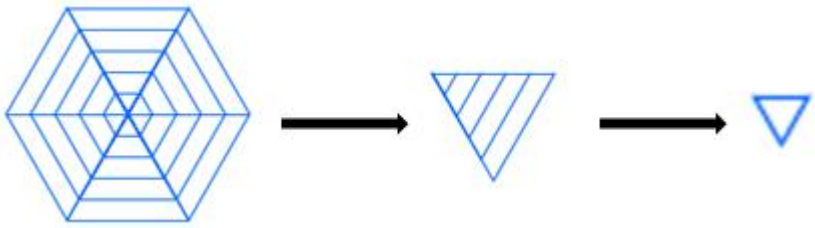


图 5 分析思路二

从案例中，可以发现对复杂问题由外到内逐层分解可以帮助学生化繁为简，体验到复杂问题到具体问题的分解过程。摆在学生面前的不再是无从下手的问题，那么学生不仅跃跃欲试，更敢于尝试。

2. 程序实现：由内到外逐层递归

通过前期由外到内的逐层分析，能够在复杂问题中分解出要解决具体小任务。集中解决好复杂问题中的小问题，把各个细节问题解决好，进行合理的组装，就能由内到外一步一步解决问题。因此，这个程序实现过程中，我们要把目光放小，着眼于具体的小任务。

在绘制蜘蛛网的案例中提到了两种分解过程。这里以第一种思路讲解如何由内到外一层一层编程。结合问题分析结果，最内层的过程是实现绘制一个正三角形。学生很快能够编写出绘制正三角形的程序：重复执行 3 次移动某一数值（这里以 100 为例）和旋转 120 度两个指令。这就是解决任务中最内层的过程，命名为 Triangle。但考虑到外层过程中，正三角形的边长是要变化的，不能固定 100，所以 Triangle 中必须还有一个自变量 length（如图 6 所示）。

第二层过程是实现每旋转 60 度绘制一个正三角形直至一周。通过思考，只要每旋转 60 度，执行一次过程 Triangle 即可。将这个绘制正六边形的过程命名为 Hexagon，同样要设置自变量 length（如图 7 所示）。

图 6 过程 Triangle 的形成

图 7 过程 Hexagon 的指令

最外层过程是实现绘制 5 个依次增大边长的六边形。利用同样的解决思路，我们只要重复执行 5 次过程 Hexagon；但是由于每执行一次，变量 length 要增加一定的值。所以 length 需要用新的变量进行控制，将这个顶层的过程命名为 SpiderWeb。主程序只要调用最外层过程 SpiderWeb，就能绘制出精美的蜘蛛网了（具体代码如图 8）。

图 8 过程 SpiderWebnhe 指令和主程序

由内到外的程序实现过程，给学生编程提供了明确的线索，帮助学生理清任务与任务之间、问题与问题之间的逻辑关系，在整个编写过程中，就像提供了一步一步的脚手架。通过用计算机自动绘制出这些神奇的图形，感受计算机程序设计的强大魅力。

3. 定义过程：减少程序的漏洞

简单是程序设计的目标。就像在工厂里的流水生产线往往被划分成若干个工位，每一个工位执行标准化、程序化的动作，这就是单一责任。在程序设计中，单一责任是指某个代码的功能，应该保证只有单一的明确的执行任务。任务越单一，代码越简单，越简单的代码占用时间少，漏洞少，并且易于修改。当其他功能部分发生变化时，也能够尽可能降低对其他组件的影响。在蜘蛛网的案例中，将每个分步都定义为一个过程（如图 9 所示）。每个过程只做单一的任务，比如 Triangle 只负责绘制某一边长的正三角形。

