C++和OSF/ Motif 面向对象的混合编程技巧

王劲松　等编

北京希望电脑公司

C++和 OSF / Motif
面向对象的混合编程技巧

王劲松　　等编

北京希望电脑公司

内 容 摘 要

本书是一本将C++的面向对象程序设计方法和Motif的用户界面开发工具结合起来的参考书。本书按内容分成两大部分：第一部分主要介绍C++和Motif，并给出了一个比较简单的例子，阐述了如何让面向对象的C++语言使用结构化的C语言编写的Motif用户界面开发工具；第二部分介绍了Motif应用程序的框架、以及应用该框架构造Motif应用程序，其中包括多用户系统中冗长任务的处理等等。

本书既介绍了使用C++和Motif混合面向对象编程的一般方法，还详细分析了书中列出的大量例子程序。因此本书将主要适用于广大软件工作者、高等院校高年级本科生和研究生，既可以作为介绍C++和Motif的教材，也可以作为开发过程中的参考书。

需要本书的用户，请直接与北京8721信箱联系，电话：2562329，邮政编码：100080。

北京希望电脑公司计算机技术丛书

C++和OSF/Motif

面向对象的混合编程技巧

王劲松 等编

希 望 审校

京准印字：3580—91580

内部成本： 19.00 元

内容简介

本书是一本将 C++的面向对象程序设计方法和 Motif 的用户界面开发工具结合起来的参考书．本书按内容分成两大部分：第一部分主要介绍 C++和 Motif,并给出了一个比较简单的例子，阐述了如何让面向对象的 C++语言使用结构化的 C 语言编写的 Motif 用户界面开发工具；第二部分介绍了 Motif 应用程序的框架、以及应用该框架构造 Motif 应用程序，其中包括多用户系统中冗长任务的处理等等。

本书既介绍了使用 C++和 Motif 混合面向对象编程的一般方法，还详细分析了书中列出的大量例子程序．因此本书将主要适用于广大软件工作者、高等院校高年级本科生和研究生，既可以作为介绍 C++和 Motif 的教材，也可以作为开发过程中的参考书。

目　录

前　言

本书产生自这样一个简单问题:“用 C++进行编程时,怎样使用 Motif 用户界面工具箱?”最直接,也是相当简单的回答是:用它提供解决问题的机制。然而,要全面地阐述这个问题却是一件相当复杂的工作,特别要指出的是,一个完全的答案必须讨论“如何将 X 和 Motif 所支持的程序模块和 C++面向对象式的程序模块有机结合起来”这一问题,这也正是本书的中心议题所在。

本书的另一个基本议题也很重要,即在最近的几年内,程序员的编程风格已有了长足的进步。我们在学校从教科书上学的东西全集中在算法和基本数据结构上,Wirth 教授在他的课本中称“算法+数据=程序”(见《Wirth76》)。没有人对“数据结构和算法是应用程序的核心”持有异议,然而现代的程序仅有这两点还远远不够,对当今编写的大部分程序来说,确定合适的数据结构和正确的算法仅仅是工作的开始。

在现今,大部分应用程序所强调的是基于象 X、Macintosh Toolbox、Microsoft Windows、neXTStep 及其他一些窗口系统具有良好的交互功能的用户界面。基于窗口系统的应用程序,其界面的开发花费占总花费的 50%至 90%,而用于程序计算部分的花费只占一小部分,这不意味着算法的重要性减弱了,只是说明现代交互式应用程序因界面部分的引入,其复杂性大大提高。

程序员有时也许会觉得开发这种新型的程序很困难,这有两方面的原因,其一是交互式界面的开发其本身就很困难;其二,交互式程序的体系结构与以前的大不一样,以前没有接触这种应用程序的程序员往往觉得无从下笔。在学校里所学的算法一般都很简单,它往往只涉及到有输入数据并返回数据的函数。而对现今的程序来说情况就不一样了,首先现代交互式程序都是实时程序;其次,现代应用程序经常需要在显示器上的多个不同区域内显示信息,并且要求有良好的容错性,其界面不论是对专家还是对初学者都应合适等等。

本书的动机之一就是帮助读者理解在应用程序中怎样解决这些问题。仅仅懂得如何显示处理象 Motif widget 提供的用户界面组成部件只是第一步,程序员还须了解 X 和 Motif 对应用程序的体系结构所施加的限制条件。

