93,2(1) 可视化 计算机 程序 映射

1—9

计算机科学Vol.20 No.1 1993

程序的可视化：将程序映射至图画的技巧

Gruia-Catalin Roman 和 Kenneth C. Cox

TP31

Roman, GC 陈治东

摘 要

本文将程序的可视化定义为：将程序映射至图画表示。程序的可视化的简单形式常见于软件工程，因此，当代程序可视化的进展有可能影响未来软件工程工具和环境的发展。本文提出了一种新的程序可视化研究的分类法并用以作为媒介，对当今程序可视化的系统、技术、趋势和概念作一个系统的回顾。

任何科学起源于哲学，最终成为技巧。

——美国教育家 W. J. Durant

1. 引 言

计算机是当今以通讯为中心的社会的重要组成部分。网络、台式印刷和电子邮件只是计算机用于通讯的三个例子。随着多媒体通讯的问世，计算机开始支持不断增长的高级形式的人类相互联系。图画的生成、传递、解释、和使用已成为一种日常活动。甚至对于计算和存储密集型的应用，如：动画制作、三维映象和实况视频，费用和性能已达到了可接受的水平。这些发展导致计算中出现了一个新的研究领域：可视化——一个关于在计算环境中使用图形表示的研究领域。

可视化的倡导者从总体上指出了映象在人们通讯中所处的重要作用，说明了人类可视系统的特别频带宽度、人们追踪和检测可视图形的速度、利用多维表达一个词汇的潜力和在图画表示中抽象继承的能力。在与计算机有关的工作中，图形元素的成功应用，如：人的界面和科学的可视化为可视通讯提供了巨大的潜力。

可视化的两个子领域：可视编程和程序的可视化，在软件工程中有特殊的含义。可视编程是关于计算机程序的图形规格说明，而程序的可视化是涉及以正文形式表达的程序的图形化表示、监视和探测。最初可视编程和程序可视化的基本内容是软件工程研究的一部分，而可视化领域的近来发展，使它被重新作为重点研究的对象。可惜，软件工程中的图形表示常被视为解决各种技术交流难题的灵丹妙药。图形常常被用在下列情形：正文用图更为经济；图文合成不很精确和流畅；图形的直观特性有时可以掩盖有力的形式化框架的匮乏。

为了使图形成为有效的通讯媒介，仅仅相信图形的价值是不够的。就软件工程而论，可视化被称为开辟了通讯的新途径。它不是让我们试图用计算机去支持我们已很好地了解的东西，如台式印刷。我们正在创造一个将计算机和可视通讯和谐地结合在一起的世界。然而，可视编程和程序的可视化目前还没有包含在我们寻求的答案中，而它们

会提供有价值的灵感、有意义的新思想，和未来CASE技术的曙光。

2．一种程序可视化的分类法

制定出一种分类法总是困难的。任务是挑选出分类原则，能明确区分所认识的领域，在各种系统的运转中提供有深远意义的理解，并帮助鉴别未来发展的可能前景。到目前为止，各出版的综述报告应用了不同的分类法。Shu[29]关注从精美的印刷到复杂算法的动画制作等程序可视化系统所展示的先进技术的增长程度。Myers[23]使用沿着二个轴的分类法：程序的图解说明部分（代码、数据、算法）和显示方式（静态或动态的）。Chang[8]虽没有提出一个分类法，但用可视化表示说明了程序、数据、系统的结构或系统动态行为的程序可视化的特征。最后，Brown[3]沿着三个方面：内容(关于程序信息的直接或人工表示)、转换(离散或平滑地改变映象)、和持续(当前状态或整个运行历史的表示)，提出了分类规则系统的思想。虽然这些分类法各自有它的理由或优点，但都不能令人满意，因为它们没有以这个领域的系统化模型或理论为基础。从试图搞清一些术语的含义着手，我们寻求一个适当的程序可视化的模型。

我们把可视化视为将程序至图形表示的映射。就此而论，术语程序的含义、可用的图形词汇、被考虑的映射种类、和构造这些映射的手段变成了一个程序可视化系统的定义特征。这种定义还假定可视化的过程是三种参与者（开发源程序的程序员、定义和构造映射的动画制作者、和观察图形表示的观察者）之间相互作用的结果。然而，这只是程式化的角色，其意图是帮助我们组织和提供材料，每种角色所需要的专长可以根据实际情况，在不同个体中进行自然的分工。

