密级：_____

浙江大学

硕 士 学 位 论 文

论文题目 基于声明式交互的数据可视化系统的设计与实现

作者姓名 张 也

指导教师 陈刚 教授

学科(专业) 计算机科学与技术

所在学院 计算机科学与技术学院

提交日期 2017.01.05

A Dissertation Submitted to Zhejiang University for the Degree of Master of Engineering

TITLE: Design and Implement of Data Visualization System Based on Declarative Interaction

Author: Ye Zhang

Supervisor: Gang Chen

Subject: Computer Science and Technology

College: Computer Science and Technology

Submitted Date: 2017.01.05

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 **浙江大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解 **浙江大学** 有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 **浙江大学** 可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

学位论文作者毕业后去向：

工作单位：　　电话：

通讯地址：　　邮编

摘要

随着信息时代的到来，越来越多的信息资源变得触手可及，人们在生产、生活当中也产生了海量的数据，因此信息在爆炸式增长。在数据分析的各种方法中，数据可视化是一种极其重要的策略。数据可视化将不可见或者难以直接显示的数据转化为可感知的图形、符号、颜色等视觉属性，以人眼更直观的方式展示数据及其结构关系，更有效地传递了信息，进而辅助人们进行决策。

基于上述观察，本文对数据可视化技术进行了深入的研究，提出并实现了数据可视化系统，以快速生成用户自定义样式的交互式图表。本文提出并解决了数据可视化技术中两大技术难题：一是系统如何将数据转换为可交互的图表来呈现给用户，并且用户如何达到个性化定制图表样式的目的。二是用户如何描述期望的交互操作，以及系统如何将交互操作产生结果反馈给用户。

针对数据的可视化呈现难题，本文基于图形语法描述了图表结构，构建生成了图表视图。同时，本文提出了主题模型以满足用户对图表的高度定制化需求。针对用户交互的难题，本文提出了声明式交互语法，此语法使用类似自然语言的描述方式以声明用户的交互操作。在交互技术的实现方面，本文提出了事件流模型，对发生在图表中的各种事件进行建模。同时，本文实现了事件处理模型以解析用户的交互操作，将交互结果反馈给用户。

本文实现的数据可视化系统，作为网易敏捷商业智能平台的子系统，在图表的可交互性和个性化定制这两方面得到了充分地检验，为探索数据的规律与意义奠定了良好的基础。

关键词： 数据可视化，声明式交互，事件流模型，事件处理

Abstract

With the advent of the information age, more and more information resources are in your reach. People produce a mass of data in production and life, resulting in an explosion of data growth. Data analysis is a very important task. Data visualization is an essential method in data analysis. Data visualization transforms data that is not visible or difficult to display directly into perceivable visual attributes such as graphics, symbols, and colors. By data visualization, we can see data and its structure in a more intuitive way. And data visualization can assist our decision-making.

Based on the above observations, this paper makes a deep research on the data visualization technology, proposes and realizes the data visualization system to rapidly generate the interactive graph which user expect. This paper presents and solves two technical problems in data visualization: one is how to convert the data into an interactive graph, and enables user to achieve a highly personalized customization of the graph in a simple way. The other is how to describe the desired interaction in the system, and how to implement feedback of user interaction.

In order to solve the first problem, this paper describes the structure of graph based on the grammar of graph, and generates graph view. In addition, this paper proposes a theme model to meet the user highly customized requirements and provides a friendly user interface. To the problem of user interaction, this paper proposes a declarative interactive strategy to declare user interaction using a description that is similar to natural language. To realize interaction, this paper proposes an event flow model to model the various events that occur in the diagram, and implements the event processing model to analyze user interaction. Ultimately, the interactive results will be fed back to the user.

As a subsystem of NetEase Agile Visualization Business Intelligence Platform, the data visualization system implemented in this paper has been fully tested for its interoperability and personalization, and lays a good foundation in exploring the rule and meaning of data.

Keywords： Data Visualization, Declarative Interaction, Event Flow Model, Event Handling

目录

图目录

表目录

第1章 绪论

1.1 课题背景

随着信息时代的到来，越来越多的信息资源变的触手可及，人们在生产、生活当中产生的海量数据导致了数据的爆炸性增长。例如在各种企业经营中，销售数据、财务数据等会被保存下来；在我们的日常生活中，银行卡消费数据、出行数据、通话记录以及流量使用情况都会被记录下来。随着数据量的不断增大，大数据背后蕴藏了很多有价值的信息。

在大数据时代，对数据合理、可信、高效的分析成为一件至关重要的任务。越来越多的企业通过对数据的分析，推动合理决策，从而最大化自身利益，间接促进社会发展。在数据分析的各种方式中，数据可视化是一种极其重要的方法。在计算机学科的分类中，利用人眼的感知能力对数据进行交互的可视化表达以增强认知的技术，称为可视化（Visualization）[1]。

人眼是一个高带宽的视觉信号并行处理器，具有很强的模式识别能力，对可视符号的感知速度远比对数字或者文本快。数据可视化将不可见或者难以直接显示的数据转化为可感知的图形、符号、颜色、纹理等，就是充分利用了人们对可视模式快速识别的自然能力的过程[2]，将枯燥的数据变成了视觉上直观、形象的图表。数据可视化通过直观的方式展示数据及其结构关系，极大的提升了人们分析和理解数据的速度，帮助人们更有效地对海量数据进行探索，发现规律，做出决策。

数据可视化是一门交叉学科，主要融合了数据的统计分析与图形设计两大领域的知识[3]，其流程中的核心要素包括三个方面：

（1） 数据表示与变换

数据可视化的基础是数据表示和变换[4]。为了有效的可视化、分析和记录，输入数据必须从原始状态变换到一种便于计算机处理的结构化表示形式。通常这

种结构存在于数据本身，我们需要研究有效的数据提炼或简化方法以最大程度地保持信息和知识的内涵以及相应的上下文。有效表示海量数据的主要挑战在于采用具有可伸缩性和扩展性的方法，以便忠实地保持数据的特性和内容[4]。此外，我们需要将不同类型、不同来源的信息合成为一个统一的表示，以便数据分析人员能及时聚焦于数据的本质。

（2） 数据的可视化呈现

为了以一种直观、易理解、易操纵的方式将数据呈现给用户，我们需要将数据转换为可视化图表。同一个数据集可能有多种的视觉呈现形式，因此，数据的可视化呈现的核心内容是在巨大的呈现空间中选择最合适的呈现方式，并以此编码数据[5]。

（3） 用户交互

对数据进行可视化和分析的目的是解决目标任务。有些任务可以明确定义，有些任务则更广泛或者一般化。通用的目标任务可以分为三类：生成假设、验证假设和视觉呈现。数据可视化可以用于从数据中探索新的假设，也可以验证相关假设与数据是否吻合，还可以帮助数据专家向公众展示数据中的信息[6]。交互是通过可视的手段辅助分析决策的直接推动力[7]。

作为数据可视化引擎的子系统，NEV（Nice Easy Visualization）数据可视化系统主要涉及到数据的可视化呈现和用户交互两方面的工作。

1.2 研究现状

下面主要从数据的可视化呈现以及用户交互两个方面来阐述研究现状。

1.2.1 数据的可视化呈现

可视化要解决的问题的本质是如何将数据用某种可以“看得见”的方式展现出来，而这种可以“看得见”的方式通常是借助图形实现。数据的可视化呈现归根结底是对可视化图形的建模。因此，对图形的类型和结构的描述与建模，是可视化系统中必不可少的一部分。

随着网页技术的发展，HTML5 中的 Canvas 图形接口变得越来越流行。Canvas 是一个底层图形接口，它负责绘制二维空间中的图元，例如在某个位置画一个点、一根线或者一个面等等。这让图形具有了一定的语义，但这种方法也只将一个图元简单堆砌到另外一个图元上，并且这些底层图形接口并不友好，用户学习成本较高、体验性较差，影响了开发效率。

Prefuse[8]是一套基于 Java 二维图形层的可视化工具集。它为用户供了一套可以重用的可视化组件，支持从结构化与非结构化数据中构建交互式的可视化应用。但是 prefuse 没有明确的图形结构的概念，它只是供了一些特定的可视化图形与布局算法，使用户可以从数据中创建出可视化图形。这些可视化图形是由一组参数化的可视物体组成，用户需要编写一定量命令式的代码来定制参数。

目前网络上与可视化有关的库层出不穷，特别是基于 HTML5 技术与 JavaScript 语言的可视化库，其中比较有代表性的可视化库是 HighCharts 和 Echarts。

HighCharts 是国外一款可视化商业产品，使用纯 JavaScript 编写的一个图表库。在 HighCharts 可视化库中，数据没有明确的意义，分散各处，格式混乱，不同格式的数据会让用户写出非常混乱的代码，而且不利于维护。另外，HighCharts 中图形结构定义过于简单，无法支持复杂的图表，用户也不能创造出自定义类型的图表。并且 HighCharts 将可视化图形中的各种成分混杂在一起，包括数据与图形结构的定义，图形结构与图形样式的定义，不利于用户的编写与维护。

ECharts 是由百度公司开发的商业产品图表库，提供了商业产品常用的图表，底层基于 ZRender（一个轻量级的 Canvas 类库）创建了坐标系，图例，提示框，工具箱等基础组件，并在这些基础组件上构建了折线图，柱状图，散点图等基本图表。由于 ECharts 的设计基本继承了 HighCharts，因此 ECharts 也有上文中 HighCharts 的各种缺点，此处不再赘述。

还有一种非常灵活的方法描述图形的结构，即图形语法的思想。图形语法（Grammar of Graphics）[9]是由 Wilkinson 提出的一种底层统计图形生成语言，可以用于构造不同类型的统计图形。Wilkinson 通过语法生成复杂的图形，以自底

向上的方式组织最基本的元素以形成更高级的元素。作为图形的一种模型，图形语法规定了有意义图形的生成规则，只有按照图形语法生成的图形，才是有意义的。对用户来说，图形语法供了一种描述图形、生成图形的新方式，这种方式更加灵活、更加有表达力，可以让用户“描述”出他真正想要的图形。另外，相比于底层命令式的编程，图形语法的声明式描述方式简单明了，降低了用户的学习成本，提高了用户制作可视化图形的效率。

可视化渲染语法 Vega[10]是基于图形语法的理论实现的。Vega 是一种基于 D3 的、可创建、可保存、可分享的交互式可视化设计。当用户按照 Vega 要求的 JSON 格式告诉它需要画什么样的图，Vega 就会自动进行图表渲染。Vega 有非常高的自由度，只要有想法，就能画出各种各样的图表。高自由度意味着高复杂性，对于某些比较复杂的图形，Vega 的代码相当冗长。

Vega 是在 D3 的基础上封装转换而来，下面重点研究 D3。D3[11]是 Data-Driven Documents 的缩写，它是是一套面向 Web 的数据可视化的 JavaScript 库，基于 HTML，SVG 和 CSS 构建，前身是 Protovis[12]。D3 以轻量级的浏览器端应用为目标，具有良好的可移植性。D3 将任何数据绑定到一个 DOM（文档对象模型）对象，并提供了基于数据的操作，这既避免了面向不同的类型和任务设计专有的可视化表达的负担，又提供了设计灵活性，同时发挥了 CSS3，HTML5 和 SVG 等 Web 标准的优势。但是，D3 对于用户的要求非常高，用户会有一个比较陡峭的学习曲线，而且不利于用户专注于数据的可视化设计。