面向对象程序设计技术能满足现代交互系统的这些要求,不过,面向对象程序设计中,所强调的是结构,比起算法的开发来,使用面向对象技术的程序员更注重的是对象的接口及相互关系。本书的另一动机就是解释 Motif 交互式程序中的面向对象技术。

有人肯定会问:“为什么要把 C++语言和象 Motif 这种基于 Xt 的工具箱结合起来使用,能不能使用一个纯面向对象的工具箱?”许多基于 X 的面向对象式用户界面工具箱对那些认为面向对象式程序必须使用“纯”面向对象库的程序员来说是有用的,这方面的工具箱有斯坦福大学开发的 Interviews 工具箱、瑞士苏黎世大学开发的 ET++工具箱及 Solbourne 等人开发的 OI 工具包等,它们都是用 C++语言写的。这些工具对 C++语言程序员来说是强有力的支持,它们也是很有发展前途的。不过在目前,它们都没有对 Xt 和 Motif 的支持,使用它们的程序员和用户都不多。总有人反对将基于 C 语言的工具包(如 Motif)和 C++语言结合起来使用,如果你也属于这一类的话,很遗憾本书不适合于你来阅读。

目前,差不多有上百种可用的用户界面工具箱,它们支持不同的语言和程序设计风格,

你必须根据它们各自的优点及你的需要来确定采用哪一种，至于工具箱的内部实现采用的是哪一种语言并不重要，你必须考虑的是这几个问题:该工具箱在现在及将来的可用性，程序员及用户的可采纳性、可靠性等等。除此之外，你还必须考虑应用程序的使用环境，及工具箱所支持的用户界面风格是否与该环境下的其它应用程序兼容，在许多情况下(甚至可以说是在大多数情况下)，这些因素是你做决定的关键所在。本书是为那些要使用 Motif，且又想获取象 C++语言所提供的各种优点的读者写的，对那些选择了 Motif 和 C++语言的用户来说，我们衷心地希望本书能对他们有所帮助。

本书中所有的软件均在运行 IRIX 4.0 的 Silicon Graphics Iris Indigo 系统上调试通过。这些例子是用 AT&T 的 C++翻译器 cfront 编译的，它们必须与 X11 R4 Xtintrinsics 及 Motif 1.1.3 连结。这些例子都具有自给性，它们只依赖于标准的 UNIX、C 语言库、X 及 Motif。自给性会给程序的设计带来一些限制，例如，本书中所描述的类如果有一个好的通用库来支持的话，它们会更简单且在性能上也会有所提高，对通用库需求最明显的例子是通用的 List 类，当然其他的也同样有用。对于那些可以使用通用数据结构的类来说，我们也愿意背离本书的主题，不去实现一些必须的类，而是采用现成的库，不过，这样做不利于本书的读者，写出来的例子也不能充分体现这些类的好处。由于实现一个可复用的通用数据结构库应该是另一本书的主题，所以本书中我们不这样做。在本书中我们注意的是单个类的外部特性和一个系统中所有类的主要特征，而不是单个类的内部具体的实现细节。

本书的编写就如同编写一个大型软件，要保证没有错误是不可能的，不过，我们尽可能地减少错误的发生。然而，不管我们尽了多大努力，错误在所难免，恳请读者批评指正，不吝赐教。本书编写的意图是希望它能起一个抛砖引玉的作用，我们希望它能对您有所帮助。

本书的资料收集和编译工作是由王劲松负责组织完成的，参加本项工作的还有清华大学的杨三辅、元涛、李涛和王晓东等。另外，中国科学院自动化研究所的汪晓农也为本书做了一定的工作。在本书的编辑和出版过程中，中国科学院北京希望高级电脑技术公司的秦人华女士提供了许多方面的便利，同样也付出大量的汗水，在此谨向她表示诚挚的谢意。

第一部分　C++和 Motif 简介

本书描述使用 C++和 OSF / Motif 接口工具箱来编写交互式程序的面向对象方法。Motif 基于 Xt Intrinsics，是一个用 C 语言实现、支持面向对象体系的程序库；C++由 C 语言发展而来，但它直接提供面向对象式的编程方法。Xt 和 Motif 支持的对象与 C++的对象完全不同，这给一些程序员以错误的印象，认为他们只能用 C 语言来编写 Motif 应用程序，或者认为他们必须使用 C++的类型来包装 Motif widget。