如上所示，这个可视化的特殊观点导致程序的可视化系统的分类是以它们支持映射的领域、范围和特性的表征为中心的。以下我们沿着四个轴线说明我们对程序可视化的分类。第一和第四分别涉及映射的领域和范围。相应地，其它的两个轴线直接地同映射有关。图1表示我们的可视化模型，阐明了分类准则和参与者角色。

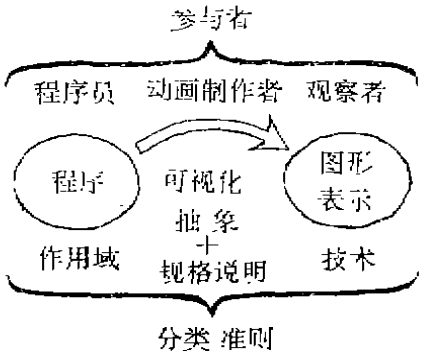


图1　可视化作为将程序至图形表示的映射

(1) 作用域——程序的哪些方面是可视的？一个单独可视化的范围是一个程序。形式上，一个程序可以以它的代码、它的数据和控制状态、它的执行行为主要特征。可视化系统常将它们的作用域限制为程序这些方面的一个子集。Myers和Chang提出了一个类似的准则。

(2) 抽象——何种信息需要通过可视化来传递？这个准则与Shu按先进技术增长程度的分类及Brown的"内容"概念有关。这个问题具有抽象程度，涉及专门系统以一个图画形式提供的概念。例如，加亮的代码提供了控制状态的一种低层次表示。另一方面，一个动画制作以图形零星表示的说明信息来增强一个程序行为的表述。

(3) 规格说明方法——可视化是如何构造的？这个问题被出版的许多研究报告所忽略，它是理解一个特定系统的功能和灵活性的基础。一些系统提供"硬连接"(hardwired)映射，而其它系统允许随意再定义映射；一些系统关注所映射的程序状态，而其它系统关注最后结果；一些系统需要代码的修改，而其它的系统则不需要。

(4) 技术——图形表示通常怎样传递信息？这个准则考虑到必须要有有效的可视通

讯的机制。它们包括可视词汇、传递特定种类信息的专门可视单元的使用、可视信息的组织、甚至包括提供给观察者材料的顺序。

3．作用域

程序可视化的目的是抽取一个程序的某些部分的信息，并用图形加以表示。要准确地考察程序的哪些部分可作为可视化系统的主要定义特征。虽然趋势是朝着使程序的所有部分均能够具备可视能力，特别是它的运行部分，但是，不同的系统可能仍强调一个方面。如果我们回顾一下历史，这是确实的。在程序文本的意义上，代码从一开始就是首先感兴趣的方面，仍有其重要性。数据状态的可视化也是早期感兴趣的，大多数程序的可视化系统仍然关注程序内部数据的表示。控制状态在体现递归和回溯的顺序程序和近来在并行程序设计中受到了特别的重视。最后，关于程序的行为，起初是用于输入/输出监视，随着计算能力和程序可视化方法的进步，已成为最先进的程序动画制作的焦点。

代码。精美印刷程序代表着面向代码的最初形式的程序可视化。它们将语法模式转换成页的版面格式，增加了代码的可读性。随着激光打印机的问世，用多种字体、色调和简单成行的图形来产生吸引人的程序代码清单已成为可能。

产生代码图形表示的程序已在几年前就可使用了。最早的一个系统[14]能产生流程图。最初，这些程序是很有限的，只能产生语句一级程序结构的框图。接下来的趋势是增加抽象能力。现代系统允许用不同的方式来观察单个程序，从一个语句内表达式的树结构到程序模块间的相互联系的框图。这些当然是可视的CASE系统，其中Garden[25]和PegaSys[21]可作为典型的例子。许多这样的系统既是程序的可视化系统，又是可视编程系统，它们允许用户观察现有的代码，又可通过合适安排图形元素来构造新代码。

