基于 java 的可扩展矢量图形编辑器设计与实现

张红鹰 （蚌埠安徽财经大学 233000）
刘连忠 （合肥中国科技大学 230000）

摘要：针对计算机工作人员的各类图形绘制工作，设计了基于 java 的可扩展矢量图形编辑器，文中给出了系统的主要功能模块设计，并作了详细的说明。

关键词：java 矢量图形 编辑器 设计

本系统旨在提供一个可通过键盘和鼠标绘制矢量图形的平台，它可以实现基本图形的绘制、移动、删除、着色、填充、保存、图形之间的关联线绘制，且关联线可以随图形的移动而自动变化，从而可以高效、准确、无失真地绘制出各种图形，大大减轻绘图人员的工作量。

1 设计概要

可扩展矢量图形编辑器采用了面向对象的图形绘制方法，通过将图形所包含的各种对象定义为图元，并将各种图元进行分类组织，形成图元库。在绘制图形时可以利用图元库中的图元进行组合，即为组合图，从而有较好的可重用性。当图形绘制完成后可对其进行编辑，可以设置图形中图元的属性，以及图形的关联等。

1.1 基本功能

本系统包括以下基本功能：

（1）绘图。通过编写脚本来实现绘制简单的图形，如圆、椭圆、菱形、矩形、线段等。

（2）图形连接。不同的基本图形连接用线段实现，而且线段只能与基本图形的特殊点（如上面定义的属性）相连。

（3）图形调整。可以对脚本绘制的图形用脚本命令进行移动、缩放，修改等操作，而且要求有关联的图形做相应的修改。如移动一个圆，则与之相连的线段也做相应的移动和修改。

（4）可以动态修改脚本，即发现错误后可以及时修改。

（5）所见所得，即脚本修改后可以立即执行，视图就是相应的修改。

1.2 高级功能

扩展系统功能，使可以完成图形的 xml 格式文件的保存、填色、鼠标、键盘双重方法输入脚本命令。

（1）用户可以根据实际需要定义自己的语法，编写相应的模块。

（2）动态加载用户自定义模块，从而实现用户的特定功能（例如绘制特定复杂的图形）。

（3）图形色彩处理。增加图形的颜色属性，可对图形着色，填充单色、渐近色等。

（4）图形的 xml 格式文件的保存，以增强生成图形文件的可复用性，例如可直接用于 WEB 发布。

1.3 运行环境

本系统采用 JAVA 实现，故与操作系统无关，用户的系统配置的最低要求如下：

Memory：32MB minimum OS：Windows x，linux，Sparc，etc

CPU：200MHz or above JDK J2SDK 1.5.0

2 系统设计

2.1 系统结构设计

本设计是基于 MVC 的架构模式。

Model 端（ShapeModel）：在 MVC 中，Model 便是执行某些任务的代码，而这部分代码并没有任何逻辑决定它对客户端的表示方法。Model 端只有纯粹的功能性接口，也就是一系列的公开方法。通过这些公开方法，便可以取得 Model 端的所有功能。在 Java 语言中，一个 Model 可以继承 java. util. Observable 类，此父

方法可以提供登记和通知视图所需的接口。在设计中 Model 为 ShapeModel 以列表的形式保存图形对象。.

View 端(DrawingCanvas):一个 Model 可以有几个视图端,而实际上复数的视图端是使 MVC 的原始动机。本设计使用 Java 中的 java. util. Observer 接口。DrawingCanvas 类继承 Observer,作为图形对象的视图,实际上就是画布。

Controller 端:在本设计中对于每一个图形的生成,都有一个相应的控制器管理。而对图形对象的编辑则由另一类控制器管理。包括移动,删除,缩放等操作。

系统设计原型图见图 1:

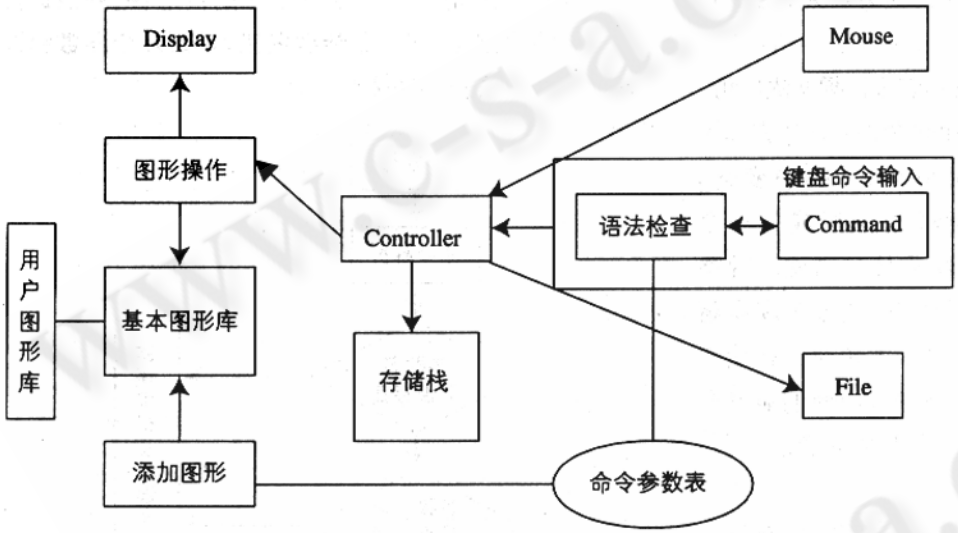

图 1　系统设计原型图

2.2 系统主要组成模块

(1) 图形模块。主要提供直线、椭圆(包括园)、矩形、文本等基本图形类,这些类提供了图形的绘制方法,以及移动、缩放、等编辑方法。

(2) 图形控制器。主要完成对图形的生成。对与每一个图元都有一个对应的图形控制器,控制该图元的生成。可以用过命令和鼠标两种作图方式进行调用。

(3) 工具箱。主要提供鼠标作图的工具,包括各种图元以及一些简单的编辑方法,如删除、填充颜色等。

(4) 命令编辑框。主要实现使用命令进行作图的功能,包括图形生成、缩放、删除、改变颜色、填充、使用渐进色、移动等。

(5) 语法检查及错误定位。此为命令编辑的主要包含部分,使用命令生成,编辑图形,必须对其输入的命令行进行语法检查,只有正确的命令才可以执行。并对其错误命令进行简单的错误定位。

(6) 转换器。本编辑器使用两种作图方式,但是真正的生成图形时只用一个命令即可,即要将命令作图和鼠标作图两种命令转换为一种命令。转换器实现了此功能。

2.3 系统实现

2.3.1 图元的设计及其存储方式

(1) 图元的数据组织。组织面向对象的文档存储机制,对图形系统来说,就是怎么组织图形系统的数据。首先需要确定,在图形系统的运行过程中,图形数据是放在内存中,还是放在硬盘中。

一种方法把必要的用于管理的数据(如图形元素的层)放在内存中,而把图形元属的实际数据(如图形元素的坐标位置)放在硬盘的临时文件中,两者建立管理和存储机制。这种图形元素的存储优点可以利用很小的内存空间实现大容量的图形系统,缺点是系统的存储组织设计复杂。且在用 java 实现时,此类的独立性差,不是完全面向对象的设计。

另一种方法是完全面向对象的实现机制,即把一个图形类的一个对象作为一个整体来管理,通过类创建很多对象,把每个对象作为一个整体来组织存储空间的分配,存储,读取等管理工作。这种方法的缺点是需要较大的内存空间,但它组织简单,结构化和移植性好。可以用 java 来实现。所有我们采用的是第二种方式。

(2) 图元的数据结构。在图形系统的运行过程中,图形数据放在内存里,因此图形数据结构的设计是否合理直接影响图形处理效率的高低。对图形元素的组织管理一般采用 2 种方法,一种是数组,另一种是链表。采用数组方法组织和管理图形元素比较简单,特别是 java 中提供了一些类似数组的 Vector 的