然而，Motif 内部使用的面向对象式体系对 Motif 的应用程序来说不会有丝毫的影响，Motif 提供面向函数式的接口，调用程序对 Motif 及 Xt 内部实现的细节不必细究，表面上，Motif 和其他 C 语言库没有任何区别。所幸的是，C++的设计允许程序员轻易地使用象 Motif 这类的 C 语言库。事实上，C++提供了一个相当漂亮的方法，它使得程序员在得到 Motif 优点的同时仍可采用面向对象程序设计技术。

利用 C++和 Motif 编程员，必须克服这样一个观点，即程序中出现的任何东西，包括 Motif widget，都必须是 C++的对象。诚然，对象和类是一个面向对象程序的主要特征，然而通常情况下 C++程序照样包含其他数据结构和函数，例如，大部分 UNIX 程序的系统调用都使用不同的 C 语言库函数，即使对 C++程序而言也是一样。尽管我们可以为文件系统、操作系统创建一个面向对象式的界面，但是通常情况下我们使用 C 语言写的 UNIX 库函数就已足够了。

在大多数情况下，面向对象程序设计方法的效益依赖于这种技术对应用体系所产生的效果，至于该应用程序是不是每一个元素都是作为一个对象来实现的并不是关键的问题。尽管如此，要对面向对象式程序设计有一个完全的评价，有一点是很重要的，那就是不论是程序员、设计者，还是系统分析员，他必须了解这种技术的原理，并能够在应用系统最主要的部分中得到应用和体现。

象 C++这类语言，它允许程序员利用面向对象程序设计技术来编写大型软件，同时它允许程序员使用系统调用、库函数及其他一些非面向对象型的基础软件，包括 X Window 系统。C++为面向对象设计方法和系统软件设计方法的结合提供了简易的方法，它使得程序在受益于面向对象技术的同时不失掉 C 的高效性，更不必重新设计象 X 和 Motif 这种标准 C 库函数。

怎样使用本书

为了更好地理解本书，读者应该具备 C++和 Motif 的一些基本知识。第一章简明扼要地介绍了 X 和 Motif。本书在引用材料上假设读者对 C++已经有一定的了解，他至少应能阅读 C++程序。关于 X、Motif 和 C++资料的大量存在，想要知道更多细节的读者有很大的选择余地。下面是关于 Xt、Motif、Xlib 和 C++有限的几本书，它们按照出版日期依次排列如为下：

Astente, Paul, and Palph Swick, The X Window System Toolkit, Digital Press,

1990.
Open Software Foundation, OSF / Motif Programming's Reference, Version 1.1, Prentice Hall, 1990.
Scheifler, Robert W., and James Getttys, X Window System, Second Edition, Digital Press, 1990.
Stroustrup, Bjarne, The C++ Programming Language, Second Edition, Addison-Wesley, 1991.

这些书可读性好，具一定的权威性，它们所讨论的主题超出本书范围之外,因此在对本书后续各章进行更广泛地讨论时，我们极力推荐以此作为参考资料。在本书最后的参考文献里还有其他一些书籍供读者参考。本书的基本议题是讨论程序员在编程时如何充分利用 C++和 Motif 的优点。除非与主题讨论密切相关，否则的话，本书不对 X 和 Motif 函数的工作原理作细致的讲解，因为有关这方面的知识读者很容易从别的资料上获取(例如上面列出的的书籍中)。

本书分为两大部分。第一部分介绍使用 C++和 Motif 的方法，着重强调面向对象技术。第二部分剖析使用 Motif 和 C++的另外一些方法,这些方法在开发支持 Motif 应用程序的小型类库时得到了应用。

本书中大部分的例子都着重强调每个程序思路的发展过程，在大部分情况下，我们的讨论涉及到选取不同决策的原因，可替换的方法等。我们不是为用 C++开发 Motif 应用程序提供唯一"正确"的方法，而是剖析，讨论不同的方法及各种可能性。

本书的编写基于这样一个假定，即读者能够从头至尾依次阅读。然而，由于读者的背景知识和兴趣有很大差异，许多读者肯定不必这样做。每章中的小结可供那些想很快了解这些基本知识、或根本不需要序言中提到的背景知识的读者参考。