数据状态。早期的数据可视化系统由于处理器速度比较慢，需要某些间接的方法。早期两个动画制作的规则系统是一帧一帧地记录在胶片上，每一帧表示一个状态的快照结果。这些胶片的可视化采用了一个相当低级的抽象，典型地仅仅是抽取如象数组里的值。提高处理器速度，就相应地提高监视和抽象技术。这已在几个调试程序中进行了开发，用于复杂数据结构的检查。数据库查询和内容的图形表示可以同这种类型的可视化系统密切相关。

最新的程序可视化系统允许生成复杂数据结构表示的平滑动画，例如：ALADDIN，ANIMUS及其有关工作，BALSA，PAVANE，PROVIDE，和TANGO等等。其中每一个系统提到一个或多个我们下述的感兴趣的地方。

控制状态。在一定意义下，一个顺序程序的控制状态是由处理器的程序计数器来表示的，但只有少数的系统使用这么低的抽象级。而更高的抽象级构造，如语句和过程已被提出。象调用序列的历史信息常需要额外地保留。例如，在BALSA中，代码的当前执行部分能显示在窗口里。其它系统使控制信息同程序的总体结构联系起来，如，把代码结构显示成一个模块互联图并加亮当前正在控制的模块。

几个已设计的程序可视化系统可供语言和其控制状态较复杂的模型使用。一个象这样的系统是TPM (Transparent Prolog Machine)，它能可视Prolog程序的目标制导的行为。目标被表示成随着目标被处理而不断生成的一种树结构；一致化、回溯、和象cut之类特殊操作的行为是用图形表达的。面向对象的框图系统采用了和TPM表示Prolog行为的大致相同方法，试图显示面向对象程序的行为(对象的消息传递)。其它系统使用一个面向对象的框架来构造可视化。在这些系统中，对象一般有一种转换对象状态至某些图形信息的方法。Duisberg等开发的ANIMUS系统就是这种系统的一个例子。Par-

Vis系统使用并行Lisp程序，提供了控制状态的低层次可视化系统，能显示子任务随时间的生成和进展情况。

行为。观察一个程序行为的最简单（和最古老的）方法是检查它的输入/输出结果。这种技术能用来对程序进行有意义的抽象处理：可简单地认为它是一个从输入转换到输出的"黑盒子"。这个方法有重要的理论含义，但只是对算法了解甚少，并不能应用于所有感兴趣的程序。更详细地观察程序的行为需要基于它的内部活动，靠直觉理解。

每个程序都可被看作是对它的状态执行一系列的原子转换。正式地称为事件，观察这些转换，我们可以加深对程序是如何完成其功能的理解。许多程序可视化系统试图捕获和提供这类信息。然而，它们所考虑的粒度和事件的范围有所不同。粒度可以从原始的机器指令到整个语句块完成的大型操作不等。感兴趣的事件可以在特定变量的值、过程的入出口、或通讯活动间变化。虽然一些程序可视化系统预先定义了感兴趣的事件，但大多数系统允许动画制作者对有意义的特定可视化事件进行定义。

随着分布和并发处理的出现，更需要识别和观察独立状态在多结点上改变结果的事件，特别是在系统监视方面，许多研究者求助于使用程序可视化技术来显示它们采集的数据。在并发程序方面，提供操作上的思考存在困难，使一些研究人员考虑下述可视化机制：不去观察程序操作的细节，而是着眼于那些并发计算中正常推理所用的抽象特性。

4．抽象

在这一节，我们讨论由图形表示所传递的信息。我们对系统的区分基于可视化系统支持的抽象层次。程序的抽象表示对于控制探测和监视的复杂度，对于在教学环境中方便程序理解都是必不可少的。在使用压缩的抽象表示时还应考虑显示尺寸限制这一因素。我们的分类采用了五个抽象层次(图2)。我们依次从最低到最高来研究程序的抽象表示。各层次间的界线并不明确，实践中，一个可视化系统能以分开和组合形式支持多个抽象层次。