1.2.2 用户交互

实现用户与图表的交互，有助于增强用户对数据的控制与探索[13]。通过交互，用户可以在已有的数据集中选择子集或者改变观察数据的角度，来关注自身感兴趣的内容，获取最大信息量。下面重点介绍国内外可视化产品对用户交互的支持。

在 1.2.1 小节列举出的非图形语法可视化图表库中，HighCharts 和 Echarts 提供了预先定义好的交互（如框选、缩放等），使图表库支持某些交互能力。这些预先定义好的交互模板数量有限，不能满足用户多种多样的交互需求，并且限制

了用户表达自定义的交互策略。为了能够实现用户自定义的交互，用户需要完成处理事件的回调函数[14]，这些命令式的回调函数暴露了各种事件在执行过程中的细节，使得用户需要手动维护事件的状态以及事件之间的协作交错，因此描述用户交互是一件复杂、易出错的任务，这种情况被称为 “回调地狱”[15]。

G2 是由阿里巴巴团队开发的一个由纯 JavaScript 编写、功能强大的语义化图表生成工具[16]。G2 提供了各种事件支持，以响应用户的交互操作，方便用户扩展交互。开发者需要监听这些事件，然后通过回调函数做出相应的处理。通过回调函数的方式来拓展用户交互，对用户的要求高，同时会将事件处理细节暴露给用户，用户需要协调各种状态的变更，极易容易出错，用户体验很差。

D3 中描述用户交互是通过回调函数的方式完成，用户需要显式指定具体操作的回调函数的细节。这样用户不仅需要清除浏览器中事件处理机制，还需要一定的编程经验来实现各种状态的控制，这些琐碎的细节会让用户深感疲惫。

Vega 不仅仅支持了图表结构的描述，还能够表达用户的交互。但是通过 Vega 表达用户交互非常繁琐，并且 Vega 中描述交互的语义是专用的接口定义，晦涩难懂，不方便用户理解与使用。

1.3 本文工作与贡献

本文的主要工作与贡献在数据到图形的可视化呈现和用户交互两个方面，下面分别展开。

首先，基于图形语法理论，本文抽象出了一种图表通用的表达结构。利用该表达结构，我们统一了图表的描述，进而生成并渲染出了最终的图表视图。在数据到图形的可视化呈现过程中，本文提出了主题模型来满足用户对图表的高度定制化需求，完善数据到图形的显示结果。通过图形渲染，NEV 数据可视化系统有效地呈现了数据中的几何拓扑和形状特征。

同时，针对当前国内外图表库交互描述欠缺的状况，本文提出了一种声明式交互描述语法，该语法基于图形语法的图表描述框架实现，有效地支持用户描述自定义交互操作。本文采用事件流模型实现事件的转换和组合，然后通过事件处

理模型实现用户的各种交互操作，从而完成了用户与图表的交互，达到了探索图表背后数据之间的关联、挖掘数据有价值的信息的目的。

1.4 篇章结构

本文的正文部分共六章，其组织结构如下：

第一章节介绍了课题的研究背景及主要内容，简单阐述了若干技术的国内外研究现状，同时总结本文的工作与贡献。

第二章节对相关技术进行综述，主要对数据可视化技术、编程范式、事件流技术以及交互技术四个方面进行论述。

第三章节详述 NEV 数据可视化系统的设计，主要从架构设计和模块设计两个个方面加以阐述，并且详细地描述了系统的图表样式设计。

第四章节首先描述了声明式交互的设计，事件流的转换与组合；然后描述了基于事件流的交互的实现，具体包括交互流程的说明、事件仲裁器、事件分发器以及用户行为模型的实现。

第五章节通过实例展示，说明了基于事件流模型的声明式交互的设计与实现的可行性，验证了主题模型的可用性。

第六章节是对全文的总结，并对未来的工作进行了展望。

1.5 本章小结

本章从数据可视化的研究和应用背景出发，明确指出了本课题的研究意义，结合国内外相关领域的研究内容和技术路线，着重阐述数据的可视化呈现以及用户交互的研究现状。然后详述论文的主要工作和贡献，阐明了课题的研究价值和实用意义，最后介绍了本文的篇章结构。

第2章 相关技术研究

本章介绍数据可视化和声明式交互的相关技术。数据可视化技术方面，主要包括数据到可视化的直接映射、图形语法以及数据到图表的构建。本文主要工作是提出了声明式交互的表达方案，来描述用户的交互操作，所以声明式编程范式至关重要。在抽象用户交互行为时，函数式响应型编程思想与事件流模型将交互事件建模为可组合的数据流。因此声明式交互的相关技术主要从数据可视化技术、编程范式、事件流模型以及交互技术四个方面展开。

2.1 数据可视化技术

2.1.1 数据到可视化的直观映射

数据到可视化的直观映射是可视化编码技术的直接体现，该技术的目标是将数据映射成为可视化元素（标记和视觉通道）[17]。可视化编码由两方面组成：一方面是（图形元素）标记（mark），另一方面是用于决定标记的视觉特征的视觉通道。数据通常包括了属性和值，标记是数据的属性到可视化中图形元素的映射；视觉通道是数据的值到标记的视觉属性的映射，用来展现数据属性的定量信息。通过数据到可视化元素的直观映射，可以完整地对数据信息进行可视化表达。

如图 2.1 所示，标记通常是一些几何图形元素，如点，线，面，体等。标记具有分类性质，不同的标记可用于编码数据属性。视觉通道则用于控制标记的展现特征，从定量的角度描述标记在可视化图形中呈现的状态，包括标记形状、尺寸、位置、角度、颜色等。视觉通道不仅具有分类性质，也具有定量性质，因此一个视觉通道可以编码不同的数据属性（如形状），也可以编码不同的值（如长度）。视觉通道与标记的空间维度之间是相互独立的，视觉通道在控制标记的视觉特征的同时，也蕴含着对数据数值信息的编码。

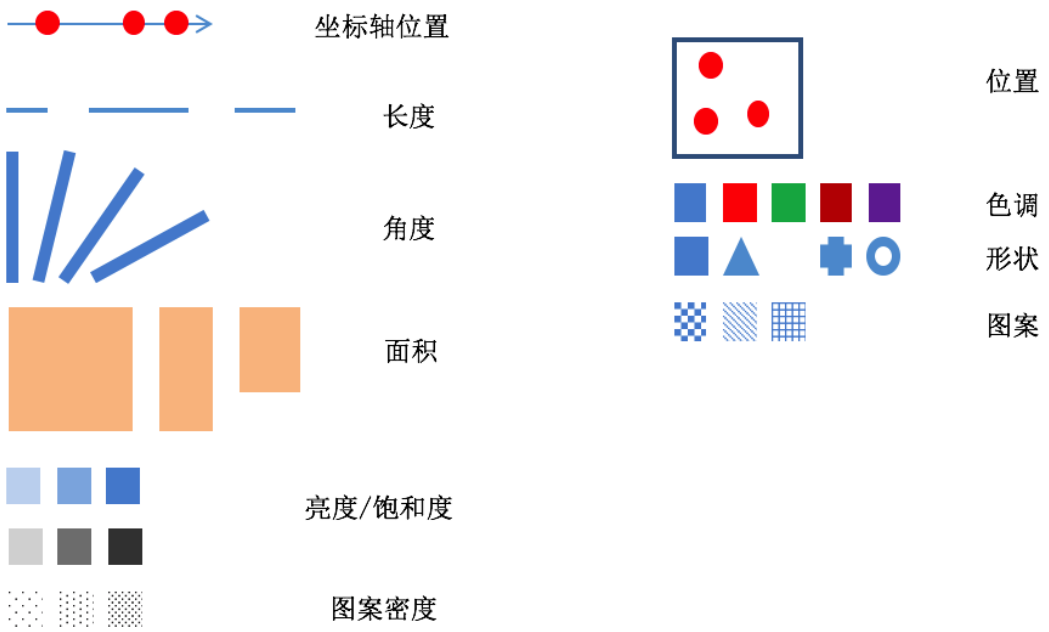

图 2.1 视觉通道和可视化的标记

2.1.2 图形语法

图形的构造过程分为三个阶段：规范定义、组装和显示。其中，规范定义是整个语法的基础，描述了不同图形对象间的转变和最终显示映射。

整个图形语法[9]规范由 7 个部分组成，详见表 2.1。

表 2.1 图形语法组成表

数据	从数据集中生成变量的数据操作
转换	数据变量间的转换
框架	变量空间，包括变量间的操作
标度	标度转换
坐标	坐标系统
图形	图形及其美学属性
参考	用于图形对象间的对齐、分类和比较等

数据和转换定义在数据空间；框架、标度和坐标定义了底层的图形几何和数据的空间位置；图形定义了不同的图形对象。

Wilkinson 提出了两个重要的可视化概念。

（1） 数据和它们的视觉表达应该被分开。本质上，图形语法中的规范定义了从数据点到视觉属性的映射。

（2） 可应用不同的算子构造数据变量的可视化。其主要思想是，可采用融合（+）、叉乘（*）、嵌套（/）等算子从各类数据变量出发定义复杂的图形空间，并通过缩放映射到将要显示的视图。这些算子支持不同维度数据的复杂操作。

Wilkinson 的方法刻画了可视化的数学表达和图形属性之间的区别。基于这套理论开发的面向对象的软件，例如 ggplot2[18]已经证明了利用图形语法生成数据可视化图形的可行性。

2.1.3 数据到图表的构建

如何将数据映射成为图形？我们以制作饼图为例，来阐述这个过程。图 2.2 显示了制作饼图简单的数据流模型。从数据源中流出的数据经过饼图构造器的“加工”处理，创建一个饼图，然后经过渲染器展示出来。

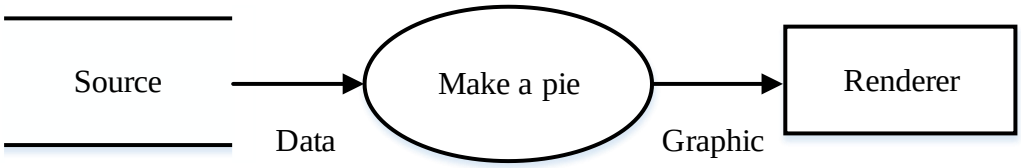

图 2.2 绘制饼图流程图

这个简洁的流程忽略了绘制图表的许多细节，例如数据如何组织？饼图的扇半如何着色？如何绘制柱状图、散点图等其他的图表？针对以上各类问题，图形语法给出了具体的解决方案。

图形语法的设计目标是通过完成灵活可用的系统，尽可能地通过简单的方式创建丰富多样的图表。图 2.3 细化了生成图表的流程，阐述了基于图形语法完成的系统是如何从数据一步步构建出图表的。在图形语法中，先要从数据源中创建

变量，根据表代数进行变量间的运算，进行可视化比例尺的计算，映射数据到可视化标记的几何图形上，通过坐标系映射到空间中，进行视觉属性上的处理，最终渲染可视化图形，得到可视化图形结果。

