面向对象的图形设计技术与范例

丁东　溪人
志洪　约克　　编译
周瑜

北京希望电脑公司

面向对象的图形设计技术与范例

丁东　溪人
志洪　约克　编译
周瑜

北京希望电脑公司

1992

前　言

现在，面向对象的编程技术正从研究实验室中出现并走进工业应用领域。越来越多的产品借助于面向对象的编程技术和工具来实现，这通常是在混合开发系统中作为传统语言的一种扩充。一些比较有名的例子是OSF-Motif,News,在NeXT计算机上的Objective-C，由C扩展的C++和LISP的一种面向对象的扩充语言CLOS。所有这些语言的开发都结合了交互式图形学。

结合一个图形系统核心的有效的面向对象的系统——这意味着计算机图形学的领域现在已变成几乎是面向对象世界的一个侧面了吗？我们不这样认为。尽管存在着令人感兴趣的个别开发，但还没有响当当的面向对象的图形系统可资利用。如果需要开发一个嵌在一个面向窗口的系统内的复杂的图形应用程序，那么仍然需要利用基本的工具。在一个窗口内将被显示和交互修改的内容必须用一些图形原语在低的层次上具体实现，或者必须用一种标准化的图形核心系统（如GKS和PHIGS）写出。即，通过核心以一种非面向对象的风格具体实现。

随着名词GKS和PHIGS，我们进入国际图形标准的世界。GKS和PHIGS构成系统，不仅仅是图形原语的堆积。这些系统的定义结合了计算机图形学领域内长期遗留的问题，并把它们放到系统级的地位。

Peter Wisskirchen在很早以前就具有从事GKS设计的经验，随后他研究和总结了PHIGS，并分析了在面向对象系统中结合图形学的可能性。他得到一个惊奇的和令人困绕的结论：现在计算机图形学领域内工作着两种完全隔离的团体，他们彼此并不了解，就象在被一条大河分开的两岸从事相似的耕作一样。

本书的目标是架设一座桥梁，在传统的方法（如GKS和PHIGS）和面向对象系统之间建立联系。这不是一条高速公路，而是作为连接两岸，帮助探索另一面情况的小径提出的，为此书中引入了GEO++系统。该系统试图把面向对象的基本原理和工具同传统的功能结构结合起来。

本书适于从事传统的特别是从事较大的图形编程系统的研究人员和专职人员。这意味着可以帮助他们在他们的工作中结合面向对象的概念。或许应该承认，面向对象的思考方法在计算机图形学中导致了针对界面设计的正确的编程模型。

然而本书同样适合面向对象图形系统的研究人员和开发人员。它究竟能为这些人提供些什么呢？或许它能激发人们进行某种尝试，把从传统的图形核心系统的功能规范化为一种标准。也可能促进新的图形标准方面的工作。

在标准化方面，一种“预制的”（prefabricated）核心的思想起着支配性的作用，核心的功能标准化，内部的实现是隐式的，直接放进应用程序员的手里。这种思想的基本优点——功能的系统地结合，软件可移植性，设备无关以及被特定的硬件或软件开发优化性能的机会——的另一面却有一个极大的弱点，这在较低的灵活性方面常常显示出来。而灵活性恰恰是面向对象系统最有潜力提供帮助的所在：继承是一种提供预定义功能修改和扩充的完美概念。所以非常有理由来说下一代图形标准将被定义成一种面向对象的系统。

为了把一座小桥扩展成一条宽阔的高速公路，我们还要加进去高性能的图形学，如真实感绘制和光照模型及面向对象的计算机动画。我们希望面向对象前景方面的潜力能够通过比较以及编程实例来反映出来。理解编程实例只需要有基本的计算机图形学方面的知识。一种面向对象编程语言方面的知识可能会有帮助但不是必需的，象smalltalk-80的介绍应该能满足人们去理解面向对象的基本原理和实例。

本书正文共分十四章。其特点是显而易见的。第一章的标题就能使我们更快地理解本书的含义；第二章则介绍了面向对象的概念以及Smalltalk-80编程语言，并在第四章讨论了smalltall-80图形核心。在第三章研究了面向对象的界面结构之后，第五章介绍了GKS以及面向对象的系统设计。第六和七章谈到了部件的层次结构，并涉及到PHIGS的有关问题。我们引入的系统GEO++则在第八章和第九章里讨论，在第十二章还谈到Smalltalk-80中的GEO++问题。其它一些重要的概念如继承、原型等集中在第十章和第十一章讨论，在那里GEO++及部件等内容都有一定程度的涉及。

最后二章，即第十三和十四章谈到了一些补充的内容以及标准化问题。并且在正文之后给出了有关名词的中英对照表以及参考文献，以便今后学习和进一步研究。

我们觉得面向对象的编程正从实验室走向应用领域。Peter Wisskirchen的有关著作既有理论分析又有编程说明，对于我们理解面向对象系统，图形学以及它们的完美结合都有相当大的指导作用。