直接表示。最初的图形表示是将程序的一部分直接映射至图画。由于只有有限的抽象机制可以使用，通常原始信息易从图形表示进行重构。代码的格式化清单、指示变量值的量规集(gauges set)、链表和二叉树的二维表示、数组中值的颜色编码是直接表示的一些例子。在CASE和调试系统中遇到的其它形式最初有：流程图和代码结构的类似图解表示、通过当前执行语句显示的控制流监视、和通过表示调用栈（可能作为重叠的窗口，每次调用一个窗口）的过程调用追踪。

结构化表示。更抽象表示可以是隐藏或封装同程序及其运行有关系的一些细节和使用其余信息的直接表示。二维或三维框图和图形一般用于描述程序的结构（如：结构图）、网络互连和数据存取能力；此时一个由许多子部件组成的复杂对象作为一个简单对象对待，其内部结构被隐藏起来。还可以引证许多其它的例子。直方图可以按类型对消息出现的相对频度进行编码，而隐藏其它细节。在操作系统中，按比例的彩色块可以标出内存的分配和使用，而不用表示内存的状态。所有这些情形虽然细节不清楚，但是显示给观察者的信息却是程序中提供的。对观察者隐藏不相关部分的这种表示以更加经济的方式简单地传递了信息。

合成表示。这种表示方法同结构化表示方法有本质的区别，它所研究的信息不是程序中直接表示的，而是从程序数据派生而来的。当程序员所选择的数据表示同动画制作者的需要相冲突时配画常常发生转变。动画制作者可能更强调没有被明确表示的算法的其它部分，尽管程序在逻辑上被保存着。这时，动画制作者必须构造和保持一个更便于其特殊需要的表示。举一个例子，一个动画

制作者可能强调有待于完成的工作，而程序员可能更愿保留已经完成的工作——反之如此。

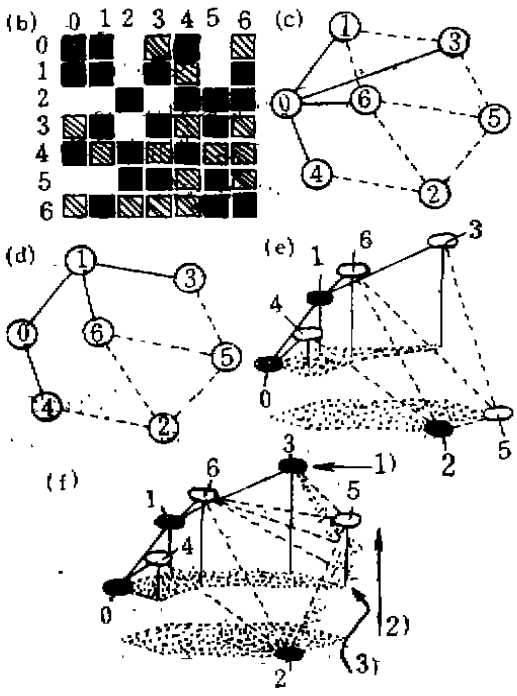
尽管许多系统提供了合成表示的形式（只用有限的形式），很少有动画制作者利用了全部的可能性。这种情况可能部分地归因于没有几个系统有能力随意地构造将程序映射至图形的事实。领域的相对不熟悉也起部分作用——甚至较老的BALSA系统产生的动画制作，也只是有限地使用了合成表示。在BALSA系统，动画制作者可以通过定义与图形过程共享的数据结构来构造合成表示。这些过程（也是由动画制作者写的）既可使用从程序、也可从共享数据结构内容里抽取的数据进行操作。

解析表示。这些是试图获取高度抽象程序特性的表示。这个思想不强调程序运行的操作技巧，而关注程序解析思想中的重要问题，如正确性。举一个排序算法中的例子，动画制作者可能不用通常已排好序的元素的直接表示，而是可能试图捕获一个不变式，如："下标小于变量k值的所有元素被正确地排序"。考虑其特性方面的一个算法常会导致新的和有意义的动画制作，而使用传统的方法学，就可能发展不了。

这个方法来源于这样一种见解，当转换至可视形式时，在程序的形式化推理中重要的特性也帮助观察者理解程序的行为。到目前为止，这个领域的大多数工作是以PAVANE系统为中心开展的，它试图为并发计算的可视化提供一个框架——关于并发计算中程序正确性的形式推理远比在顺序计算领域更为重要，在顺序计算中，算法行为的操作表达方面常常是充分的。