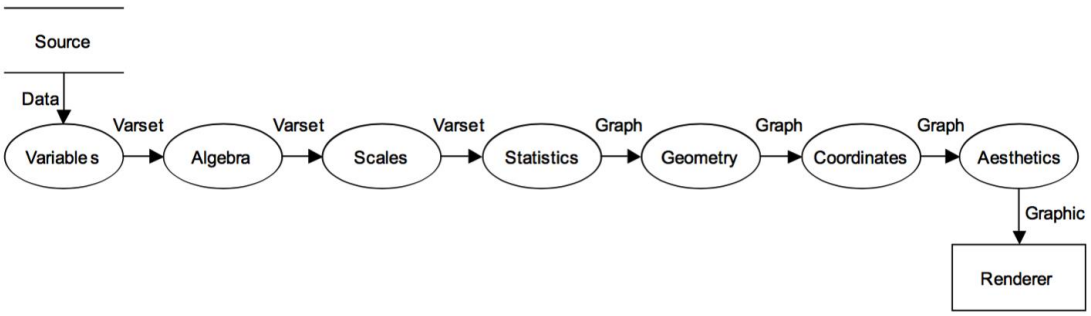

图 2.3 图形语法中的绘制图表流程图

2.2 编程范式

2.2.1 函数式响应型编程

在计算机中，响应式编程（Reactive Programming）是一种面向数据流和变化传播的编程范式[19]。这意味着可以在编程语言中很方便地表达静态或动态的数据流，而相关的计算模型会自动将变化的值通过数据流进行传播。利用函数式编程的思想和方法（函数、高阶函数）来支持响应式编程就是函数式响应型编程（Functional Reactive Programming）[20][21]。

函数式响应型编程将可变的量建模成随时间变化的连续的数据流[22]。函数式响应型编程有各种变体，本文关注的技术是事件驱动的函数式响应型编程（Event-Driven Functional Reactive Programming ）[32]，E-FRP 编程模型的关注点在于离散的事件，其将离散的时间建模成流（streams）与信号（signals）；streams 是无数的事件序列，signals 是最近发生的事件流，signals 类似于变量，可以进行组合来表达各种不同的交互行为[32]。在运行过程中，当产生新的事件时，E-FRP 可以生成相应的事件流，并且传递相对应依赖的信号，同时会自动地对依赖的信号进行评估。Flapjax 与 EML 已经将 E-FRP 思想应用到可交互式 web 应用中。

2.2.2 声明式编程

声明式编程是一种编程范型，与命令式编程相对立。它描述目标的性质，让电脑明白目标，而非流程。声明式编程不用告诉电脑问题领域，避免随之而来的副作用。而命令式编程则需要用算法来明确的指出每一步该怎么做。

目前，声明式编程已经广泛应用于数据可视化中，来满足用户对图表的定制化需求。用户通过声明式图形语法描述数据到图表中的基本图元元素的映射，可视化图表库在运行过程中决定数据处理以及最终的图形渲染方式。因此，用户的关注点集中在想要表达的可视化图形上，而不是如何来实现这些可视化图形。

Heer 和 Bostock 提出了一种声明式表达语法来描述用户的交互操作[23]，使得用户不必受到“回调地狱”的困扰。

2.3 事件流模型

2.3.1 事件流建模

对事件建模成事件流，如图 2.4 所示：

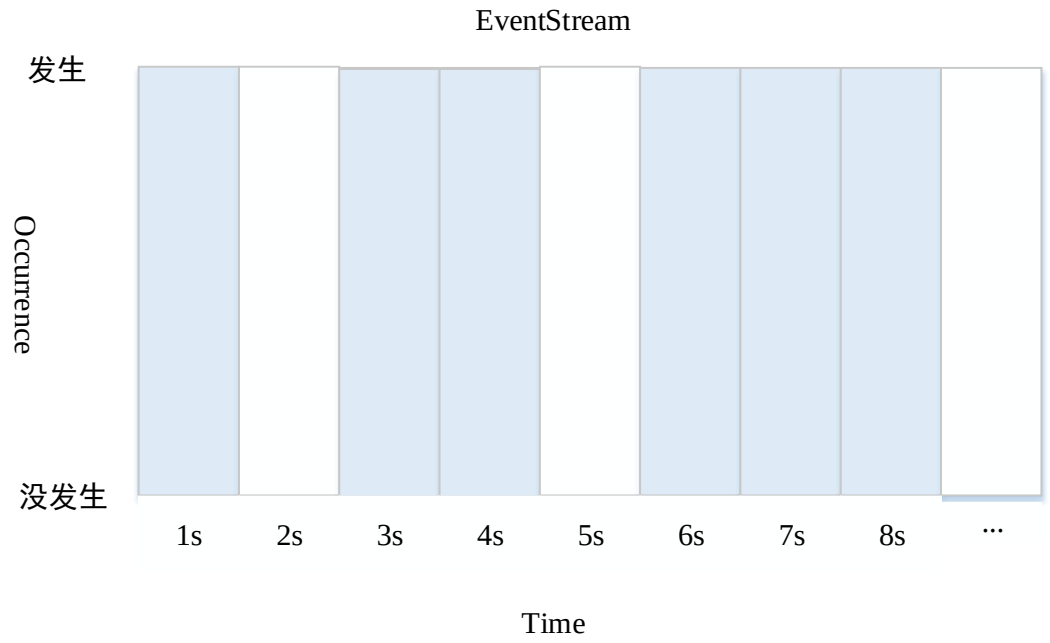

图 2.4 事件随时间变化图

我们与图表发生交互的过程中，发生鼠标点击事件，1 秒时我们点击一次，然后 3 秒时再点击一次，随着时间推移进行下去，那么我们与图表发生交互最根本的自变量是时间，变化的环境和各种事件（比如键盘、鼠标、网络）归根结底都是由时间的变化引起的，时间是本质的特征。因此点击过程中产生的鼠标事件对象，可以看成是分布在时间轴上的数据点。这些对外部环境具有反应能力、随时间变化的数据点被抽象称为事件流（Event Stream）[24]。从图 2.4 可以看出，事件流被看做时间轴上无限长的数据流，时间轴里面流淌的数据代表着一个个事件，它们在时间轴上是离散的。从函数编程的角度看待事件流，事件流可以看成这样一个函数，详见图 2.5，输入某个时刻，如果在这个时刻有事件发生就把相应的事件数据返回，否则就返回一个特殊的 nothing 表示没有任何事件发生。

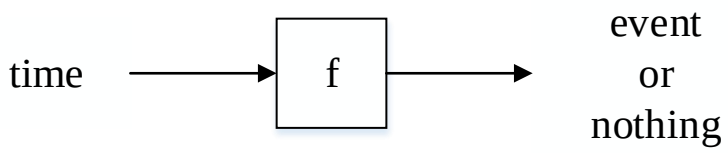

图 2.5 函数化事件示意图

2.3.2 事件流的变换与组合

如图 2.6 所示，当用户与图表发生交互后，将各种独立的事件看成事件流，传递事件流的通道称为事件流管道（pipe），在管道中的事件以流的方式进行传递，也可以通过管道上的端口（port）去提取（pull）管道中的事件，不同端口之间互不影响[25]。

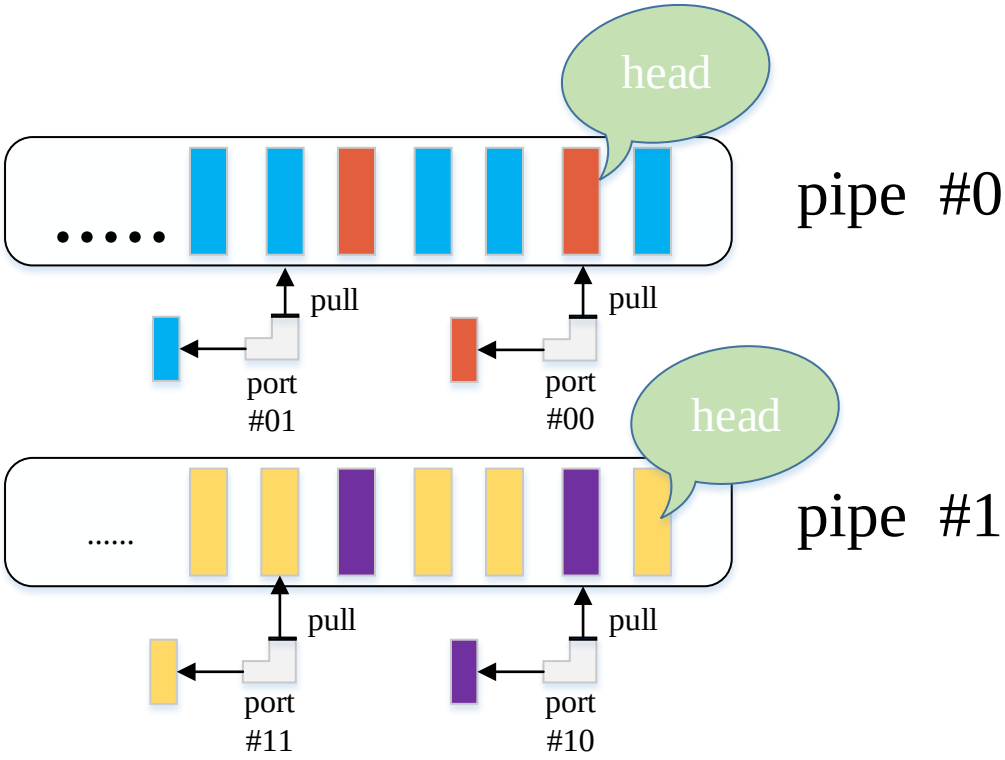

图 2.6 事件流模型图

在这个事件流模型当中，事件流被看作是第一类的值（first-class value），它可以进行各种各样的转换与组合，得到的还是一个新的事件流，而新生成的事件流又可以通过某种转换和组合方法进行处理从而得到更加复杂的事件流。在转换、组合过程中，原来的事件流并不会被改变，这种特性称为不可变性（immutability）。事件流能够进行的转换与组合操作包括：过滤操作、映射操作、与、或操作等等。下面着重介绍这些操作的细节：

（1） 过滤操作

图 2.7 概括了过滤操作。函数 filterE（sourceEs, pred）通过谓词函数 pred 对事件流 sourceEs 进行过滤，生成一个新的事件流，即所有出现在 sourceEs 并且被 pred 判断为 true 的事件都会出现在新生成的事件流中。