目　录

第一章 绪 言

近几年来，在计算机科学领域，“面向对象”或“面向目标”（Object-oriented）这一术语已经变成最为时髦的口号了。而这一术语可用于两个不同的方面——用于描述用户界面与及用于面向对象的程序设计语言和环境。

所谓面向对象的用户界面，是指象APPLE公司的Macintosh机器那样的，用图符和鼠标器直接操纵对象或目标(Object)的用户界面。而对象这一概念，在这里意味着以一种直观的形象来表示整个屏幕信息的各个组成部分。用户把这些组成部分视为一些单元，从而对之进行操作。例如：用户可以通过鼠标的按键，移动、擦除、或者改变其所表示的内容。这些对象的实例均是可视图形文件的折迭（folder），通过在鼠标器的操作，它们能够被打开；它们可以是弹出式菜单，菜单的内容项可被选中激活；它们也可以是窗口，可以被打开、扩大、缩小、移位或关闭。这种面向对象的界面设计并不要求采用面向对象的程序设计语言。

上述的面向对象的用户界面，从交互式计算机图形学的观点看，可以说没有实质上的新的东西。作为交互式图形学的一个研究领域，面向对象的界面的设计一直受到注重。然而具有新意的是，上述面向对象的界面突破了一度是计算机图形学中关注的那种界面形式，面向对象的界面不再局限在过去的领域中，如象计算机辅助设计（CAD）或制图学中。新的界面代替了广泛使用的面向命令的用户界面，而作为一种新的媒介选择。近期流行的许多系统表明，面向对象的用户界面技术取得了成功，它将不断得到推广。

而所谓面向对象的程序设计，我们理解为面向对象的程序设计语言，亦即是采用面向对象的程序设计环境来编程。面向对象的程序设计语言是通用的设计语言，适合于各种各样的任务。然而我们相信，面向对象的这种范例为交互式的计算机图形学提供了特殊的功用，面向对象的程序的思想精髓正成为设计交互式计算机图形学编程工具的指南。

1.1 面向对象的语言和工具

面向对象的技术，作为直观的界面方式被采用经已有几年了，而它的概念却不甚明确。

按照Cardelli和Wegner等人的描述，面向对象的技术是以下几方面的组合：

· 数据抽象（即界面和隐含的局部状态）加上对象类型（或类——classes），再加上类型的继承性（即那些从超类继承下来的属性）。

· 处理过程是通过向对象发送和返回消息而实现的。

语言无须受限于面向对象的技术所具有的特性。面向对象的语言继承了Simula语言的特性，而由Smalltalk-80作为范例。在Hewit的规范中，他极为强调并发和消息的传递。NeWs的窗口系统，采用了PostScript语言来描述面向对象的用户界面，其方式极为自然。Meyer讨论了面向对象程序设计的一般特征。

支持面向对象的程序设计的工具，几乎随处可见。其范围从杂交的到完全的面向对象的语言，之所谓杂交是指前者是在现存的普通程序设计语言中加进了面向对象的概念，而后者

则是全新的，具有完善的编程环境。

·在人工智能的领域内，面向对象方面增强了的LISP语言已经出现了相当一段时间了。在这里，面向对象语言的形式结合到了其它语言的规范之中，被嵌入在为基于知识的系统而的语言外壳中。多年来的这种语言扩展的经验，使得人们要求对LISP语言的这种扩展制定标准。

·Smalltalk-80语言及其精制的编程环境几乎在各种机器上都得到普及，并适用于各种操作系统。Stalltalk-80语言是绝对的面向对象的语言，因而它被广泛的认为是面向对象语言的典范。由于多方面的原因，我们选择了Smalltalk-80与及Xerox Park的程序开发环境作为本书的参考系统，用于描述各种实例。Xerox Park的书籍中，有关Smalltalk-80的文档资料广为流行，非常容易得到。由于基本性能方面的问题，使用未能成为商用软件开发的常用语言。

·“对象Pascal”和最新版本的Turbo Passcal都是面向对象化的Pascal语言，被用于Apple的Macintosh机器。

·Objective C和C++是C语言面向对象方面的扩展。为使面向对象的语言能够与C语言的高速度相结合，这种C语言的扩展版本得到了迅速的发展。

上述例子表明，面向对象语言的各种实现结合了各种规范和概念。而本书则是注重了图形化的编程，并非一本面向对象的系统方面的综述。这样的综述应该讨论现存的多种面向对象的语言实现，注重其异同，而我们则正好与此相反。我们忽略这种语言实现的差异，而着重讨论它们之间的共同特性与及它们与计算机图形学之间的关联和提供的支持。计算机图形学在本书中将占有突出的地位。应该强调的是，本书中Smalltalk-80描述的程序能够很容易地转换到其它面向对象的语言。

1.1.1 对图形的支持

现在流行的许多面向对象的系统都提供图形编程的工具。本书中，我们将在一定的程度讨论Smalltalk-80的图形核心。

一般的，对于面向对象的图形系统这一概念，我们可以从现有的图形编程工具中提取有益的思想用于面向对象的系统。我们的讨论将要涉及图形系统中的一些最为重要的因素，然而所涉及的，却不象通常在图形学中的那样广泛及系统化。