注释性表示。高级的可视化超出了程序状态的简单表示范围，它使用多种可视化的技术来说明程序的行为。这些可视事件常常在由动画制作描述的计算中没有对应物。它们是为了改善表示的外观质量、厚望于传达一个特殊计算事件的含义、和为了集中观察者的注意力而增加的。本质上说，动画制作者擅自在程序的表示中增加事件是为了更好地"告诉故事"。例如排序算法，两个矩形可以平稳地交换位置，指出了两个数组元素的交换；而矩形在移动时中间映象不表示任何实际的计算状态。

```
(a) AllPairs (integer N, real array A[0..N-
      1, 0..N-1])
    begin
      integer k, i, j
      for k:=0 to N-1 begin
          forall i:=0 to N-1, j:=0 to N-1
          parbegin
              A[i, j]:=min(A[i, j], A[i,k]
                    +A[k, j])
          parend
      end
    end
```



1)处理中的节点闪烁并标上颜色；2)当找到路径时移动节点到新的高度(最后画上路径)；3)当向上移动时节点的"增长"线。

图2 可视化中各种可能的抽象层次

图2(a)。一个Floyd-Warshall算法的并行版本，求出一个图的所有成对节点之间的最短距离。图是以数组A的形式给出的，如

果边连着两个点，A[i, j]是节点i到节点j的距离，否则A[i, j]无穷大。在算法的每一步，下标k一经扫描，数组的所有项被同时修改。

图2(b)。几步以后(具体地，当k等于0、1和2时，执行代码)，程序数据状态的一个直接表示。数组A的内容用矩形表示，其颜色（灰度）由数组值编码，黑色矩形块代表值为0，如果数组值为无穷大，则略去矩形块。

图2(c)。上述的同样数据的一个结构化表示。距节点0的距离已知且用粗黑线表示，这是在最初用虚线的图形表示上叠加的。同样只当距离为已知时才被表示。这是一个信息隐藏的说明，一些细节已对用户隐藏；这个图可以视为通过图2(b)第一行的一个“片断”。

图2(d)。一个合成表示。画出了从节点0到每个其它节点的当前最短距离。这个信息没有在程序中表示出来，但能通过可视化系统提取。象在图2(c)中一样，只有在距离和路径已知时才表示出来，但不是实际的距离。此外还要注意，任何节点可以用来代替节点0。可以想象一个系统，观察者可以选择图上的任何节点，并可立即看到相应的路径。

图2(e)。一个解析表示，试图说明一个不变式“如果已知从节点i到j节点j的距离，所有在路径上的内部节点都被扫描”。这是由着色已扫描的节点来表示的，让观察者容易看出该特性。这个表示也使用了Z座标来对距离进行编码，将节点0放在零高度上，已知距离的节点放在合适的Z座标位置上，未知距离（无穷大）的节点放在一个小的负Z座标位置上——一种多少是随意的选择。

图2(f)。这里试图传递一个解释表示的本质，它使用额外的可视事件来支配观察者的注意力。由于印刷媒介的限制妨碍了一个完整的说明，但是我们希望这个图能说明一些概念。在算法中，节点3刚好被扫描，并且已知从节点0到节点5的距离/路径。观察者的注意力首先被闪烁节点3所支配。通过节点5的动画制作再定位，到节点5的路径已经知道；这比节点5用一个分离的步骤，从一个位置到另一个位置——“跳跃”，在视觉上要清楚得多。注意，当节点5移动时，该映象不表示任何实际的计算状态。

5.规格说明方法

凡一个可视化系统将语法单元、程序状态、或事件映射至一个图象时，其关键是容易使动画制作者根据现有任务的需要来构造各类可视化。在定义和再定义映射时，包含测试和调试的探测活动需要高度的灵活性，而在计算过程中运行监视需要最少的干涉。在这一节，我们讨论不同的程序可视化系统实现特定可视化任务的方式。