实现，封装了完成各种数组操作功能的函数。当图形中的图形元素之间的关系是离散的时候，用数组管理是高效的，但如果需要记录图形元素之间的拓扑关系，特别是拓扑关系可能有变化时，链表则显得更方便有效。如果采用变量化的设计思想，图形元素之间的关系可以用有向图来表示，在 java 中这两者可以说没有太大的区别。我们采用了 java 中 ArrayList 来管理图形元素。但我们对图形关键点的存储采用 Vector。总之这两种方式交替应用。得益于 java 强大的 Collection 机制。

(3) 图形数据的存储及交换。图形系统一般采用数据文件的形式存储图形数据，它应具有存储容量小、效率高以及与内部数据结构转换方便等特点，因此我们选择了 XML 格式。

2.3.2 模式的实现(Model)以及观察者设计模式的应用

ShapeModel 类扩展了 Observable，从而允许 Observer 注册监听器，以监听 ShapeModel 中的变化。ShapeModel 类包含有 ShapeObject 对象组成的一个 Collection 对象，以及增加删除形状实体的方法。观察者模式定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态上发生变化时，会通知都有观察者对象，使他们能够自动地更新自己。这里 ShapeModel 即为主对象。

2.3.3 图形控制器(Controller)

ShapeController 为基类，实现基本的方法，而每一个图形都有一个控制器，为 ShapeController 的子类。在这里 creatNewShape 方法使用了 java 中的反射机制(reflection mechanism)来创建新的 ShapeObject 子类实例。反射机制能够在 java 运行时确定关于类和对象的信息。能够动态的创建 ShapeObject 的任意子类。

2.3.4 工厂方法设计模式与单态设计模式的应用

当创建一个新的 ShapeObject 子类时都需要一个 ShapeController 子类的控制器。为了在增加新的 ShapeController 时不再需要修改现有的代码，这里采用了两种设计模式——工厂方法设计模式和单态设计模式。应用程序使用工厂设计模式时，用户能够在运行时选择合适的 ShapeController。正如其名字暗示的，利用工程方法来创建对象，工厂方法可以基于只有在运行时才能知道的标准来创建对象，这些标准可能是用户的输入，也可能是系统的属性等。用户选中特定的 ShapeObject 子类之在运行时才能知道，因此在编译时不能确定使用何种 ShapeController 来控制用户的输入。必须在 ShapeControllerFactory 类中使用工厂方法，从而为用户选定的 ShapeObject 构造合适的 ShapeController。

ShapeControllerFactory 类还使用了单态设计模式，用来控制其他对象获得 ShapeControllerFactory 类的实例。单态设计模式保证在一个应用程序中只有一个特定的对象实例存在。

2.3.5 语法检查设计

语法检查是命令绘制图形的主要部分，命令必须经过语法检查，确定其正确之后才能进行图形绘制。其主要功能就是检查命令行语法的合法性，并能进行简单的错误定位，例如是参数名错，是第几个参数错误等。其主要流程如图 2。