我们将会反复引用传统的过程化定义的交互式图形系统，亦即在相当长的时间中所称的图形核心系统。GKS和PHIGS即是这种系统的型范。

应该认为，现在的面向对象的系统中的图形工具，并非旨在与PHIGS之类的完备的图形系统进行竞争，实际上，这些工具更应该与现代工作站上的工具进行比较。它们都具有预先定义好的基本的图形对象库，用以用户与系统的对话。这些库包含了窗口系统、图形菜单、对话箱等等。

面向对象的应用工作主要是用户界面方面的开发。这种思想的最早实现和流行，是Smalltalk-80的MVC模型和APPLE公司的MAC机。面向对象的用户界面管理系统，尤其是其中界面的创建，使得我们更向前了一步，其目的是通过提供一个图形用户界面，使得用户能够最大程度地对其设计的界面施加影响。面向对象的方法，也可以用于计算机动画领域，其可行性有充分的保证。

1.1.1.1 标准化的前景

与图形系统的标准化比较而言，面向对象的系统和程序设计方面的标准化，似乎并未引起多大关注。因为对于图形数据的处理这一重要工作实际很少有人问津，多数人的兴趣集中在面向对象的程序设计方面。我们希望本书中所讨论的例子和建议能够对面向对象系统的标准化有所促进。

1.1.1.2 计算机图形学是面向对象的

如上所述，从计算机图形学的观点来看，现在流行的面向对象的用户界面并非什么新鲜事。我们直观地考虑为对象的东西，实质上是需要处理的实体，依此观点，交互式图形学一直都是面向对象的。也就是说，这一切都是有关对象的表示和对其的操作的。这包括：表示构造；编辑；几何变换和图形输入设备对其的识别。图1.1所示是MACDraw交互式图形编辑器的界面。通过对这个编辑器的使用，我们容易直观地理解面向对象这一概念。在这里，我们不难得到下列结论：

- 对象是具有自己的数据的东西
- 对修可以有不同的显视属性
- 对象可以包含别的对象作为该对象的构成部分
- 对象可看作若干个单一的实体，它们能够被选择和操作（删除、修改、移位、旋转）
- 有的对象可以认为是我们普通描述文本的代表。

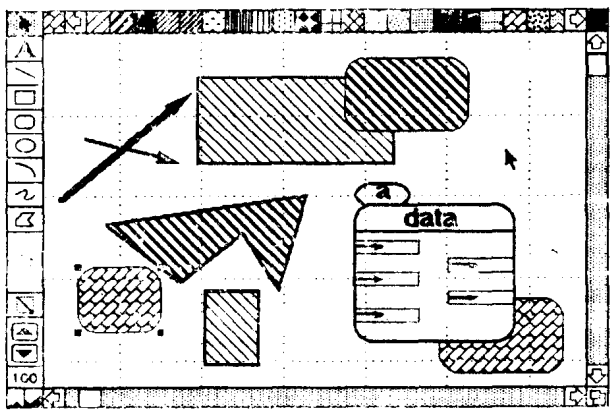

图1.1 MacDraw的面向对象用户界面。计算机图形学的典型应用是图形编辑器。编辑器创建的赖以编辑的实体被直观的解释为对象

1.1.1.3 问题的提出

如果面向对象这个概念是如此的自然，那么还有什么问题有待于解决呢？迄今，计算机图形学得到了相当程度的发展，用程序设计语言编程，然而如果仅仅依靠计算机图形学的基础，是否能够提供处理图形对象足够满意的手段呢？回答是否定的。

1.1.1.4 传统的核心系统

图形标准的生命力依赖于它广泛的功能。图形数据结构或对象结构的构造和进行编辑的规则，被定义为一个模型。建立在图形标准基础上的一个完整的模型，并非只是一些图形功能的总和。按这种观念，图形原语并不是任意使用的，它必须被安放在某个预先描述的模型

之中，例如：将PHIGS的原语插在某个现存的结构之中。图形数据结构能够被联接起来构成结构网络，并支持多层次的网络。这种严格的模型概念，对图形程序设计构成一种约束。好在图形核心所提供的服务足够的丰富：硬件的无关性、精密定义的属性和变换、查询功能、输入输出功能的自动耦合、文档中图形信息的管理。在这个定义上，传统的图形核心不只是一个工具箱，而是一个真正的系统。在标准化的过程中，预先制定核心这种思想起了决定性的作用。这意味着核心的功能是标准的，而其内部实现却是隐藏的，它可随时为应用程序所调用。我们把这种思想叫做隐藏内核原理。