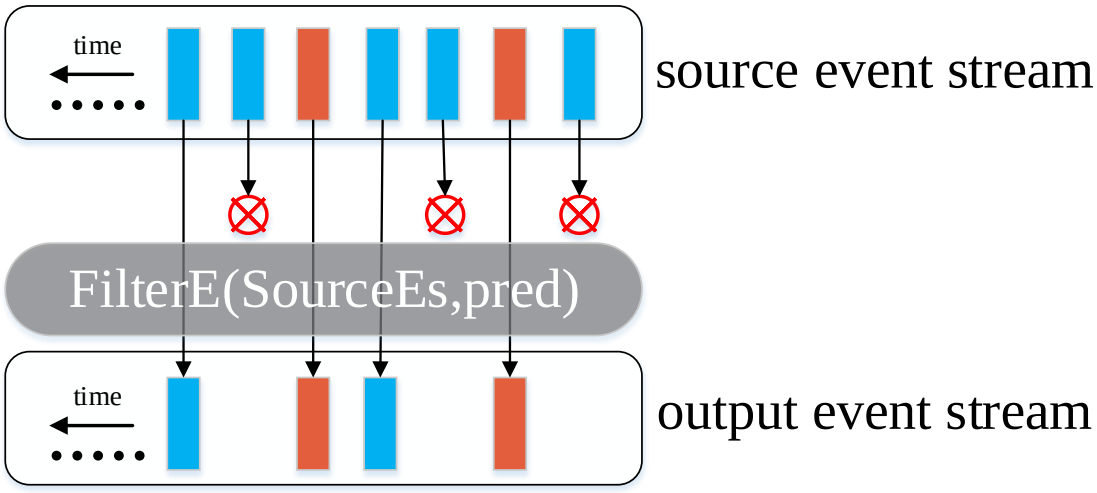

图 2.7 事件流过滤操作图

（2） 映射操作

图 2.8 概括了映射操作。函数 mapE（f, sourceEs）会用转换函数 f 对事件流 sourceEs 进行转换，生成一个新的事件流，即所有出现在 sourceEs 中的事件都会被转换函数 f 转换成新的事件并出现在新生成的事件流中。

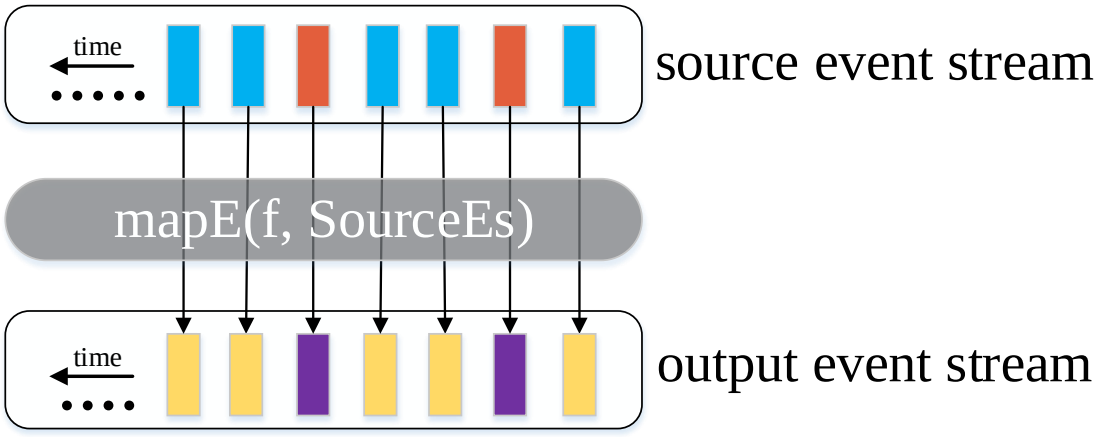

图 2.8 事件流映射操作图

（3） 合并操作

图 2.9 概括了合并操作。函数 mergeE（es1, es2）会把两个事件流 es1 和 es2 合并成一个新的事件流，所有出现在 es1 或者 es2 中的事件都会出现在新生成的事件流中。

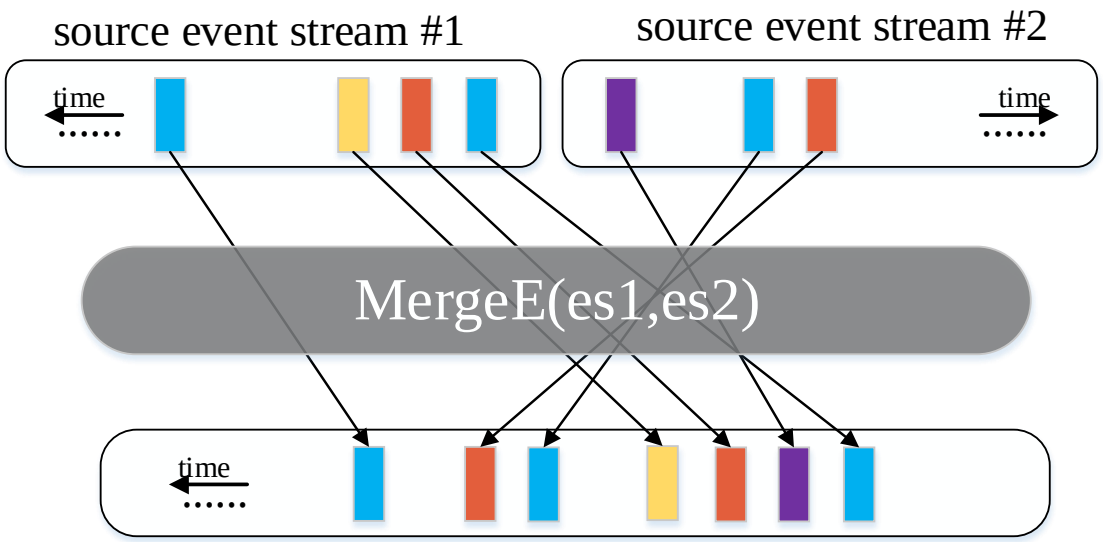

图 2.9 事件流合并操作图

（4） 与操作

图 2.10 概括了与操作。通过与函数选择出多个事件流之间同时发生的事件，生成一个新的事件流，相当于事件流间的交集。

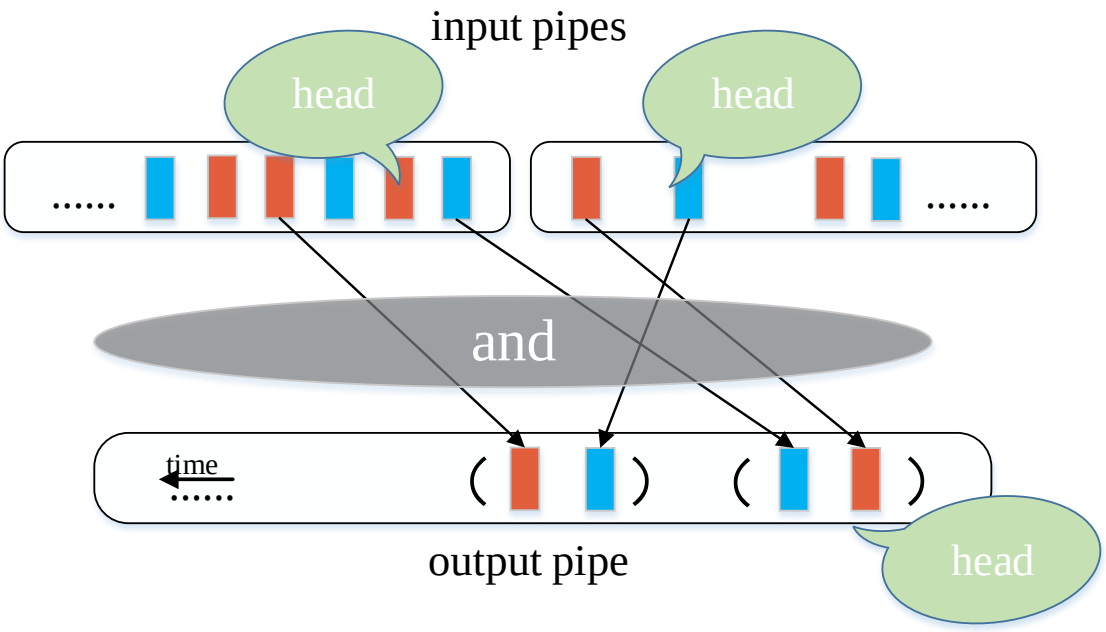

图 2.10 事件流与操作图

（5） 或操作

图 2.11 概括了或操作。通过或函数生成一个新的事件流，这个新生成的事件流包含了输入事件流中所有的事件，相当于事件流间的并集。

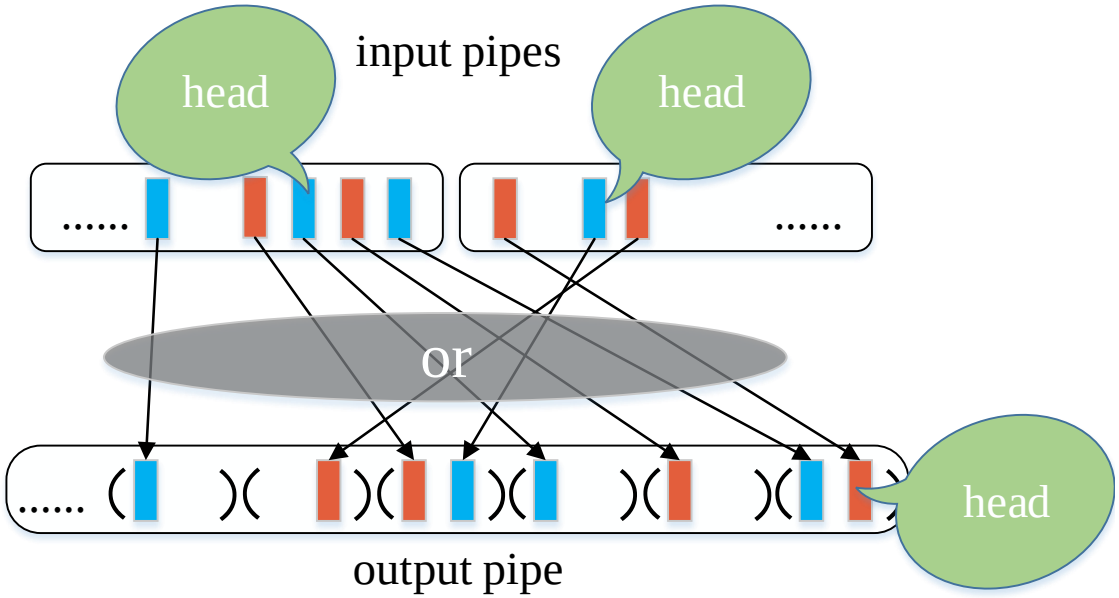

图 2.11 事件流或操作图

2.4 交互技术

交互是用户通过与可视化工具之间的对话和互动来操纵与理解数据的过程[13]。具体而言，交互在如下两个方面让数据可视化更有效。首先，有限的屏幕尺寸不足以显示海量的数据，交互可以有效缓解有限的可视化空间和数据过载二者之间的矛盾，另外常用的二维显示平面也对复杂数据的可视化提出了挑战，例如高维度的数据。交互可以帮助拓展可视化中信息表达的空间，从而解决有限的空间与数据量之间的差距。其次，交互能够让用户更自由地对数据进行探索，从而理解和分析数据。特别是对可视化工具来说，其目的不是向用户传递定制好的知识，而是提供工具和平台来帮助用户探索数据，得到结论。在这样的工具中，用户与图表的互动是不可缺少的。