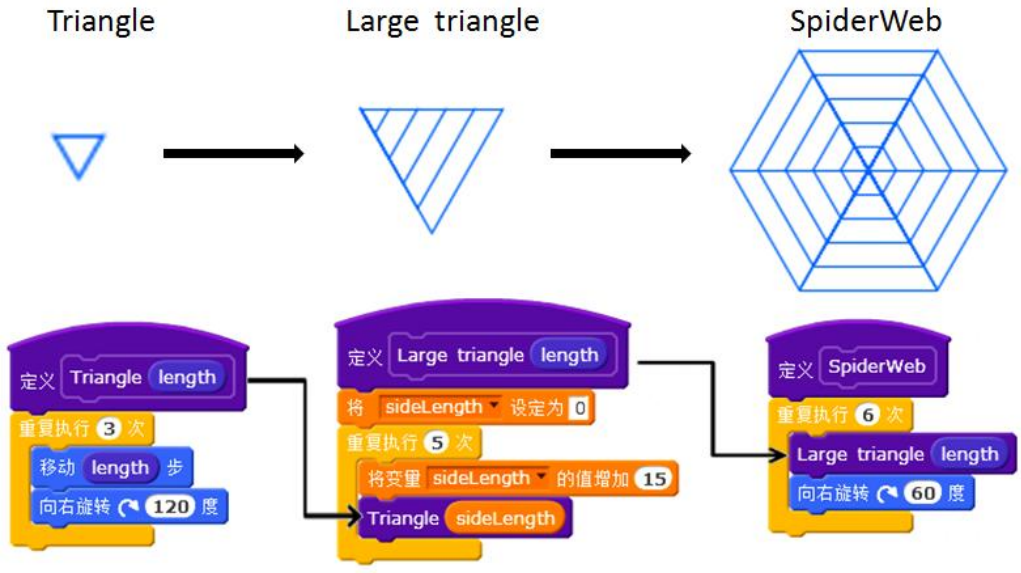


图 9　蜘蛛网案例中的各个过程

定义过程不仅有利于理清编程的思路，也方便程序的调试。在调试程序过程中，可以将总目标分解成一层一层由内向外的过程进行逐一调试。案例中可以先调试是否能画出一个三角形；调试成功后，再调试是否能成功画出六边形，以此类推，直至成功。

4. 应用拓展：从范例学习到个性创意

Scratch 的教学本身就是“为了创作而教”，无论什么主题的作品都不能忽视自由创作。在绘制图形的案例中，学生最主要是学习了问题的分解与具体问题达成的逻辑，重在方法的

剖析，而非图形本身。这为学生走出范例进行个性化创作奠定了良好的基础。

（1）修改变量过程，变化效果

几何图形的变换千姿百态，但核心思想不变。在图形绘制中只需要修改某一个变量（边长、边数、旋转角度）或者修改过程中执行的命令，就可以带来各种各样的效果。当然，修改之前需要先有总体的架构，明确自己的目标图形。比如在学生学会绘制正三角形和正三角形变换得到的蜘蛛网案例后，自行设计绘制正方形，并将绘制正方形的过程作为底层过程函数，创作了更加复杂的图案（如图 10）。

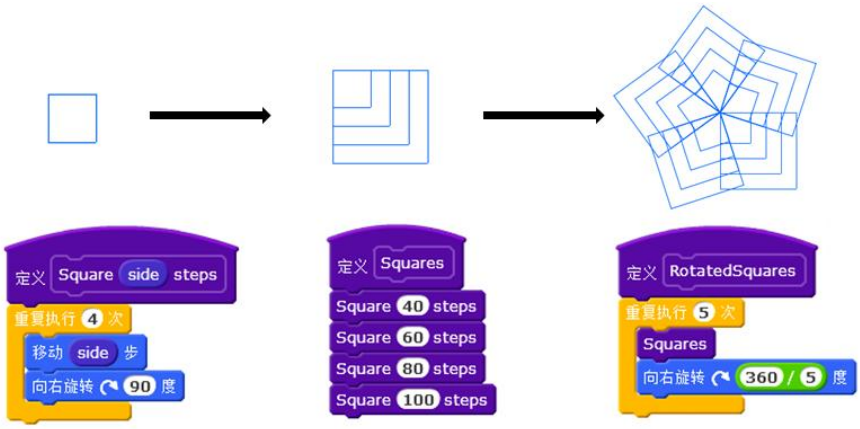


图 10　对蜘蛛网案例的变换设计后的作品《花》

（2）巧用问询模块，实现交互

在绘图的创作过程中，某学生提出是否不要固定绘制的图案，比如上面案例中实现实时绘制 6 个花瓣、7 个花瓣……只要将第三个过程函数的重复执行次数设为变量，并在主程序中加入问询的指令（如图 11）。那么当回答不同的数值时，count 的值就会不同，调用过程函数 RotatedSquares 后的图案也发生了变化（如图 12）。