第一部分　C++和 Motif 简介

第一章　使用 C++示教 Motif，本章从 C++程序员的角度介绍了 X Window 系统、Xt Intrinsies 和 Motif widget 集合。那些已熟悉 Motif 或其他基于 Xt 的工具箱的读者可以略过本章。然而，每个人都必须阅读本章中的例子及有讨论 C++语言风格对 Motif 程序的影响的有关部分，这些讨论从第 1.2 节开始。

第二章　C++类和 widget，介绍本书中用来获取 C++的类和 Motif widget 优点的方法。本章内容是本书其他部分的基础。

第三章　用对象进行设计，讨论面向对象程序设计的一些方法。本章介绍的设计技术及记号在本书其他章节的例子中经常用到。

第四章　九格子游戏的设计，应用第三章所讨论的技术来设计一个简单 Motif 程序。

第五章　九格子游戏的实现，实现了第四章中设计出来的程序，它采用了第二章所介绍的有关技术。

第二部分　应用框架

第六章　MotifApp 应用框架，介绍了一个简单的应用框架体系，它是一个用 C++和 Motif 编写的可复用的类库。本章介绍了一个被称之为 MotifApp 的“框架”,Application 类和 MainWindow 类是构成这个框架的基础，亦在本章中讲解，此框架中的其他类将在后续各章中得到讲解

第七章　对话，讨论了 Motif 应用程序中使用“对话”的方法，同时设计了一些类，这些类的使用使得在 Motif 应用程序中使用对话变得异常简单。

第八章　命令，讨论了一些抽象类集合，主些抽象类常用来作为 MotifApp 框架中的命令模型。

第九章　一个简单的菜单程序包，讲解了 MotifApp 中的菜单系统，它使用了第八章中所提到的类。

第十章　冗长任务，讨论了程序员编写交互式应用程序(它的一些操作的完成往往需要一段比较长的持续的时间)时所面临的困难。本章中讲解了象 MotifApp 这类框架中所提供的便利措施，利用这些措施可以邦肋程序员解决这一公共的难题。

第十一章　颜色选择器，提供了一个比较复杂的例子，该例子自包含用 C++语言和用户界面组成部件。这些可复用的组成部件允许用户交互地选择颜色，它们基于一个被称之为模型-视察-控制(MVC)的面向对象式的体系结构。

第十二章　一个 MotifApp 应用程序，讲解了一个更大的例子，本例使用了本章前面各章中提到过的 MotifApp 框架的诸多特点。

本书使用的约定

本书遵循通用的 C++和面向对象程序设计方法中用的命名和编码约定。所有的变量名、类名都是大小写并用，其中每一个词的词头大写。类名以大写字母开头，而实例名则以小写字母开头。例如:

```
MainWindow      * myWindow;
```

成员和成员函数也是大小写并用，它们都以小写母开头，例如:

```
myWindow->classname();
```

有时，一个程序中的某一特定类也许只有一个实例，在本书中如有这种情况发生,则我们在该实例名冠以单词“the”，例如:

```
Application * theApplication;
```

本书中的类都尽可能的使用封装技术，因此所有的类都不含共有数据成员，对于私有符号我们仍然采用 c 程序员通用的约定:在成员数据前冠以字符"–"以提醒读者该成员私有，其他类要存取一个类的成员数据必须通过该类所提供的内联函数来进行。对私有数据成员的命名冠以下划线可允许访问函数与之同名，当然访问函数不能也以下划线开头。基于这些约定，一个含有其他类要访问数据的类的样板可以这样来写:

```
class SampleClass {
    private:
        int    __someData;      / / An internally–used int member
        char *   aString;       / / An internally–used string member
    public:
        SampleClass();          / / Constructor;
        / / Acess functions
        int someData() { return(__someData); }
        const char * const aString() { return(__aString); }
}
```

如果要为一个类留一个外部入口以便设置它的成员数据，可以为该类设置一个赋值函数用来给保护数据赋值。在上面的例子中，如果我们想把以前的例子加以扩展,以便可以修改成员数据 someDate，我们可以提供这样一个 setSomeDate()成员函数:

```
class SampleClass {
    private:
        int    __someData;      / / An internally–used int member
        char * __aString;       / / An internally–used string member
    public:
        SampleClass();          / / Constructor;
        / / Acess functions
        int someData() { return(__someData); }
        const char * const aString() { return(__aString); }
        / / Function sets private data
        void setSomeData(int newValue) { __someData = newValue; }
}
```