这种思想的主要优点有：

·系统功能相互配合

·使用这种内核的软件具有很好的可移植性

·与设备的无关性

·对内核的性能可以自由的优化。这部分的工作，可由内核的生产者通过适应特殊硬件或软件编程来实现。

传统图形系统的主要缺欠在于，流行的核心系统缺乏对对象进行合符规范的精确描述提供支持。图形对象的生成和编辑，是两项最基本、最常用的功能，它们通常都是用程序设计语言间接描述的，这种间接的描述，使得图形用户界面的实现更加的困难。例如：PHIGS描述的主要是某种特殊类型设备的程序代码。面向对象的交互式的用互界面却不是直接描述的而是通过定义遍历表和这种设备的遍历表置标记而实现的。直观地看，这与程序设计者脑中的模型是不一致的。设计者希望在屏幕上实现的对可视对象的操作，只能以间接的、非自然的，并常常是复杂的方式进行描述。例如，当用户需要对其选定的对象交互式地赋予某种特性时，这种复杂的对象的描述便是必不可少的。在9.4节中，我们将通过几个例子来讨论图形核心的缺欠。

1.1.1.5 现有面向对象核心系统的评价

与传统的图形数据处理技术相并列，面向对象的语言和系统的出现标志着另外一个技术分支的发展。面向对象这一分支与面向对象的语言和面向对象的程序设计环境有关，而后者是作为图形编程环境而实现的。图形编程环境需要图形工具，其结果，由面向对象的环境发展起来的许多工具都成为了通用软件开发的支持工具。随之而来，面向图形的编辑器，支持多个文本并发地工作的窗口系统和支持图形的调试工具等等软件都被开发出来。这些开发深刻地影响了整个计算机技术领域，尤其在现代工作站领域。

与传统的图形核心系统相比较，面向对象的开发系统不能提供丰富的硬件无关的图形编程功能，Smalltalk-80除外，面向对象的开发环境还远不是完整的，这包括：MacAPP，OSF_Motif或Lisp机上的系统。这些系统没有提供复杂的预先定义的类似于传统图形核心中的类型，例如：没有对创建图形菜单和对话箱提供支持。另一方面，它们没有将这些对图形的支持集成在一个构造的模型中，并维护图形对象的层次，也没有提供可以广泛使用的图形语义学的类似类型。

对标准输入设备的控制，在传统图形系统中，程序通过调用很少的核心程序就可以实现；然而对选取（PICK）设备这类更高层次的输入设备的概念，GKS和PHIGS都略去了，而这种概念因其代表输入和图形对象之间的联系而变得极为重要。

缺乏好的图形支持通常会带来不利的结果。在开发专家系统这类基于知识的系统中，现

代杂交软件开发环境以复杂的方式集成了多方面研究成果，使得开发环境往往是面向对象的，规则化的，逻辑化的，约束性的程序设计。相比之下，传统图形学的用户界面通常限于不系统的增加一些所谓的“附加过程”，这意味着程序是纯过程性的。

1.1.1.6 综合的必要性

总的来说，将面向对象的图形学的新思想和传统计算机图形学积累的实际经验结合起来是有必要的。对于一个更精制的图形核心系统的规范来说，传统图形学能够提供所有功能，而面向对象的结构，则能提高质量，例如，由于继承性而带来的灵活性，面向对象的名称概念带来的透明性与及新的系统结构带来的模块性。

1.2 图形系统设计准则

以下提供了交互式图形系统设计的一些指导线索。

1.2.1 从思维模型到程序代码的转换

交互式图形系统的应用程序开发者，通过调用系统功能来实现交互式图形应用系统。因而交互式图形系统有时又叫应用程序开发者的界面。一个交互式的图形应用程序的开发通常都要分成多个阶段，在初期阶段中，开发者之间讨论，最终用户提出对系统性能的要求。这一过程如何进行呢？按我们的观点，在最初的阶段，这一般是以直观面向对象的方式进行的。讨论包括，在对话的什么状态应该存在什么样的对象，这些对象怎样才能被观察到，并且这些对象在什么样的程度上可被扩大、修改或删除。在这个过程中，将要实现的应用系统从功能方面形成了“思维模型”。要实现一个真正的交互式应用系统，思维模型必须被转换成交互式图形学系统的构造。这个过程如图1.2所示。

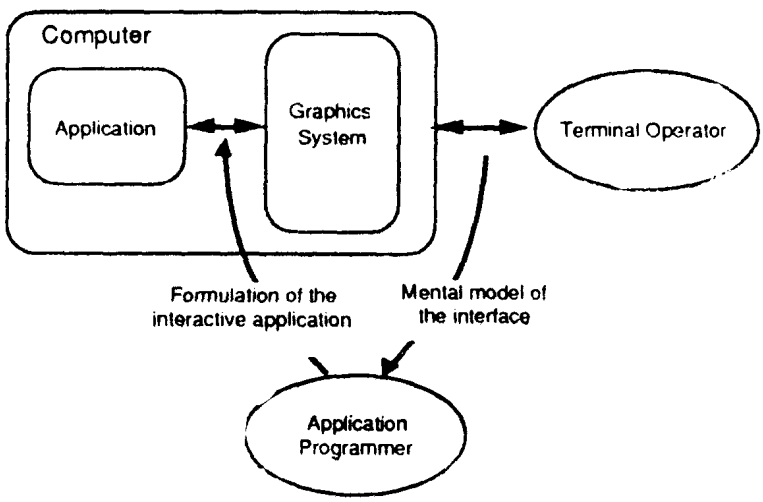

图1.2 思维模型的转换。应用程序设计者根据用户使用交互式图形系统的情况建立一个思维模型，然后将这个模型转换成图形学系核中构造。哪一个图形系统允许模型的自然生成？面向对象的办法比别的办法更优吗？这样问题都是本书的中心问题