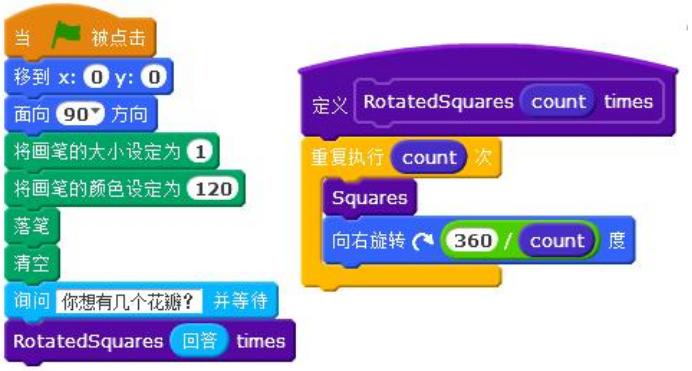
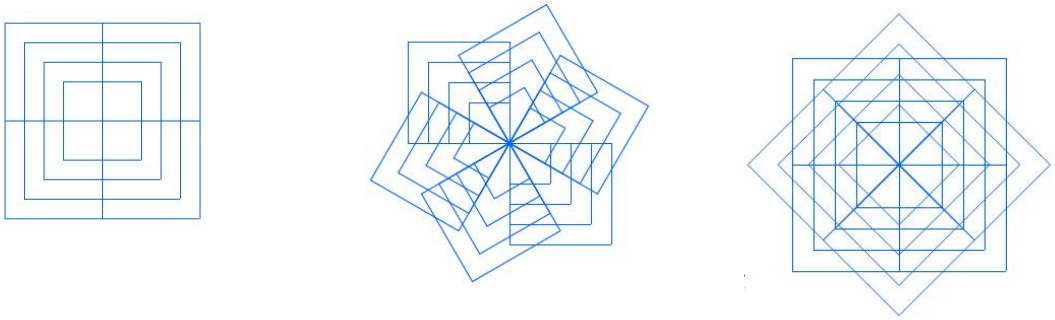


图 11　问询指令的运用



回答=4　　**回答=6**　　**回答=8**

图 12　不同回答形成的不同图案

（3）设置画笔性质，美化图案

画笔模块中有多条与画笔属性相关的指令，学生将这些指令合理地添加到子程序中，可以绘制出更加精美的图案。比如某位同学在以上案例中的子程序 Square 中，每次执行就增加画笔的颜色值，并在 RotatedSquares 子程序中恢复画笔起始颜色，成功绘制出了色彩丰富的花瓣（如图 13）。

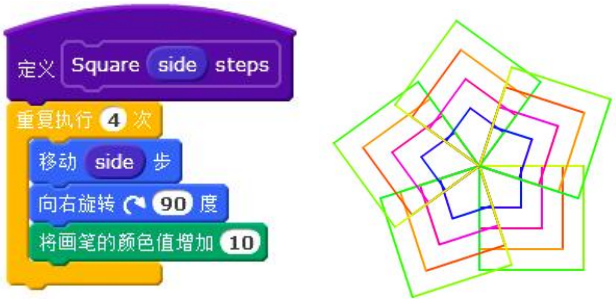


图 13 设置画笔颜色的指令和作图效果

几何图形的创作周期和需要考虑的内容明显小于一个好玩的游戏、动画等互动性作品的创作，也正因为如此，学生发挥创意可以很小，也可以很大。但是，无论是简单的修改变量还是颜色，通过不同逻辑的嵌套后，最后的成品是丰富多彩的。这样简单的操作，却能收获丰富的即时的可视化的作品，更加激发学生对于逻辑的探究，体验到图形绘制的趣味。