这本书给出了许多 C++的类，按惯例我们将这些类分开来单独存放，每个类都放在两个文件中，即在头文件中声明类的定义，而在另一文件中定义类的成员函数。头文件的文件以中类名和.h 后缀（如 Stack.h 等）构成，而源文件的文件名由类名后加.c 后缀(如 Stack.c 等)构成。当然，你所用的环境也许使用.C、.c、.cc、.cxx 或其他一些约定，这是可行的，有些程序员喜欢给头文件冠以.H 后缀，这也未尝不可。在本书中，我们一律使

用.h 和.c。

本书中大部分的源码清单都带有行号。对源码的讨论常常分别进行，每一段包含一些独立的函数，然而对任一给定文件来说其行号是连续的，每一文件的行号都由 1 开始。有时候，对那些不完全的例子，或与本模块无关的例子，我们不加行号，以示区别。

为了防止重复定义(在有嵌套头文件时，这类操作容易发生)，所以的头文件都具有这种形式:

```
#ifndef   CLASSNAME__H
#define   CLASSNAME__H
//   class declaraction goes here
#endif
```

在实际的文件中，将类的名字取代上面的 CLASSNAME，例如，一个含有名叫 Stack 的类的头文件可以这样来写:

```
1   ////////////////////////////
2   // stack.h:Header file for the Stack class
3   ////////////////////////////
4   #ifndef STACK__H
5   #define STACK__H
6
7   class Stack {
8
9   public:
10
11    stack();   //   Constructor
12    //   Miscellaneous members
13  };
14  #endif
```

文件 Stack.c 含所有 Stack 成员函数的定义如下:

```
1   ////////////////////////////
2   // Stack.C:Implementation file for the Stack class
3   ////////////////////////////
4   #include "Stack.h-
5
6   Stack: :Stack()
7   {
```

```
8    // Various initialization statements
9    }
```

我们可以注意到 Stack.c 中包含 Stack.h，在 Stack.h 中有类声明，其他任何要引用 Stack 类对象的类也同样需要包含 Stack.h。由于头文件可能被同一程序中不同的文件多次引用，因此头文件中只可有声明，而不要有函数及变量的定义(内联函数除外)。

C++语言中使用的术语和其他面向对象型语言有所不同。C++中的"成员函数(member function)"相当于"方法(method)"，"调用成员函数"相当于"发送消息(message)"，"基类(base class)"和"派生类(derived class)"分别等同于"超类(superclass)"和子类(subclass)"等等。本书大体上使用 C++的术语，但当别的方式更有效时，我们也使用其他一些系统的面向对象型的术语。

第一章　使用 C++示教 Motif

本章介绍 X Window 系统和 Motif widget 集。X Window 系统有时也简称为"X",它是一种工业标准，提供了一个可移植的图形用户界面系统。Motif 是一个高层工具箱，它为 X 应用程序提供通用的用户界面组成部件。本章中的例子全都写 C++写出，但我们不注重讲面向对象技术，在第二章中我们介绍了使用 Motif 和 C++的一些面向对象技术。

1.1 节介绍了 X 和 Motif widget 集的体系结构，然后，在 1.2 节中我们讨论了函数原型，还讨论了将象 X 和 Motif 这类的 C 语言库和 C++应用程序连结起来的方法; 在 1.3 节中介绍了 Motif 应用程序所使用的基本程序设计模型; 1.4 节概要阐述了组装 Motifwidget 的用户界面组成部件; 最后，在 1.5 节中我们介绍了 X 的资源管理器，它是定做应用程序和 widget 的工具。

1.1　X／Motif 的层次结构

Motif 应用程序建立在三个不同的层次上。Xlib 为基本的 Window 系统提供底层的服务接口; Xt lntrinsics 库提供了一个高层的工具箱体系; Motif 提供一些组成部件集，用它可以创建用户界面。本节介绍了 X、Xt Intrinsics 和 Motif 之间关系。

客户与服务器

X Window 系统的体系结构基于"客户一服务器"模型。一个被称之为服务器的负责管理输入输出物理设备，服务器在屏幕上创立并管理窗口、显示正文及图形、并接收输入信号。基于 X 的应用程序本身并不直接往显示屏写任何东西，而是通过申请 X 服务器来完成画图功能。与此相类似，基于 X 的应用程序其本身也并不直接从键盘或鼠标器接收输入，而是依靠 X 服务器来完成(当这些设备数据准备好以后，服务器给应用程序发送消息)。通过服务器来处理输入、输出，使得 X 应用程序与显示的硬件设备分离，只要使用正确，基于 X 的应用程序可以在任何显示设备上显示窗口、正文和图形。当然该设备必须支持 X 模型。