交互式图形系统优越性能的一个基本方面，在于它易于将思维模型平滑而自然地转换成图形系统的语言构造（数据结构和操作）。在思维模型的转换过程中，数据结构方面的冲突

越少，交互式图形系统的性能也就越优。因此，验证面向对象的程序设计的框架是否对图形对象的直观概念提供了自然的表达，是一件很有意义的事。然而在这一阶段，应该假定，面向对象的途径在以下几方面提供了很好的潜在因素。

· 面向对象的道路，促使人们按对象这个概念去思维，也即是使他们与高层结构和方法结合起来。这种程序系统中的对象的结构，能够通过适当的定义与直观的图形对象的项很好地一致起来。

· 面向对象的方法允许人们构造系统，最大限度地超越现成的系统。交互式图形界面的功能和其描述代码间的概念上的矛盾，能够得到避免，这是优于其它方面的、

按对象的概念去思考，导致了图形系统功能的自然描述。这包含：

· 对可视项的直接命名。

· 描述、赋值及修改等对对象属性的处理，作为这些对象的继承信息。

· 对并发的图形对象的更方便的捕捉，

· 就象观察普通类那样，可以以相同的方式去观察不同的对象。

除了思维模型到程序设计语言的转换，信息的隐藏性也是面向对象的图形系统区别于传统系统的重要因素。

1.2.1.1 信息的隐藏

对信息的隐藏也是软件技术质量的标志。对交互式图形系统，这意味着：

· 在构造图形信息的时候，没有过份的描述信息，如果可能的话，每一个接口的过渡都应该由图形系统的相应操作一一对应地表示。

· 系统核心的内部实现应该是隐藏的。对传统系统而言，即是说，核心的所有功能的增加或修改，其实现由核心以上的软件层来完成，核心的内部实现不作任何改动。在面向对象的系统中，不修向核心的功能这一原则仍然应该尊守，但是层次模型可以被程序扩充，例如通过定义子类。

1.2.1.2 传统系统的局限

传统系统存在多方面的缺陷，这可以从图形系统发展的历史去寻找到原因。

· 交互式图形设备多年来一直停留在较低的性能水准上，这不能不提及。较低的硬件性能只能产生对图形实体的非面向对象的编码。

· 较低的性能只能针对设备的能力引发面向界面的程序设计。因而应用程序设计者从图形核心系统所得到的功能与设备驱动界面没有太大的区别。

· 没有面向对象的程序设计语言。从实际应用的观点看，现在它们才被广泛使用起来。因而图形编程系统只能以常规语言来表示。此外图形领域长期处在FORTRAN世界的领地中。

由于历史原因导致的问题，缺乏合适的程序设计语言、甚至对现在程序设计的概念的不熟悉都导致了系统的实际功能以一种迂回的方式来实现。一个典型的例子便是PHIGS的旅行者模型，其中线性排序的结构产生了太多不自然的描述，这里既没有实现面向对象的原理，也没有实现信息的隐藏。由于序列化的安排，使得并行显示内部结构或对部分层次结构中的可视对象进行赋值变得比较困难。其结果和解决办法在后面我们将会详细讨论。另一个例子是GKS不能令人满意的功能。通常复杂的程序都必须处理多级的层次结构，而GKS只能支持单层的对象结构。程序设计因而常常要解决应用系统和图形核心之间的结构的不一致。

1.2.2 利用面向对象方法的优点

虽然在以后的章节中，我们将会讨论面向对象方法的实质，但是这里我们不妨对这些章节的特点先作一个概述，这将有助于对图形模型的构造和程序设计。

· 如前所述，面向对象的概念用于图形的最大好处是其直观性。对象的思想，使得图形在外观和行为上得到了更加直接的描述。局部性的原则得到了采用，如果命名了一个对象那么它的继承信息，如属性，就可以赋值，并按名字查询。直接的命名避免了这么一种情况——重要的信息仅能够在进一步的操作之后才能获得。这个概念使得应用程序设计者对多层次结构的图形信息的赋值和查询大大简化，第九章供出的例子说明了这一点。

· 在面向对象的系统中，多重引用是琐碎而平常的。在计算机图形学中，要实现对预先定义的几何描述的访问，多重引用复杂结构是必可少的。

· 使用面向对象的引用机制，对几何描述的一致修改，亦即描述的新版本，是能够很容易得到的。另一方面，如果希望不一致的化，个别的改动亦是可以的，面向对象的系统提供了有效的拷贝机制。

· 面向对象的命名概念，提供了更进一步的优点。实际上，在符号标识符的名称和其所代表的实际对象之前，并无差别。在传统的系统中，段式结构必须被地址所标识，这通常是整数或变量。等价的名称和对象则根本不存在，存在的仅是不同的实体之间的双向关系，这种关系由应用程序设计者来安排。