预定义。特定应用的可视化常常使用一个固定的或有许多限制的映射。动画制作者只有极少或者没有控制权来确定可视些什么及用什么方式显示信息。大多数CASE系统选择这个方法是由于它们打算支持和采用一个专门的方法学和图形表示。虽然这些系统的表示能力常常有限，但是它们有速度的优势，这些约束能使他们使用优化性能的专门可视化算法。

注释。这个方法是BALSA首创的，并且在动画制作领域的算法中得到了普遍的认可。动画制作者通过调用允许他们构造和修改图象的过程来扩展程序。安排这些调用（注释）对应出现在代码事件中，被认为对算法的行为是有意义的。经过被调用动画制作过程的参数，传递了有关程序状态的信息。用这个方式，将计算事件映射至可视事件的任意序列，即：图形表示中的变化。TANGO使用了一个类似的方法，但强调了平滑动画的生成。TANGO可视系统被看做是从程序事件到动画动作的一个映射。一个感兴趣事件的出现触发了在图象中一个或更多图形对象的某些平滑动画制作。

注释方法有几个优点，主要的是动画制作者能在任何合适的层次下定义事件。例

如：一个排序算法可以通过检测每一次比较和交换来进行动画制作，然而，如果同样的排序算法在另一个算法中作为一个子程序使用，动画制作者只可能决定显示其最终结果。书写特定应用的动画过程来处理每个事件的能力既有优点，也有缺点，它提供了通用性，但需要额外的工作；通过程序库的使用可减轻某些工作。这个方法最显著的缺点是需要访问并修改程序的代码。

关连。一个影响较少的替换方法是让用户在程序状态和图形对象或图符的属性之间定义一个联系。每个所选状态（有代表性的是数据值）的改变都会引起图象的相应改变。这是PROVIDE使用的方法，它允许动画制作者象使用量规或滑杆一样，直接将变量值映射为图象的属性。PROVIDE也允许观察者通过操纵图符与计算进行对话，从而改变相应的程序值。PVS使用了一个类似的方法，但它试图监视现实世界的过程（如：一个发电厂），而不是程序。Garden的可视化编程系统允许程序员定义可视的数据类型，它们的属性是程序状态的一部分，是基本关连思想的一个有意义变种。

关连方法——随状态的改变而自动地更新图象——的主要优点也是它的一个主要缺点，因为在一个合适的层次中，去获取状态的改变是困难的。当几个原始状态改变应考虑成单一的逻辑改变时（例如：当两个变量用三个赋值语句交换时）一般会有困难；图象不能加以修改以反映中间结果。使用关连的大多数可视化系统提供了指示（一般是通过注释）何时图象应该修改的机制。另一个缺点是用简单变量至图形属性的映射，给出的是一个相当低的抽象层次。

说明。动画制作者在程序状态至最终图象之间确定了一个映射，状态的改变能够立即反映在映象方面，这种说明性方法类似于关连方法。然而，说明性方法允许任意复杂的映射，而不是关连方法中简单的一对一的映射。这样，状态的任意属性能够被表达或可视地获取。PAVANE是这类系统的一个例子。PAVANE将程序状态塑造成一个“元组”的集合，并使用一个类规则的表示方法来定义状态和图象之间的映射；映射可能是由几个子映射组成的。ALADDIN也使用一种说明性方法来定义程序变量和图象间的联系，但是，它组合了注释的方法来指出在程序中哪些方面，图象应该修改；注释是用图形给定的。ANIMUS及其先驱系统Thing-Lab以两种方式使用说明性方法。它们是面向对象的系统，每个对象可以有一个图形表示，能响应对象的变化，自动地做修改。更有意义的是，动画制作者能以说明方式给定对象间关系的约束，并且系统将确保维护这些约束（例如：移动一个对象的表示）。

说明性方法的最大优点是它的抽象能力，但是，所增加的功能也需要更多地处理程序到最终图象的映射。另外，观察者（通过图形对象的修改）所进行的程序操作比在关连情况下要更加困难，因为变量与对象属性的一种一对一的关系是不存在的。PAVANE开发者最近开始使用“逆向规则”——将观察者同图象的相互作用映射到程序状态的改变。