交互的分类方法有很多，有各自的依据和适用情况，并不存在一个可以放之四海而皆准的分类。针对不同的交互任务，将交互技术分为 7 个大类[26]：

（1） 选择：标出感兴趣的数据对象。

（2） 抽象/具象：展示概览或者更多细节。

（3） 关联：展示相关数据。

（4） 导航：展示不一样的信息。

（5） 过滤：根据条件展示部分数据。

（6） 重配：展示一个不同的可视化配置。

（7） 编码：展示一个不同的视觉编码。

选择交互技术使用户对感兴趣的部分进行标记，即用户可以选择数据的一部分来进行某些变换，如高亮、置灰、显示或者隐藏标签等。并且在选择之后需要展示数据的提示性信息，陈列在视图上，同时取消选择交互后，这些提示信息也要随之消失[27]。

抽象/具象交互技术可以为用户提供不同细节等级的信息，用户可以通过交互控制显示更多或更少的数据细节，来达到浏览各个层次级别信息的目的。在实际应用中，抽象/具象交互技术往往体现为概览 + 细节的交互模式。概览 + 细节（overview + details）的基本思想是在资源有限的条件下同时显示概览和细节。概览指不需要任何平移或者滚动，在一个视图上集中显示所有的对象。概览 + 细节的用户交互模式指显示全局概览，并将细节部分在相邻的视图或者本视图上进行展示，其好处在于非常符合用户探索数据的行为模式。概览提供一个整体的印象，使得用户对数据的结构等全局信息有大体的判断。这个阶段往往出现在数据探索的开始阶段，随后可以引导用户进行深挖，以获取更多的细节。这个设计理念引用广泛，例如 Goggle Maps，它能展现概览和某一个层次的细节。

关联（connection）技术往往被用于高亮显示数据对象间的联系，或者显示与特定数据对象有关的隐藏对象，这在单视图和多视图应用中都有所体现，多视图尤甚。多视图可以对同相同数据在不同的视图中采用不同的可视化表达，也可以对不同但相关的数据采用相同的可视化表达，这样的好处是用户可以同时观察数据的不同属性，也可以在不同的角度和不同的显示方式下观察数据。然而用户首先必须清楚数据在各个视图中的具体位置，这就需要一种标识相关联的视图对

象的技术。链接（linking）[28]和刷动（brushing）[29]应运而生：当用户在一个可视化视图上框选了一些对象后，其余的视图上都能显现相应的关联结果。

导航（navigation）是可视化系统中最常见的交互手段之一，由于人眼可以观察到的区域以及屏幕空间都非常有限，当可视化的数据空间更大时，通常只能显示从选定视点出发可见的局部数据，并通过改变视点的位置观察其他部分的数据。这种操作可以想象成在一个空间中的某个点放置一个指向特定方向的虚拟相机。相机所捕捉的图像是当前的可视化视图，当相机的位置或者指向改变时，它所捕捉到的图像自然也发生了变化[30]。在信息可视化领域，导航被扩展到更为抽象的数据空间中，如树、图、网络、图表等不包含明确空间信息的数据，在这些抽象的空间中移动视点同样能有效进行交互概览。缩放（zooming）、平移（panning）和旋转（rotating）是导航中三个最基本的动作。

过滤操作可以向用户展示数据集中用户所关心的数据。用户可以设定一定的约束条和件，将不符合要求的数据项隐藏，从而专注于部分关键数据的分析和处理。通过过滤交互，结果数据被高效实时地返回给用户，这样既改善了过滤手段，又提升了过滤查询的效果。

重配交互旨在为用户提供观察数据的不同角度。常见的方式有重组视图、重排列等，其中一种重排列思想的交互技术是灰尘与磁铁方法[31]。

编码交互技术，是交互式地改变数据元素的可视化编码，如改变颜色编码、大小、方向、形状等，或者使用不同的表达方案以改变视觉外观，来展示数据的各个侧面，可以直接影响用户对数据的认知。

2.5 本章小结

本章节重点介绍了 NEV 数据可视化系统中基于事件流模型的声明式交互的相关理论与技术，包括数据可视化技术、事件流模型、声明式编程范式、事件驱动的函数式响应型编程技术和交互技术。在对声明式编程范式的介绍中，本章节阐述了这种编程范式的优势，以及与声明式交互的优势。然后，本章节阐述了事件流模型的建模思想，并且列举了事件流的变换和组合方案。事件驱动的函数式

响应型编程技术将会作为实现事件流模型的基础。最后本章节总结了常用的交互技术。

第3章 系统设计

本章主要介绍 NEV 的系统设计。本章首先描述系统的架构，从整体上把握系统设计；然后介绍系统各个模块的设计；最后本章介绍主题模型，进一步明确在 NEV 内部如何构建出用户期望的个性化图表。

3.1 架构设计

NEV 数据可视化系统是在 Canvas 语义的基础上构建的，基于图形语法来生成交互式图表。NEV 主要分为四个部分，分别是：用户接口层、图表描述层、图表生成层和图表渲染层，如图 3.1 所示。

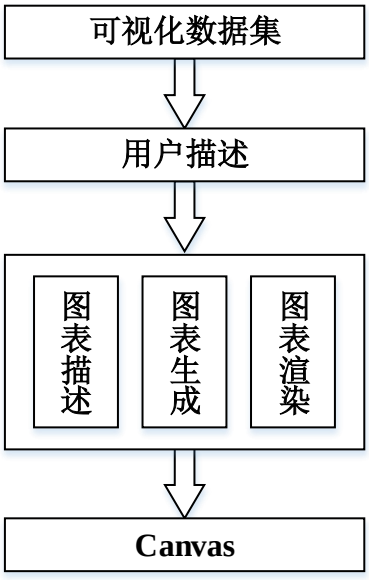

图 3.1 系统架构设计图

其中，用户层接口通过声明式的方式描述用户期待的图表结构、图表样以及交互操作，上述元素和可视化的数据源组成了图表描述层的输入。

图表描述层解析数据到可视化编码的映射规则，编译解释生成描述图表结构的图形语言 Babel。Babel 语言用来描述图表的结构、样式和交互。图表的结构包

括标记（mark）、坐标系（coordinate）、比例尺（scale）、辅助图表显示的坐标轴（axis）、图例（legend）。Babel 数据作为图表生成层的输入。

图表生成层将 Babel 数据解析成 NEV 系统内部的数据结构（innerData）来创建图表的场景，实例化标记精灵（Actor），计算精灵的布局数据，整合图表的样式配置，解析定义的交互操作。innerData 数据作为图表渲染层输入。

图表渲染层基于 Canvas 将交互式的图表渲染到终端设备上，终端设备是指各大主流浏览器。

3.2 模块设计

3.2.1 图表描述模块

图表描述模块将用户层的图表描述转换成图表描述层的 Babel 图形语言。Babel 语言的设计来源于 Wilkinson 的图形语法[9]思想，Babel 语言描述图表的结构、样式和交互，专注于数据到图形的映射，降低图形与数据的耦合性。

在 Babel 图形语言中，图表的结构是由数据在坐标系的影响下，映射成标记图形组成的，具体组成细节如图 3.2 所示。其中，红色箭头表示图表结构的必要组成部分，黑色实线箭头表示图表的辅助组成部分，黑色虚线箭头表示图表组成部分之间存在某种关系。

从图 3.2 看出，图表结构包含四部分：需要可视化展示的数据、比例尺、坐标系和标记图形。一个图表理解为标记图形在某种坐标系下的可视化呈现，具体的标记需要嵌入到某种坐标系中才会与数据产生联系。在数据与标记图元的映射关系中，比例尺决定了标记的空间位置、视觉属性与数据的对应关系。例如在柱状图中，连续型度量可以映射为柱形颜色的深浅程度，还可以映射为柱形的宽度属性或者高度属性。通过解释比例尺的含义，坐标系描述了数据与空间位置的映射关系。

拥有了标记、比例尺和坐标系，NEV 可以生成与数据相关的可视图形，但是呈现的图形不利于用户解读。例如用户通过柱状图查看 2016 年每月的销售额，只通过比例尺、坐标系和标记生成的图表，在屏幕上展示的效果是一排高低相间

的柱形，这种可视化结果对用户解读数据的含义毫无意义。所以，一个有意义、有内涵的图表还需要一些辅助理解图表的组件，这些组件包括坐标轴、图例、标签等。NEV 系统通过坐标轴、图例实现比例尺的可视化展示，坐标轴是对坐标系的可视化呈现。

图表的样式直接影响标记的表现形态，进而间接影响了整个图表的外观。通过交互，用户对数据进行探索与发现，在此基础上加深对图表的理解。

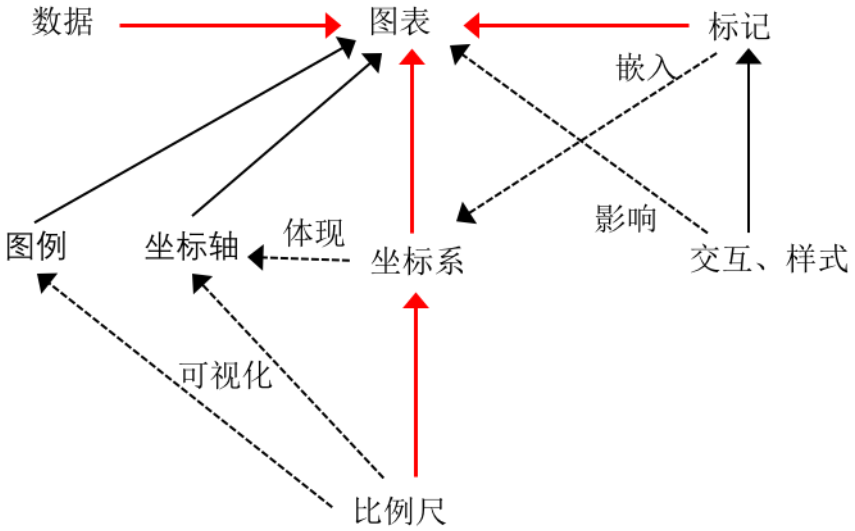

图 3.2 图表结构组成图

Babel 图形语言在描述图表结果时，主要数据结构的设计如下：

```
{
    "area": Object,
    "data": Object,
    "scales": Array,
    "coords": Array,
    "cells": [[cell]],
    "theme": Object,
    "interactor": Object
}
```

（1） area 字段决定了可视化区域的大小，包括宽度和高度。

（2） data 是数据源，其数据格式是同构的，同时可以指定数据显示的顺序。

（3） scales 字段声明了比例尺的定义域与值域，描述了数据维度向可视化编码的映射关系，在一个图表中存在多个比例尺。

（4） coords 字段用来描述坐标系，例如直角坐标系、极坐标系、蜘蛛坐标系等，坐标系能够进行嵌套使用。

（5） cells 字段是一个二维数据结构来着支持透视图表的描述，其中每个 cell 都包含了对图表标记图元和坐标轴的描述。

（6） theme 字段描述的是图表样式的个性化定制，以满足用户定制样式的需求。

（7） interactor 字段用来声明用户的交互操作，来表达用户希望图表具有的交互能力，通过声明式方式来描述事件流，详细设计与实现会在第四章节和第五章节进行介绍。

3.2.2 图表生成模块

NEV 不仅能够生成统计类图表，而且支持透视图表的绘制。图表生成模块通过场景分析将二维屏幕动态分割成多个部分，具体分割方式如图 3.3。图表展示空间被分割成多个部分，每个部分对应与图表相关的组件，每个组件在实现的过程中对应一个 Canvas。所以，多 Canvas 的设计将可视化空间分为图形区、公共轴区、透视轴区、图例区和标题区。