· 在面向对象的系统中，对象被看成是一种抽象的数据类型，代表一种并发的存在和智能化的实体，它可以与其它的对象进行灵活的通讯。对象之间可以相互交换数据并请求服务。并发存在的具有一定功能的对象与其它方面实体进行准确的通讯，这些方面包括：由图形核心管理的，实例原语与及这些原语的收集，具有灵活通讯机制的应用对象。

· 在传统的编程环境中，取用核心系统的原语来构造图形对象的工作，通常是在相应的子过程中进行的。这种子过程不提供内部存贮空间，因而也不能保存任何非图形应用的数据。这些数据要在名程序与子程序间传递，这只能通过参数表的方式来完成。图形数据的存贮系由统图形程序提供的，如PHIGS中的结构。相比之下，面向对象的系统提供了构造虚拟图形对象的可能性。它们可以通过预先定义的核心类的子类来构造。从应用程序设计者的观点来看，这些对象象普通类一样被集成到系统中去了。但在内部，实际上虚拟对象存贮其自己的数据，并可以接其自己的方式来显示。

· 面向对象的系统提供了灵活的通讯机制。在传统的层次模型中，系统核心的部件通常是被动的，它们决不会自身去激活应用过程。但在面向对象的系统中，图形核心的对象可以向应用对象发送消息。通过被称为“唤回”（call-back）的示例性的特殊消息，从核心对象到应用对象的通讯便可以实现，而不需要修改核心，因而不会违背隐藏性和不修改核心的原则。有了唤回的机制，交互式系统对用户操作。反应，就能以一种更加自然的方式建造模型，这一点远优于传统的系统。用户激活一个输入工具以给图形对象发送信息；图形对象接收到信息，判断其含意，也许会将用户的操作通知给另一个应用对象。相对照，传统的系统需要一个由中央控制的对话机制。应用程序员实现一个中央对话控制器，询问输入工具，解释输入内容，并据此分支转移到应用程序或者图形核心系统中的其它独立代码段去。

· 在多重应用的图形系统中，采用了继承性。为应用的特殊需要而规定图形核心的一些属性，这作为一种程序设计技术广为采用。这比在系统中要容易得多。更进一步，采用预先

定义的核心类来构造特殊的子类，并修改其功能，在很多情况下，这种方法比在核心外增加新的层次要自然得多。继承性，使得面向应用的知识可以被集成到图形对象中，而不失去其功能。甚至，如象新的属性或是特殊原语之类的新的图形功能能够被集成起来，作为预定义的核心类的子类范例。这是重要的，如同在传统的系统中一样，在核心系统之上的更高的层次，可以通过引入新的类和方法来实现，而这些类正是通过消息的传递来引用核心功能的。

通过将预定义的类与隐藏实现、通信的灵活性：预定义类的继承性等因素结合起来，一个新的体系结构就可以建立起来，我们称之为扩展层次模型。

1.2.2.1 精制，而不仅是功能强大

在这本书中，我们比较了图形系统，我们感兴趣的不是性能。精制——应用程序设计者所创建的接口的舒适性，对我们才是更重要的。我们的讨论限于两维图形的范围，因为两维图形已经足够阐明面向对象办法的精髓。

在事先确定的功能上，面向对象的系统比传统的系统并不占优势。用两种方法都可以实现复杂的图形算法。对我们来说，降低开销是重要的，这包括在模型译码和交互式图形界面的设计上的开销。进一步的衡量标准是：在设定图形系统以满足某种类型应用时应有的灵活性，与及经过适当工作，对交互式图形界面的现存版本升级的可能性。要求系统精制，并不意味在设计现代图形系统时，要忽略功能强的新算法，比如三维图形目标的表达。

1.2.2.2 什么样的类和对象？

面向对修的方法并非灵丹妙药，它不会把交互式图形系统的理想化的规范自动实现。它仅是一种指导思想，给用户编程留有很大的自由度；在构造对象，方法和类的层次结构上，一切工作仍是由应用程序设技者来确定的。应用系统的细节描述不是一个可以自动进行的公式化过程。

按照下面的方法，我们来定义图形系统的类。首先通过相应的类和方法，我们可以定义属于同一类的可显示实体。这些实体包括单个的图形对象和其几何描述，前者我们称之为部件而后者称之为建造模式，简言之，即是所有可被操作、修改和赋予属性的东西。这里，可视对象的属性不仅应该是可以赋值的，并且对象的属性对于应用程序设计者来说，应该是可获得的，甚至提供一套由不同种属性组成的属性集。很多这类的思想都是由已有的面向对象的图形系统引申而得的，如Smalltalk-80图形核心。其基本的功能规范，则由传统的计算机图形学给出，这在图形学标准中体现出来。

1.2.2.3 部件层次结构的语义扩展

传统的图形系统深受图形概念的影响，这在部件的层次结构中反映出来。即使GKS也允许将图形源语收集在段落中，向单元那样处理，它们可以被赋予某种属性并被描入到其它段落中。PHIGS在模型的层次结构方面走得更远，它提供了完全可编辑结构的多层次系统。