三、Scratch 作图对培养计算思维的意义

进行 Scratch 教学的目标是创造，而为了更好地创造，Scratch 教学的最终目标仍旧应该包含对学生思维的培养，如设计规划、逻辑推理、创新能力等。因此，对于学生而言，体验编程的过程、理解编写意图比编出完美的动画更重要。在利用 Scratch2.0 作图的教学过程中，孩子们学会了问题分析，懂得了逻辑推理，更发挥了自己无限的创意。

1.通过任务分析提高了算法思维

在 Scratch 绘图教学活动的设计时，避免了为知识点而进行教学的现象，成功跳出“教师演示、学生模仿”的教学模式，在教学目标的设定上，不再是让学生掌握了哪些指令为目标，而是让学生在完成主题活动的过程中掌握知识点，以培养他们的分析问题和解决问题的能力。比如案例“美丽的花团”（如图 3），要绘制这幅作品，必须由外到内进行逐层分解。

任务分解	内容分析
绘制一团花 	每支花画完之后旋转 60 度（360/6）画下一支花； 重复 6 次以后就可以得到一团美丽的花。
绘制一支花 	移动一段距离后画一个花瓣代表树叶； 再移动一句距离，画一朵花； 回到起始位置，为绘制下一支花做准备。
绘制一朵花 	每个花瓣画完之后旋转 72 度（360/5）画下一个花瓣； 重复 5 次以后就可以得到一朵完整的花。
绘制一个花瓣 	可以用细小的锯齿绘制成曲线，作为花瓣的一边； 旋转 90 度后，重复操作绘制另一边。

在由外到内进行逐层分解出具体问题后，就要针对每个具体问题逐一解决。

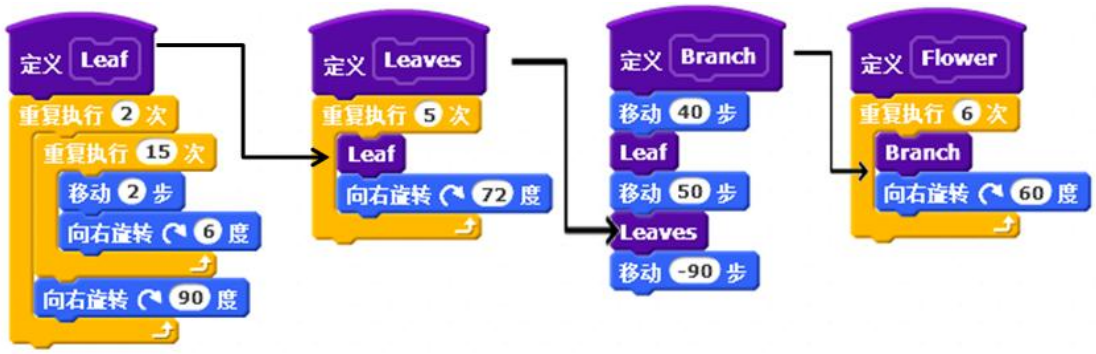


图 14 绘制美丽的花团问题结果过程

2.通过编写程序发展了逻辑思维

几何图形具有简洁直观、易变换的特点，相比与动画故事、游戏更适合用于培养学生的逻辑推理与编程能力。通过图形的创作能很直观的理解并运用参数、过程、分支、循环、递归、模块等逻辑思维。比如在绘制正方形时，通过分析，不难得出绘制正方形时画笔要做的动作应该是“移动 N 步”，然后“向左旋转 90 度”后继续“移动 N 步”，然后继续做相同的动作，直至回到起点。绘制正方形的这段脚本中重复了 4 次“移动 N 步”、“向左旋转 90 度”。学生很快就能得出当编程时候遇到重复的指令时，可以使用“重复执行”“重复执行多次”来让脚本变得简洁。再比如，绘制完正三角形、正方形、正五边形后，学生很快得出在绘制正 N 多边形时，重复执行的次数和旋转的角度，与变量边数 N 有关，理解变量的运用。为了绘制更复杂的图案，还可以将绘制正 N 边形的程序封装成一个过程函数，用来方便的调用。