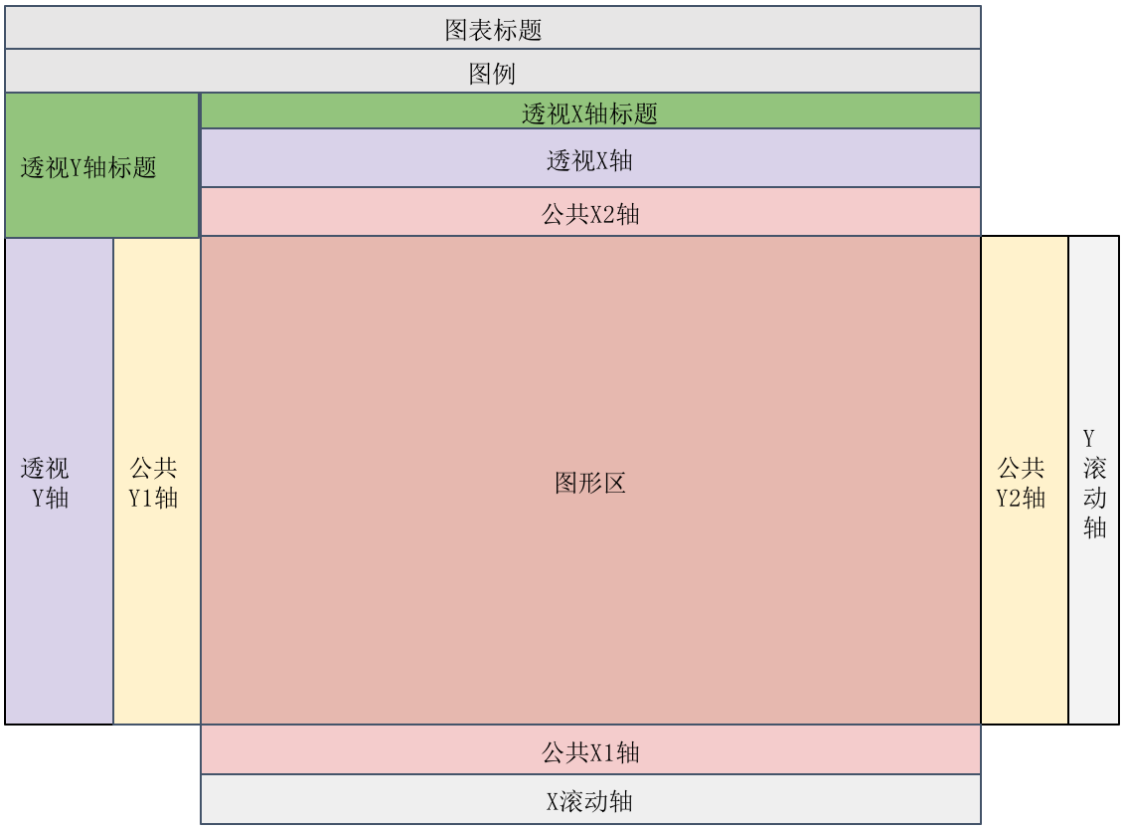

图 3.3 图表分区图

相对于整个图表而言，各个区是最小的通信单元，这些通信单元被系统称为图表分区单元（morph）。当数据量增大时，图形区不能显示全部的可视化图形，系统会产生滚动轴来扩展可视化空间。

morph 主要完成了标记（mark）的呈现以及图表交互的管理，如图 3.4 所示。在标记呈现方面，morph 负责创建场景树，实例化标记对象，设置标记对象的样式，计算图元布局数据。在图表交互方面承担了事件的管理和图表空间的管理。

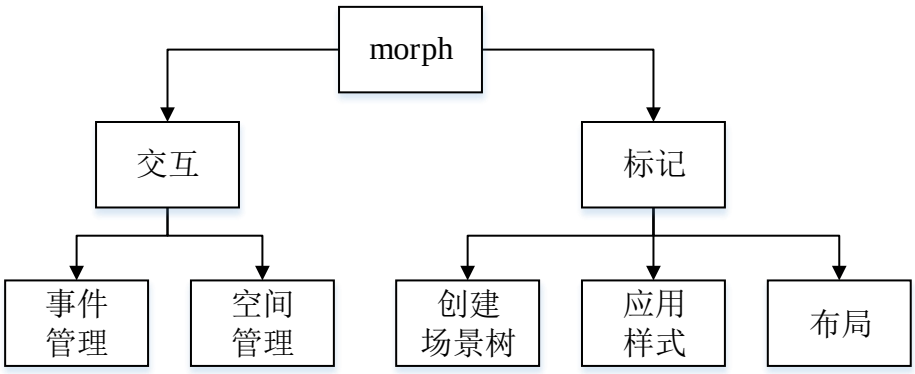

图 3.4 图表分区单元职能图

morph 单元完成的功能总结如下：

（1） 交互管理器的创建

交互管理器完成对多种鼠标事件的处理，构建空间树来选择发生交互的标记精灵。

（2） 主题管理器的生成

主题管理器解析各个 morph 单元的默认样式信息，并且融合用户自定义的样式以满足个性化的定制需求。

（3） 场景管理器的构建

场景管理器创建树形场景结构（场景树）来描述图表结构。

（4） 布局管理器的生成

布局管理器决定了各个 morph 单元的布局策略。

每一个图表有多个 morph，每个 morph 维护各自的数据集、管理各自的场景、样式和交互。morph 之间通过消息管道进行通信。每张图表有用一个全局的 morph 管理器，来接收和派发 morph 间的消息。

3.2.3 图表渲染模块

图表渲染层基于 Canvas 构建图表，承担了底层几何图元的构建、场景管理、交互管理和视图渲染的职责。

（1） 构建几何图元

在 Canvas 的接口基础上，图表渲染模块封装了 NEV 需要的的几何图元，这些几何图元实体被称做精灵（Sprite）。精灵是图表实现的基本单元，提供了设置样式和捕捉原始交互消息的功能。

图表渲染模块几何图元实体包括：扇形精灵、弧线精灵、圆形精灵、矩形精灵、折线精灵、路径精灵、凸多边形精灵和文本精灵，文本精灵根据文本行的特征细分分为单行文本精灵、多行文本精灵、自动换行文本精灵和携带背景色的文本精灵。凸多边形精灵用于构建其他使用率较低的几何形状，例如五角星、雪花形状的图形。

各种精灵都会维护各自的属性集合，包括 x/y 空间坐标、z 值、笔触的颜色、填充绘画的颜色、阴影的各个属性等。不同的精灵所提供的属性会根据各自的特征有所区别，例如扇半精灵会提供半径属性，而折线精灵拥有线宽属性。

（2） 场景管理

精灵是场景中的基本单元，图表渲染层在组织底层的精灵时，采用场景树结构来完成。场景树是一棵多叉树，场景中有一个 Root 节点，所有的精灵都被挂在以 Root 为根节点的场景树上。如图 3.5 所示，饼图主要由扇半、引导线和标签构成。

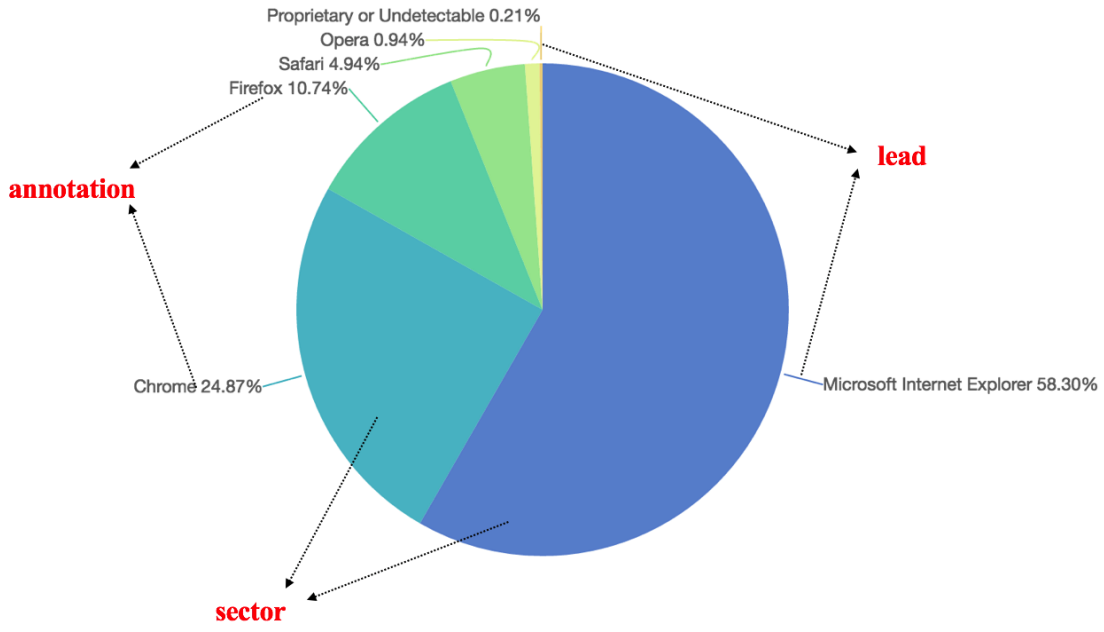

图 3.5 饼图结构图

这个饼图在图表渲染过程中，其场景结构如图 3.6 所示。由根节点出发，生成的场景树维护了多个饼图的场景子树，每个场景子树维护多个扇半、引导线和标签的场景子树。在这种树形结构的组织下，精灵节点被层级监控，场景结构得到统一管理。

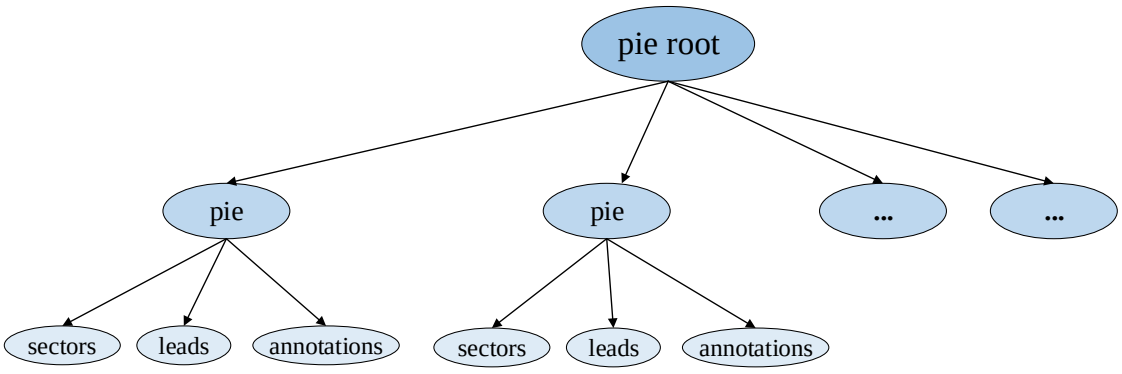

图 3.6 饼图场景图

（3） 视图绘制

视图绘制子模块将图表生成模块构建的精灵绘制到空间画布上，主要负责了 Canvas 元素的生命周期的管理、视图的渲染和更新重绘。

（4） 交互管理

交互管理子模块处理用户的交互操作，具体细节在本文第四章节进行详细描述，这里不再赘述。

3.3 主题模型

在 NEV 中，图表的样式是关系到图表显示美观与否的直接决定性因素。通常情况下，图表的样式分为与数据相关的视觉属性和与数据无关的视觉属性。

与数据相关的视觉属性中，包括标记的高度（垂直尺寸）、宽度（水平尺寸）、大小（半径长短、线的粗细等）以及颜色的色调、饱和度等。

与数据无关的视觉属性中，包括了图表中各种样式的细节。例如轴线的线宽与颜色，文本的字体、字号以及颜色，网格线的线宽等。虽然这些样式细节对观察数据本身没有意义，但是这些样式的设计直接关系到了整个图表的美观性，并且对理解图表的含义，起到重要的辅助作用。