操纵。系统通过例子使用“操纵”（也称为通过演示进行动画制作）来定义可视化。Duisberg开发的手势（gestureal）系统是使用这个方法的少数几个程序可视化系统的一个例子。这个系统试图获取被动画制作者用来直接操纵一个图象的手势，并将这些手势同特定的程序事件联系起来。Duisberg用一个排序算法来说明他的系统，其中数组元素交换的动画是定义一个“交换手势”，交换两个矩形的位置。然后，这个手势通过选择合适的程序编码部分与排序算法中交换事件联系起来。虽然在原理上是振奋人心的，但是被操纵的规格说明在定义手势和程序事件的确切关系上遇到了困难。它可能在特定情况下实现，但是，还必须定义一个通用的方法。一个混合方法（动作用手势表示，而事

件和一些动作的属性之间的连接用更传统的方式表示）的研究毫无疑问是富有成果的。Stasko最近描述了供TANGO使用的一个手势获取系统。

6．技术

在这一节我们探讨用于有关程序信息可视表示的技术，重点放在方法上，而不管它是基础的，试验过的或是加强可视通讯的。表示方法稍侧重于算法动画制作（即：表示而不是监视和探索），因为在这个领域，正在探索最先进的技术。最近，Brown完成了一份有关程序可视化技术的研究报告，对BALSA和Zens上的工作进行了说明。

程序的可视化是一个年轻的领域，没有现成定义好的技术词汇。另外，新技术（如声音合成）在不断增加，由于这个原因，尽管我们试图给出这个领域的一个适当的完整概括，但我们不自称是全面的。

样本执行选择。一个可视化表示的复杂度一般同传递信息的数量成正比。因此，可行的办法是从具有问题比较少的实例的表示开始，逐渐引入具有问题较大的实例，以利于观察者从一开始就能够理解显示在屏幕上的可视图形的含义。如果用于教学，就可能要考虑教学的情况，如提供已排序或倒排序的数组给一个排序者，或偏重数据的选择，例如保证一个数组的所有元素都是不同的。当展示非确定算法的行为时，影响用程序进行选择的能力是有用的，象并列地给出每个可能选择结果的能力一样。一个类似的技术是给出所选择计算方面的历史，要么是在分离的窗口中，要么混合成基本图象。这使当前的活动在先前执行步骤的更广的上下文关系中得到真正的理解。已表明，所有这些技术在建立观察者的自信心和直觉力方面提供了有价值的帮助；采用分类和路径算法的BALSA动画制作的综合库，提供了这些技术十分有效的一个例子。

屏幕设计。这个问题的范畴涉及到最终图象的全局结构，包括屏幕上的位置和图形模型的内部特性。最基本的决策是模型的维数。早期的系统被限制在二维模型，但现在已应用三维模型。有些专门应用的科学可视化系统使用高维空间，被投影到一个三维空间上显示。可以想象，增加维数能有效地加强数据的表示。在录像带的扩散计算(Diffusing Computation)中可看到这样的例子，它给出一个三维空间最终检测的算法。

另一个要考虑的因素是视点（viewpoint）的位置（在模型空间中，即观察者眼睛的位置）。有少数系统是固定位置的，但大多数系统允许观察者移动视点。在二维系统中，视点一般是在对象所处的平面上，从某距离的"上方"直接"向下"处理的，视点的运动被限于简单的图象滚动和视点接近或远离图象平面的移动。在三维空间中情况要复杂得多，因为观察的方向也须修改。允许观察者"飞"过一个三维布局的控制已用于科学可视化系统。PAVANE系统使用一个极座标系统来安置视点，并允许视点简单的变换，如：可自动完成一个整体模型的慢速旋转。

有些系统允许同时显示单一算法的多个视图和几个不同算法的视图，这一般是在不同的窗口上并使用分离的视点控制。当用不同的表示法进行试验，以决定哪一个为最有效时，第一种技术是特别有用的，而且在最终表示中也是有用的，因为观察者能关注所找出的最有效的表示。第二种技术允许对几个算法的性能进行对照；BALSA系统在进行排序和装箱算法的表示时，使用这种技术获得很好效果。通过使用这两种技术，比较两个图象的能力使得人们对程序的行为或所代表的程序有了很好的理解。