图 15　绘制正 N 边形的过程函数推理过程

3. 通过拓展应用提高了想象力

编程的学习一个重要目标就是培养学生自主探究和创新的能力，而用 Scratch2.0 绘图的学习，从直线到曲线，从图形组合到图形动画，极大激发了孩子们的学习兴趣，很好地满足了教学的需求几何图形虽然简单，但在 Scratch2.0 中能很好的与计算、推理、益智、游戏等密切结合，因此在教学中给学生留下足够的探究与创新的空间，提升学生的学习能力。从各自制作出的程序作品看得出学生的探究力与创造力是惊人的，由此也感受到，只要提供足够的发挥空间，学生就能展现自己，他们一定会带来惊喜的表现。通过比如调用图 15 中过程函数 Polygon ，经过合理设计，创作出多样的图案，凸显了学生的无限创意。图 16 中展示了部分学生调用过程函数 Polygon 设计的不同效果。

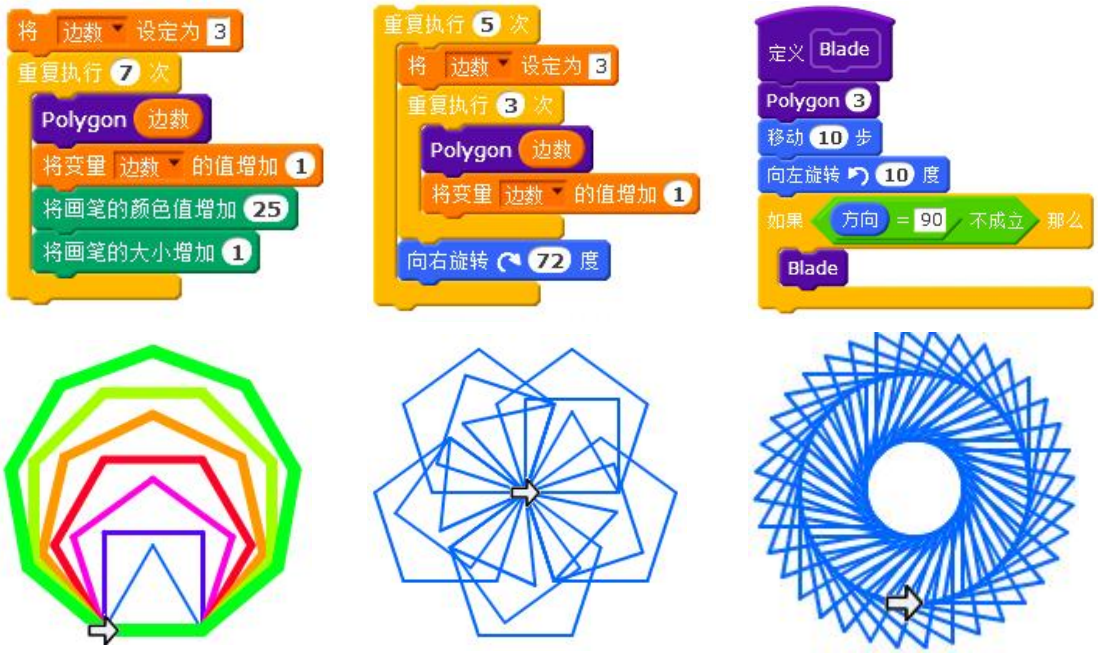


图 16　调用过程函数 Polygon 设计的不同指令及其效果

总之，由于 Scratch 软件的存在优势，比如可视化编程环境、画笔的多种性质、过程函数的方便调用可以结合其他指令达到更多交互功能，使得 Scratch 作图变的既有趣又简单，大地吸引了学生的创作激情。但是几何图形的创作并不能涵盖 Scratch 软件的所有功能指令，不能仅仅依靠 Scratch 作图对 Scratch 实现完整的教学。因此， Scratch 作图旨在培养学生的编程思维，而非 Scratch 指令的系统学习。笔者希望学生通过 Scratch 作图的学习，学会分析问题、体验解决问题的方法，并将学习中渗透的计算思维运用到 Scratch 指令的学习中，对 Scratch 作品的创作提供解决问题的策略。

【参考文献】

[1][美] MajedMarji 著于欣龙李泽译.动手玩转 Scratch2.0 编程[M].北京：电子工业出版社，2015.10

[2]李晓艳，毛爱萍． Scratch 与创意设计［M］. 武汉:华中科技大学出版社，2013.