为了 NEV 系统能够清晰表达这些视觉属性的默认样式，使用户摆脱设置全部样式的负担，同时满足用户对图表样式局部的高度定制化需求，本文提出了主题模型。下面重点介绍主题模型在默认样式和用户自定义样式两方面的设计。

3.3.1 默认样式

在主题模型中，NEV 将图表看做是多个原子单元的组合体，例如笛卡尔坐标系下的柱状图，可以看成是图表的标题、横/纵向坐标轴、图例以及一排柱形的组合体。这些原子单元的样式在主题模型中，是通过声明式方式描述，以表达出各个原子单元的样式是什么。例如为了清晰阐述图表标题的样式，需要声明标题的水平相对位置，竖直相对位置，标题的字体大小、粗细、颜色和对齐方式。主题模型提供了默认的色盘、字体库等默认配置来表达与数据无关的视觉属性的值。

同时，针对与数据相关的视觉属性，主题模型声明了默认比例尺的设置。在默认比例尺中，数据到颜色的映射关系被转换成每条数据记录都被映射为色盘的第一个颜色，数据到尺寸的映射被归一化到统一大小范围内。

3.3.2 用户自定义样式

关于图表的视觉属性，用户可以进行自定义来满足其可视化需求。针对与数据相关的视觉属性，通过自定义比例尺来声明数据到视觉属性的映射，这些视觉属性包括颜色、尺寸以及形状。用户通过声明比例尺的类型、定义域的数据类型以及映射到视觉属性的值域范围，来描述与数据相关的样式。

针对与数据无关的视觉属性，用户通过声明式方式来描述其想要自定义的样式。用户在声明图表结构过程中，针对想要自定义样式的原子单元，通过指定索引名称，从用户自定义的样式列表中索引到描述该原子单元的样式。这样的设计保证用户可以统一描述自定义的样式，并且这些自定义样式可以被复用。用户自定义样式的方法如下：

```
"theme": {
        "visualScales": {
           "scaleName": {}
        },
        "classStyle": {
           "axisName": {},
           "markName": {}
        }
}
```

"visualScales"部分用来描述自定义的比例尺，"classStyle"部分用来描述自定义样式。主题模型通过自定义比例尺和原子单元的样式来实现细粒度的样式定制，以满足用户关于样式的高度定制化需求。通过上述声明式描述，用户指定了图表样式的定制。

主题模型内部维护了两张表，一张表用来声明图表所有原子单元的默认样式，另外一张表用来描述图表分区单元的结构。主题模型在解析图表样式过程中，优先使用自定义的样式，其余使用主题模型的默认样式来构建出图表的全部样式。主题模型进行用户自定义的样式与默认样式的融合，将用户自定义的样式覆

盖默认样式相对应的部分，生成新的样式表。通过使用新的样式表、图表分区单元的结构表，主题模型解析出各个图表分区单元的样式集合。图表分区单元在需要样式时会主动向主题模型查询样式，主题模型根据不同的查询条件反馈相对应的样式集合。

3.4 本章小结

本章节主要概括了系统的架构设计、各个子模块的设计和主题模型的设计。系统分为三大部分，分别是图表描述模块、图表生成模块和图表渲染模块。图表描述模块说明了 NEV 中 Babel 图形语言的设计。图表生成模块讲述了图表分区和生成图表的设计。图表渲染模块通过视图渲染功能将最终的图表视图渲染到指定的二维空间画布上。最后，本章介绍了针对图表的样式描述，提出了主题模型。

第4章 声明式交互设计与实现

本章节重点阐述声明式交互模型的实现。本章提出了一套声明式交互语法来描述用户的交互动作。这个声明式交互语法是在图形语法基础上拓展而来的，为事件流提供了低层次的组合方式，来统一描述用户对图表的交互操作。然后本章节将详细介绍在数据可视化系统中，用户的交互操作是如何被解析并处理，最终通过图表将交互结果反馈给用户。

4.1 事件流模型设计

本文提出的声明式交互模型采用事件驱动的函数式响应型编程语义，通过捕捉用户与图表发生交互时产生的输入事件（例如鼠标事件、键盘输入等），将这些事件表达为事件流，而不是使用回调函数的方法来对用户交互进行建模。回调函数与回调函数之间是无法直接传递类型丰富的数据的，它们只能通过修改应用程序的共享状态来间接地通讯，这样不得不把应用逻辑分割得支离破碎，会丧失掉核心的组合能力。本文提出的声明式交互模型增强了描述交互的组合能力。

在本文的声明交互模型中，每一个事件都会被捕捉并处理成为一个唯一的事件流，这个事件流可以通过对原生事件转换、组合等操作来生成。以声明式的方式来描述事件流的过程是通过四种基本操作符来完成的。四种操作符包括逗号符号（,）、花括号符号（{ }）、中括号符号（[ ]）以及大于号符号（>）。

4.1.1 原生事件

原生事件包括鼠标按下事件（mousedown）、鼠标抬起事件（mouseup）、鼠标移动事件（mousemove）、鼠标悬浮事件（mouseover）、鼠标移出事件（mouseout）和鼠标滚动事件（mousewheel）。

对于简单的交互操作，交互产生的原生事件可以直接看做事件流，不需要经过操作符的组合来表达。例如当鼠标悬浮在柱状图中的柱形上时，mouseover 原

生事件可以看做是发生在条柱形上的 mouseOver 事件流。如图 4.1 所示，鼠标在柱状图上移动的过程中，产生了 mouseOver 事件流。

图 4.1 原生事件作为事件流

4.1.2 逗号操作符

逗号操作符表示通过融合事件产生新的事件流，如图 4.2，表达式 mousedown，mouseup 表示通过融合用户交互操作产生的 mousedown 和 mouseup 事件来组合产生新的事件流。用户的选中操作和取消选中操作即可通过逗号操作符来组合成新的事件流来表示这个用户操作。

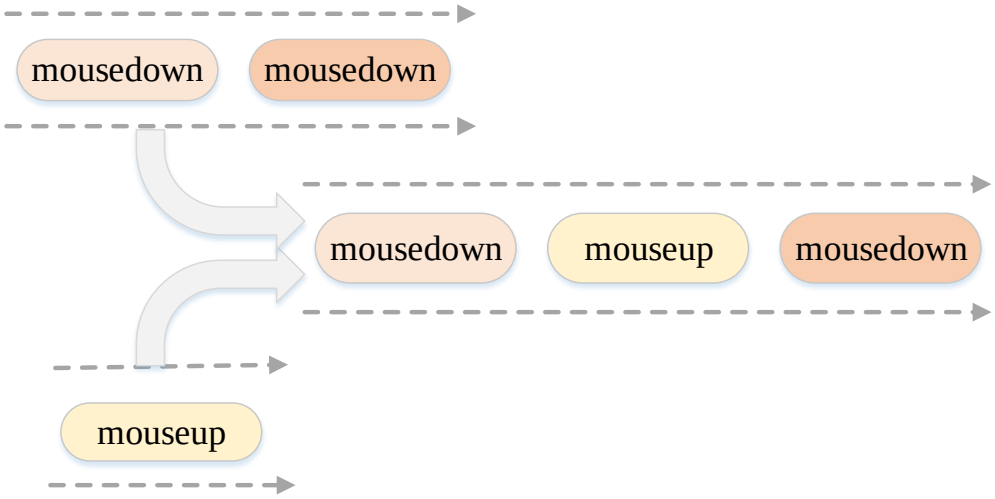

图 4.2 融合 mousedown，mouseup 事件流示意图

4.1.3 花括号操作符

花括号操作符表示在事件发生的过程中最小、最大的时间间隔，如图 4.3 所示。表达式 mousemove{3ms, 5ms}表示选择出 mousemove 事件组成新的事件流，

并且这些 mousemove 事件需要满足以下条件：发生的时间间隔在 3 毫秒到 5 毫秒之间。

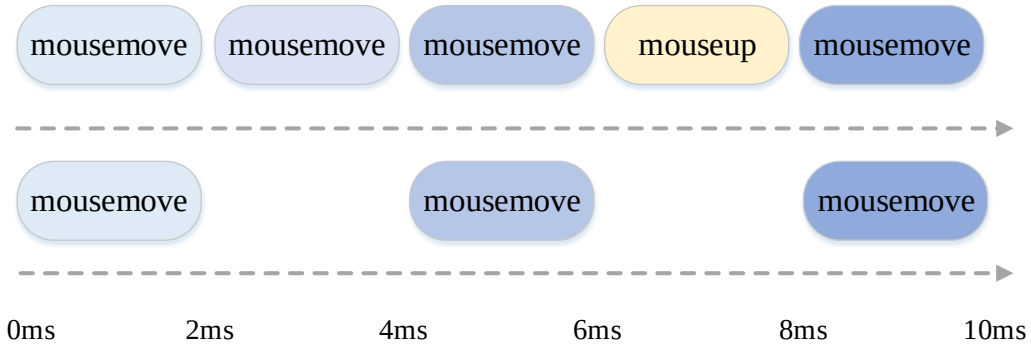

图 4.3 使用花括号过滤事件流

在图 4.3 中，我们首先将第一个 mousemove 事件加入新的事件流。由于第一个 mousemove 事件发生在 0ms，第二个 mousemove 事件发生在 2ms，二者不符合 3ms 到 5ms 的时间间隔，因此第二个 mousemove 事件将会被过滤掉。同理，我们对后面的事件进行过滤，最终形成了新的事件流。

4.1.4 中括号操作符和大于号操作符

中括号操作符表示基于某些条件过滤事件以生成新的事件流，中括号描述的是筛选条件，如图 4.4 所示，表达式 click[event.pageY >= 300][data.price < 500]表示选择出 event.pageY 大于等于 300 并且 data.price 小于 500 的点击事件来生成新的事件流。