多层次系统支持由小的单元建造聚集体的行为，并将聚集体作为新的实体，该实体可以处理为更高层的图形源语。多层次系统的这种结构上的便利支持了部件层次结构的语义概念。

面向对象系统的继承性将一个新的语义概念引入图形学中，即：从类到子类的特性的传递。将部件和继承性结合起来，两个重要的基本语义概念就可以形成，这些概念在人工智能

基于知识的系统中是很重要的。

应该认识到，计算机图形学领域一直是围绕着部件层次结构化的应用的，但在其中却处于从属的地位，别的语义起着支配性的作用，如：不同对象之间的几何关系的规则，而不是它们在层次结构中的顺序。其它的语义关系通常被调整以满足某种或多种不同类型的应用，因而在通用图形系统中，对它们提供直接的支持意义不大。要支持应用规定的语义，那种能够与局限评价器或规则解释器等工具结合起来的图形核心系统，将是理想的。规则被调整以适应于图形数据与应用之间的复杂关系的构造。要实现图形系统的这种复杂关系，要考虑的东西不仅于此。对层次结构的精制的访问和修改是不够的。对图形部件层次结构无关信息的简便访问也是必不可少的。做到这一点的先决条件，是对图形信息的有效询问机制与及对与部件层次结构无关的信息进行操作的可能性。面向对象的方法对这种操作的实现也提供了支持。

1.3 综述

为使读者更容易了解本书的组织，这里我们对本书的组成作一综述。

1.3.1 全书的焦点

我们将集中论述交互式图形学中的问题。图形数据结构的构造和编辑将放在突出的位置上。我们将以图形核心系统贯穿讨论，对于几何模型、光照模型、计算机动画等问题，则不予讨论，

1.3.1.1 Smalltalk-80中的编程

通过简要介绍Smalltalk-80，我们解释了面向对象程序设计的基本原理。这样读者便应该能够理解按Smalltalk-80语法写的GEO＋＋的功能，并读懂程序实例。

1.3.1.2 面向对象系统的界面设计原理

与交互式图形系统有关的问题中，交互式用户界面及其支持的设计是十分有趣的。我们将用两个例子来讨论这类系统的基本原理：Smalltall-80中的模型——观察——控制器和GINA。在这类系统中，有可能给应用程序设计者提供预先定义的、可执行的应用程序，因而应用程序设计者可以由通用程序剪编生成自己的面向对象的应用程序。

1.3.1.3 图形学在Smalltalk-80中

在Smalltalk-80中的图形学的应用，被从几方面进行了讨论，这在后面的几章中是很重要的。我们将特别注意图形学在Smalltalk-80、GKS和PHIGS中应用的不同。

从功能上看，Smalltalk-80图形核心在很大范围上与GKS和PHIGS不同。一个重要的特征，便是它不支持对象层次结构，哪怕仅一层。其结果，它没有预先定义的图形的图画结构模型和选取操作。大体上，系统仅提供了单个的对象。所有图形对象层次结构的类型，必须由编程工具通过程序设计来实现，这些编程工具使用起来非常舒适。

我们前面提到过的许多面向对象的原则，都在Smalltalk-80图形核心中得到实现。因此虽然核心的功能是相当低级的，但对于图形系统功能的增强，Smalltalk-80能提供有益的参考。

所有Smalltalk-80中的图形源语均是对象，因而它们能被按名字调用并重复引用。它们能够被直接赋予某种属性，而且也能被直接查询。核心的一个非常重要的特性便是，源语可以被编辑。可编辑的源语，对于图形输入的实现特别有用。除了对图形源语的编辑外，对对象几何参数的设置和查询的可能性，也是要紧的。曾经在GKS和PHIGS开发中作为指导思

想的最小化原则，不再使用，系统提供了一整套丰编的功能。

1.3.1.4 从面向对象的观点来看GKS

GKS和PHIGS的功能都可以用特别的程序设计语言独立地描述。这种特殊的语言可以由Smalltalltalk-80来定义。从纯推理的观点来看，由于Smalltalk-80是面向对象的，这种特殊语言将导至一个面向对象的系统。当然，新的系统在某种程度上有所退化，但这确是一种面向对象系统发展的方向。

1.3.1.5 部件层次结构模型

部件层次结构中的对象组织，是最常用的用来对现实世界建立模型，并进行描述的方法。部件层次结构模型基于这样一个基本思想：用较简单的对象来构成及描述较复杂的对象。部件层次结构在科学和技术的许多领域中起着很重要的作用。

在计算机图形学中，只要复杂的图形能解释为简单的部件的聚集体，部件层次结构就是很有用的，大多数应用领域都是处于这种情况。交互式图形编辑器，如Macdraw，就是典型的以组成概念为基础的应用程序。这里所谓组成，指从菜单上可以选择到简单的图形学源语，并以之组成较高级的集合体。拷贝操作用于复制部件。并支持多次复制具有相同结构模型的对象。