信息编码。Tufte完成了一份优秀的信息编码的综合研究报告。广义地说，在动画制作者的处理方法上，可视表达的手段分为三个范畴：可视对象及其属性、对象间的可视联系、和可视事件。

可视对象一般用于相对地编码程序的具体部分，如：各个变量的值或能够操作的个

体。最简单的可视对象是抽象的几何实体，如：点、线、矩形、圆和球。实际上，所有程序的可视化系统提供了上述的基本对象。它们的使用是高效率的并受到了动画制作者的喜爱，因为它们能够以任意方式进行组合、有各种特性可用于对值进行编码，并且无需预定义语义内容。简单的几何实体可以组合成复杂的对象或有多个活动部分的对象。复杂的组合能用来模拟物理过程或机器。这个方法的一个变体——widget被广泛使用。一个widget基本上是一个预先定义的复杂对象，可以作为具有一定特性的简单对象来对待；一个游标尺是这种widget的普通例子，还有更复杂的例子，如饼图和二维数据的绘图。可视对象的第三种类型是图符，它被定义为一个有固定语义解释的图形实体，(一般地)对它的应用有固定的语法规则。对于可视语言的设计者来说，图符是很受欢迎的，但在动画制作中应用不广泛。

大多数程序可视化系统的一个显著缺点是缺乏定义新的可视对象的设施。ANIMUS提供了一个将新对象定义为一个或更多对象的集合的系统。新对象的特性能被组分对象访问，附加的约束能放在成员对象之间的联系中。如果融合在其它系统里，无疑证明这个方法是有用的。

在一个典型的可视化系统中，有些被传递的信息是通过对象的属性（颜色、尺寸，等）来表示的，但是大量的信息是通过对象间的可视化联系传递的。这个工作多数仍是一种技巧，但能鉴别出有些通用的规律。程序的结构特性能通过模拟几何结构来获取，例如：两维数组可以通过一个瓦片矩形来表示，或在一个圆环里安排循环方式的执行过程。更多的抽象特性可通过增强特殊的几何线条和着色规则来进行可视化。例如：在装箱算法中，当前封装的元件不适合于已确定的箱子，这一事实是一个程序不变式，通过用矩形表示的箱子和元件可容易进行观察，矩形的长度同箱子的容量或元件尺寸成比例，并且排齐这些矩形，其长度就能可视地比较了。尽管布局很重要，但是，极少系统提供显式的方法来加强这些限制，而代之以隐式信息。例如：一个动画制作通过使用下述转换功能：把数组元素的下标转换为所表示数组对象的X-座标，可隐式地表示一个数组元素的布局。ANIMUS是为数不多的这种系统，它允许通过使用元素间几何联系的约束来给出布局的显式规格说明。

可视事件在图形表示方面包含离散的和连续的改动。离散转换一般用于改动很容易得到的场合，此时试图传递一个程序全局行为的直觉；BALSA的QuickSort可视化系统可以作为这样的例子，其中每个离散改动显示每个分离步骤后的结果。连续转换适用于说明小的步骤，当必须向观察者解释状态改变的特性时也可使用。平滑的改变也能容易区分变化的属性和不受影响的属性，后者在离散改变时可能是一种不明显的细微变化。TANGO系统强调了这些平滑动画制作的重要性。

表示的增强。特殊效果常被用来改善表示的可视质量。例如：光滑反射面的使用有很强的美感。然而其它的增强，既令人愉快又能传递语义信息。按照预先定义的轨迹移动时，一个对象从一点慢慢演变为最终尺寸不只是为了可视，而且还能给观察者以时间来处理信息，也能用来定义原始点出现的上下文和被引入的新对象之间的一种联系。通过颜色的改变或透明色的叠加，加亮专门对象，可以引起观察者的注意，这些对象是当前状态转换的主角。一个类似的技术是使用颜色的改动来为对象的"年龄"编码，可使我们既关注了最新的变化，又加强了对算法行为中历史趋势的理解。

参考文献32篇(略)

〔陈海东译自"第十四届软件工程国际会议论文集"，1992 ACM 0-89791-504-6，董士海校〕