请求和事件

当一个客户想使用 X 服务器时，它向 X 服务器发送请求信号，例如，用户程序可以请求服务器在显示器上开窗口或画一条直线。一个请求信号是一个简单的信息包，X 服务器依次处理所有客户发送来的请求。它串行地处理发送来的请求，因而不是对每一请求都立即处理。还有一点，由于服务器和客户程序都是异步运行，因此 X 服务器不可能保证按先后次序来处理来自不同用户的请求。

当输入准备好以后，服务器经过发送"事件"来通知客户程序。事件在 X 中是用 C 语言的联合结构来表示的，每一事件结构指明该事件的性质类别及其他一些信息。例如，当用户在键盘上敲击一键时，X 服务器会产生事件 KeyPress，除了指照该事件的类型外,KeyPress 的事件结构还指照用户敲了什么键，如果是鼠标的话，则指照鼠标的位置及其

他一些信息。

X 服务器为每一进程都设置一个先进先出(FIFO)的事件队列，应用程序可由此读取事件。服务器只向那些发送了相应事件请求信号的客户程序报告事件的发生，且事件的发生应在客户程序的某一窗口内。尽管服务按照发生的先后顺序依次报告事件的发生，客户程序却没有必要立即处理这些事件，它可以在任何时候读事件队列。在事件发生时，客户程序可以处于"忙"状态。

通常，X 应用程序连续地处理事件队列，尽可能早地读入事件。这有多方面的原因，其中之一是:X 窗口不是"坚持型"的，也就是说，当窗口被移动、覆盖或改变大小时，其内容会丢失。X 保存每一窗口的背景模式及窗口的操作边界，而用户在窗口内显示的任何东西都是瞬态的，因而可能丢失，当这种事情发生时，X 向用户程序发送 expose 事件以通知某一特定窗口的内容已丢失。设计好的应用程序应随时准备接收这种信号，以便能及时恢复窗口的内容。

窗口

X 服务器最基本的工作是显示并管理窗口。在 X 窗口，一个窗口就是显示屏上的一个矩形区域，服务程序根据用户的要求而创建、删除、管理窗口。每一个窗口都有唯一的标识，在对窗口进行管理时，用户必须提供该标识。

X 中的窗口形成一个层次结构。X 的服务器在启动时，它创建一个被称之的根窗口(root Window)的特殊窗口，该窗口即整个显示屏，其他窗口都是根窗口的子孙，每一个窗口都可以有子孙窗口;除了根窗口外，任何窗口都有父窗口。

X 支持可重叠窗口模型，窗口就象书桌上的报纸一样可以重叠起来。兄弟窗口(即共享同一父窗口的子窗口)有一个重叠次序(stacking order)，用以决定究竟哪一个窗口在最上面，哪些窗口完全或部分地被其他窗口覆盖。

窗口不一定非要在显示器(CRT 或显示屏)上出现。服务程序创建的窗口起初是不可见的，用户通过发送 map 请求来让 X 服务器在显示屏上显示窗口，通过 unmap 请求来删除显示屏上的窗口。不管窗口是否已在显示屏上显示，它都可以被移动、改变大小或删除。

用户根据自己的需要来请求服务器对窗口进行管理，他可以增加或减小窗口的重叠次序号，改变和父窗口的相对位置，在屏幕上显示或撤消窗口等等。窗口根据某一用户的请求而创建，在该用户退出或切断服务器之间的联系时窗口消失。然而，在窗口的生命期内，任何知道该窗口标识号(id)的用户都可以据此请求 X 服务器对其进行管理。

X 的图形软件

X 提供一些基本的绘图功能。客户可以请求服务器在窗口中画直线、圆、矩形和多边形。X 允许用户以各种线宽，以显示设备支持的任何颜色，以及各种模式画直线。封闭图可以用某种模式或颜色填充。基本的 X 协议仅支持整数型座标系。

所有的图形请求都必须是可绘画的，可以画成一个窗口或一个象素图。象素图可以是内存区域内的一段内容。并可以作为图形请求的目标。象素图可以在复制到某一窗口之前，在屏幕之下进行存贮和合成图象。象素图可以保存在 X 服务器中，而且有唯一的标