第六章描述了交互式图形系统中部件层次结构模型的基本要求，为讨论图形系统（如PHIGS)的功能和我们自己提出的GEO++的概念，在第八章中，我们将对一个心信的规范进行描述。

1.3.1.6 PHIGS和部件层次结构

标准的程序设计者的层次化交互式图形系统，是一个程序的开发系统，它能比GKS更直接地支持层次化图形数据结构的建造和编辑。PHIGS支持多层数据结构，这允许部件层次结构在其上实现。这些层次结构象结构网一样，组织起来。

PHIGS的编辑机制有助于任意结构的元素的编辑。几乎PHIGS提供的所有操作都能用来进行部件层次结构的编辑与构造。然而，只有熟练运用PHIGS提供的元素的操作，才能实现部件层次结构的典型操作。PHIGS不能验证一个元素的编辑操作是否正确，也即是，在图形对象的层次结构中是否是无效操作。因此我们认为，PHIGS仅具有部件层次结构的语义方面的有限知识。

1.3.1.7 面向对象的系统GEO++

GEO++（可编辑对象的图形系统）是为面向对象多层次图形系统而设计的，在这里将具体讨论本书提出的思想。GEO++在其描述及应用功能上受面向对象程序设计的影响，其功能是GKO和PHIGS功能的继承和扩展。GEO++是典型的图形核心系统，是独立于应用的，功能足够丰富和普遍的系统，可用于广泛的应用领域。

GEO++采用了面向对象程序设计的普遍原理。图形源语、源语分组、建造模式、属性等都可以按名来编址，并按名来查询，这种对象的标识符是由应用程序设计者来规定的，这在面向对象的系统中是相当普遍的。

GEO++功能规范的另一个方面是，通过普遍接受的对元素的操作来实现对部件层次结构的操作。在这方面GEO++超过了PHIGS提供的低水平编辑操作。GEO++中的单一的原操作能够代替PHIGS所要求的整个一系列的操作。这一原操作，包括询查部件层次结构、编辑层次结构和再建层次结构，如建组及分解组。

1.3.1.8 程序实例

遗憾的是，带有程序实例很好注解的PHIGS的详细描述很难得到，因此在介绍面向对象系统的书中，PHIGS通常占有不大的篇幅。然而另外一方面的原因却要所扩大对PHIGS的处理。我们将把PHIGS作为传统方法的典型代表与GEO++进行比较，以显示面向对象设计的图形系统所具有的优点。

我们将用几个典型的程序实例来作为面向对象概念的示例。然而第九章的示例尚未用到对象的继承性。它们仅受对象这种思维方式的影响。结果，GEO++提出的总的原则，将应用于非面向对象的程序语言中去，例如：ADA，它仅用抽象的数象的数据类型，这种语言甚至可用Pascal和Fortran。

1.3.1.9 预定核心的继承性

第十章给出了几个例子来说明继承性。这里假定核心是隐含实现的。另外几个普通的例子还说明，使用对象的继承性，GEO++的功能很容易被修改和扩展。

1.3.1.10 原型与授权

类实例概念的替代是“原型——授权”模型，当程序运行时，实例的协议也许改变，变量和方法能从已现存的实例中删除或增加进去。实例之间的相互依赖关系，使得一个实例能从另一个实例的功能上获利。

许多专家都强调计算机图形学中原始模型的相关性。在计算机图形学中，对象被交互式编辑程序动态地创建，它们的行为也动态地被改变。要构建模型并实现如此灵活的系统，“原型——授权”概念是非常自然和强有力的。在原型的基础上对GEO++进行修改将会说明这一点。

1.3.1.11 实现的情况

第十二章将要描述GEO++的一个有效实现。GEO++的内部实现可以与从外部看到的模型完全不同。尽管GEO++是以树型部件层次结构为基础的，一个非循环图形的紧凑实现是可行的。这个实现至少可以象PHIGS的实现那样紧凑。

1.3.1.12 附加概念及工具

迄今，在图形系统中，部件层次结构的支持起着主导的作用。当然在计算机不同应用领域中，还需要大量的其它不同的语义。在十三章中，我们将用一些实例讨论这类应用。“联接”这一语义概念被引入，例子将说明预定义的类的子类是如何支持这种概念的。通常，附加的语义是由应用决定的，因而需要对每一种特殊的应用而编程。为减少高级语言编程的工作，讨论了约束的情况，采用的方法是，现特定形式的对象间引入描述其复杂几何关系的概念，并提供评价这一关系的机制。问题的关键是如何才能把约束平滑地加入到预定的面向对象的核心中去。

把现实世界中的现象尽可能自然地映射到计算机系统，这种愿望促进着新的知识表示机制的提出。作为进一步的研究，将计算机图形学的功能与混合表示的机制自然地联系起来，对这种思想我们亦进行了讨论。

1.3.1.13 面向对象的标准？

在第十四章中，我们涉及了面向对象的图形学是否应该有一个标准的问题。目前制定面向对象的标准的条件是非常有利的。但仍然有许多工作要做。