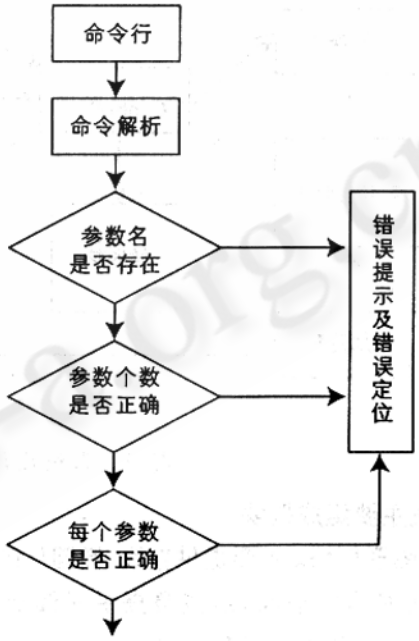

图 2 语法检查流程图

参数命令举例

命令名称	图形名称	X1	Y1	X2	Y2
Line	Line0	34	45	234	345

2.3.6 图形关联设计

图形关联比较复杂，考虑一下几种情况：

(1) 当一个图形删除时，若有图形与其关联，则其

两个图形之间的关联线也要同时删除，并在另一个图形中取消关联。

(2) 当一定关联的其中一个图形时，其关联线要跟着移动。

(3) 当删除关联关系时，删除关联线，同时关联的两个图形撤消关联关系。

(4) 一个图形可能有两个以上关联，此时要对每一个关联关系的具体情况进行保存。

(5) 当图形保存之后，其关联关系也要保存，在下一次打开图形时，其关联关系仍然存在。

关联表的参数格式：

关联名称，图形1名称，图形1控制点，图形2，图形2 控制点

gl0 rect0 3 oval0 4

即是矩形图形 rect0 的第三个控制点与椭圆图形 oval0 的第四个控制点关联，其关联名称为 gl0，关联名称是唯一的。它作为查找关联表的关键字。

另外在每一个图形对象中保存与其关联的名称，当删除或移动等操作时，查找图形对象是否有关联存在。然后根据其关联名查找关联表得到具体的参数。在根据其参数进行图形对象的操作。

2.3.7 自定义图形对象设计与组合模式的应用

用户自定义图形即将基本的图形组合成用户需要的图形形式。在此采用设计模式中的组合模式，它也是处理对象的树结构模式。由于一个复杂图形是由基本图形组合而成的，因此一个合成的图形应当由一个列表，存储对所有的基本图形对象的引用。组合图形的 draw( ) 方法在调用时，应当逐一调用所有列表上的基本图形对象的 draw( ) 方法。

2.3.8 转换器设计

转换器实际上就是一个控制器，它集成了鼠标和键盘的响应事件，以及对图形的操作如移动，缩放，删除，等一系列操作。并且它还起着将命令和鼠标作图转换为相应的命令执行。它是图形编辑器的中心，可以说是本次设计的重点。

主要方法的设计描述

方法	描述
shapeSelectOperation	选择图形对象，然后对图形对象进行操作。只有选择图形之后才能进行操作。被选中的图形对象应处于激活状态，和图形的高亮显示。
shapeDeleteOperation	图形对象的删除操作，只对当前选中的当前图形进行删除。删除时考虑关联情况，若存在关联则要将关联线一并删除
shapeMoveOperation	图形对象的移动操作，只对当前选中的当前图形进行移动，同样移动图形时也要考虑关联。
shapeNameChecked	命令作图时用于判断图形是否由重名的，若有则给出错误提示。
shapeZoomOperation	图形对象的缩放操作，只对当前选中的图形进行缩放，其缩放时同样也要考虑图形的关联。
shapeModifyOperation	图形对象的改变操作，包括上下左右，以及各个端点的改变。同缩放的区别是：改变只在某一个或两个方向上进行操作。主要是对8个关键点进行操作。在图形的改变时同样也要考虑关联。
setShapeNoActive	用于改变图形的状态，使其处于不激活状态。

2.3.9 文件存储(DrawingFileReaderWriter)

图形对象的保存采用 XML 格式，对每一个图形对象的属性，及其关联进行保存。在打开时可以再现图形对象，以及其关联的关系。保存打开文件中使用了一个文件过滤器。DrawingFileFilter 类是一个文件过滤器(FileFilter)，它使 JFileChooser 对话框只显示符合应用程序需要的一些文件，对 llc(保存文件的扩展名)文件的描述，并指定文件的扩展名。只有当给定的文件与 llc 文件的扩展名相匹配时，accept 方法才返回 true。

DrawingFileReaderWriter 类的设计，它实现的是文件的保存及读取，定义了从磁盘读取和向磁盘写入图形的方法，它用于图形的保存和加载。

参考文献

1 Harvery M. Deitel 等 Advanced Java2 Plantform How to Program[M] Prentice Hall 2003。

2 阎宏，Java 与模式[M]，北京 电子工业出版社，2002。

3 刘润东，UML 对象设计与编程[M]，希望电子出版社，2001。