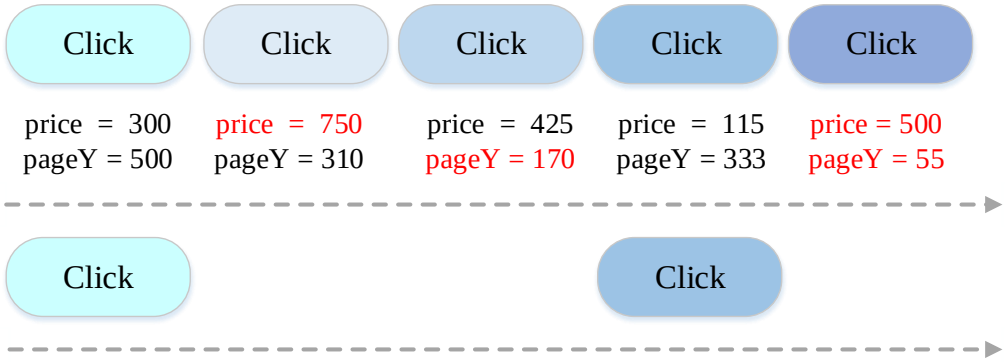

图 4.4 使用中括号和大于号过滤事件流

当在中括号后加上大于号（>）时，表示在中括号操作过滤生成的事件流基础上，对这个新生成的事件流再进行某种条件的筛选，如图 4.5，[mousedown, mouseup] > mousemove 表示筛选出发生在 mousedown 和 mouseup 事件之间的 mousemove 事件，来合成新的事件流。

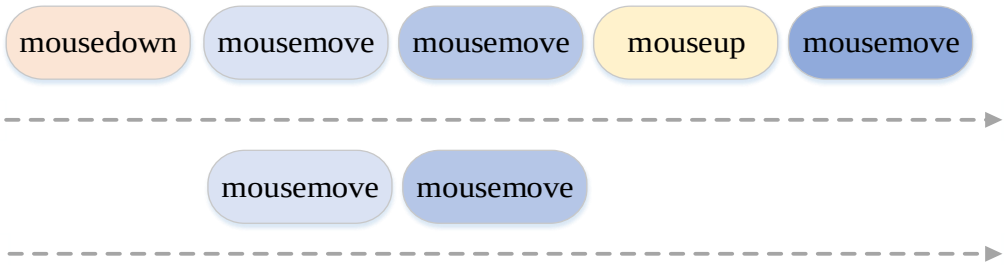

图 4.5 合成组合事件流

4.1.5 事件流的组合

我们可以通过四种运算符对事件流进行组合，来生成新的事件流。新生成的事件流还可以通过这四种运算符组合出更复杂的事件流。例如[mousedown, mousemove] > [keydown, keyup] > mousemove{2ms, 6ms} 这个合成的事件流表示在发生的 mousedown 事件和 mousemove 事件之间选择出 keydown 事件和 keyup 事件，然后在此基础上选择出时间间隔在 2 毫秒到 6 毫秒之间的 mousemove 事件。

原生事件是基本事件流，基本事件流可以在不做任何修改的情况下被转换、组合成新的复杂的事件流。而新的复杂的事件流又可以在不做任何修改的情况下，作为运算符的输入，被转换、组合成更复杂的事件流，就像搭建乐高积木一

样，这个过程可以一直进行下去，直到构建出足够复杂的信号以满足用户的交互需求。

在本文描述的模型中，事件的关注点从事件的控制流层次转换到事件的数据流层次，以简单直观、符合人类思维的方式，来描述用户交互具体要做些什么，而不是去描述如何来完成用户的交互操作。因此，通过这种声明式的方式描述的用户交互，不仅使用户容易理解，而且强大的组合能力使得用户可以描述各种复杂的交互需求，以满足对图表背后的数据的探索。

另外，文本声明式交互模型中设计的事件流操作符很容易理解，这些操作符的设计灵感来源于 CSS3 的选择器，对于可视化设计者来说会更加熟悉，使用起来也会得心应手。

4.2 交互流程

当用户与图表发生交互时，鼠标事件或者键盘事件将产生事件流，图表中的某些元素需要对这些事件进行响应和反馈，以达到用户操作图表来观察数据的目的。其中，从事件流的产生到最终事件的反馈，整个流程如图 4.6 所示。在用户与图表进行交互的过程中，触发的事件被看做事件流，经过转换和组合，将会形成新的事件流，新形成的事件流依次经过分发和处理，最后处理结果在图表上显示为视觉信号的变化来实现对用户的反馈。

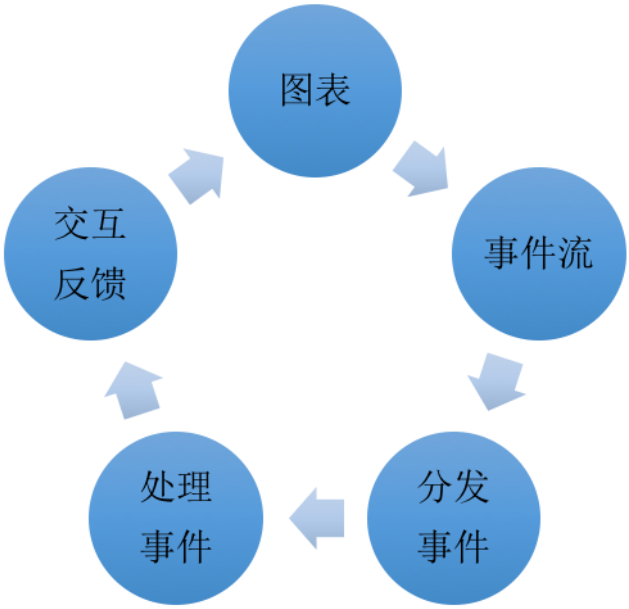

图 4.6 事件处理流程图

为了实现上述交互流程，保证系统一定会对用户的交互反馈结果，本章建立了一个事件处理模型，如图 4.7 所示，依次实现了事件仲裁器（EventDecider）、事件分发器（EventDispatcher）以及行为处理器(Action)。

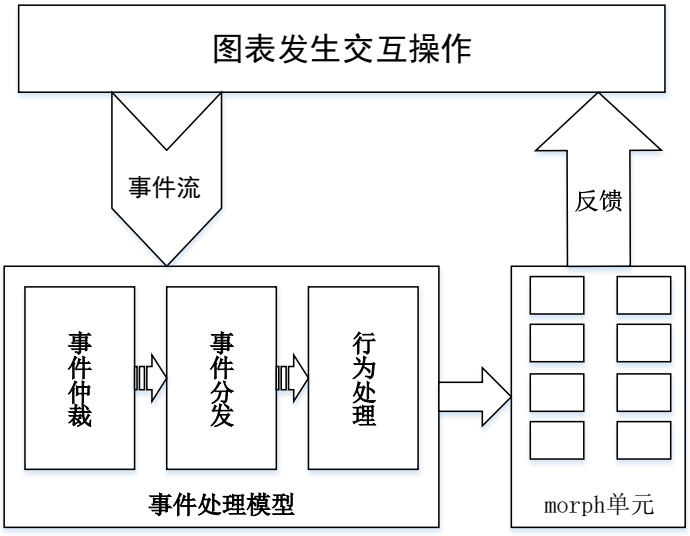

图 4.7 事件处理模型

在这个事件处理模型中，交互事件经过事件仲裁器、事件分发器和行为处理器，每一步骤都会产生相对应的消息模型，具体每一层级产生的消息模型见图 4.8。事件仲裁器对用户描述的交互操作进行解析，转换成内部的事件流格式；事件分发器直接接收事件流的交互消息，将事件流的交互消息经过一系列的逻辑判断转为操作（operation）消息；行为处理器收到操作消息后，对这些操作消息进行解释处理，生成描述用户交互行为的行为（action）消息，并且实现了用户的交互意图，最终将交互处理结果响应在图表上，反馈给用户。

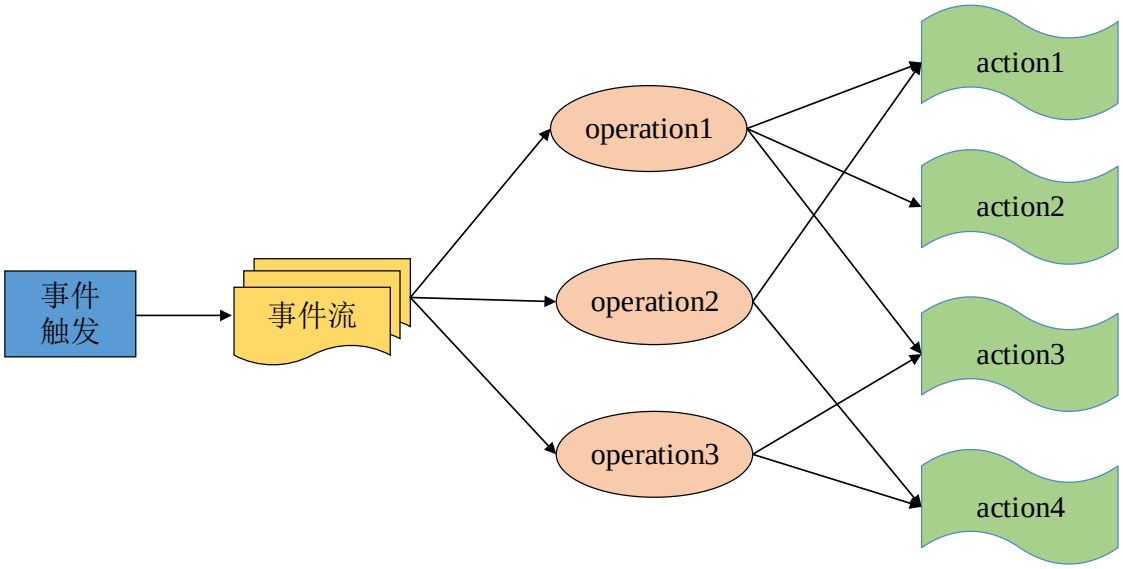

图 4.8 数据流转换图

下面的章节将详细讲述事件仲裁器、事件分发器以及行为处理器如何将用户的交互事件依次转换为内部事件流、操作消息和行为消息。

4.3 事件仲裁

事件仲裁器将用户描述的交互模型转换成系统内部的事件流，包括 mouseOver 事件流、mouseOut 事件流、mouseWheel 事件流、mouseClick 事件流、mouseDrag 事件流以及 mouseMove 事件流。

产生 mouseOver 事件流的声明式描述是 mouseover；

产生 mouseOut 事件流的声明式描述是 mouseout；

产生 mouseWheel 事件流的声明式描述是 mousewheel；

产生 mouseClick 事件流的声明式描述是 mousedown；

产生 mouseDrag 事件流的声明式描述是 mousedown, mouseup, [mousedown, mouseup] > mousemove；

产生 mouseMove 事件流的声明式描述是 mousemove；

4.4 事件分发

在数据可视化系统的事件处理模块中，事件分发器将事件仲裁器输入的各种事件流，经过一系列的逻辑判断封装成操作消息（opMsg），操作消息包括 over 消息、out 消息、move 消息、drag 消息、leftClick 消息、rightClick 消息；并且选择出节点精灵，将操作消息以及选中的标记图元精灵一起分发到相关的图表分区单元（morph）中。

下面重点介绍事件分发器三大核心模块，分别是操作消息模型的构建、图元选择器，以及事件分发机制。

4.4.1 构建操作消息

事件分发器在对事件流处理的过程中，将事件仲裁器输出的事件流抽象转换成操作消息，具体转换细节参见图 4.9。

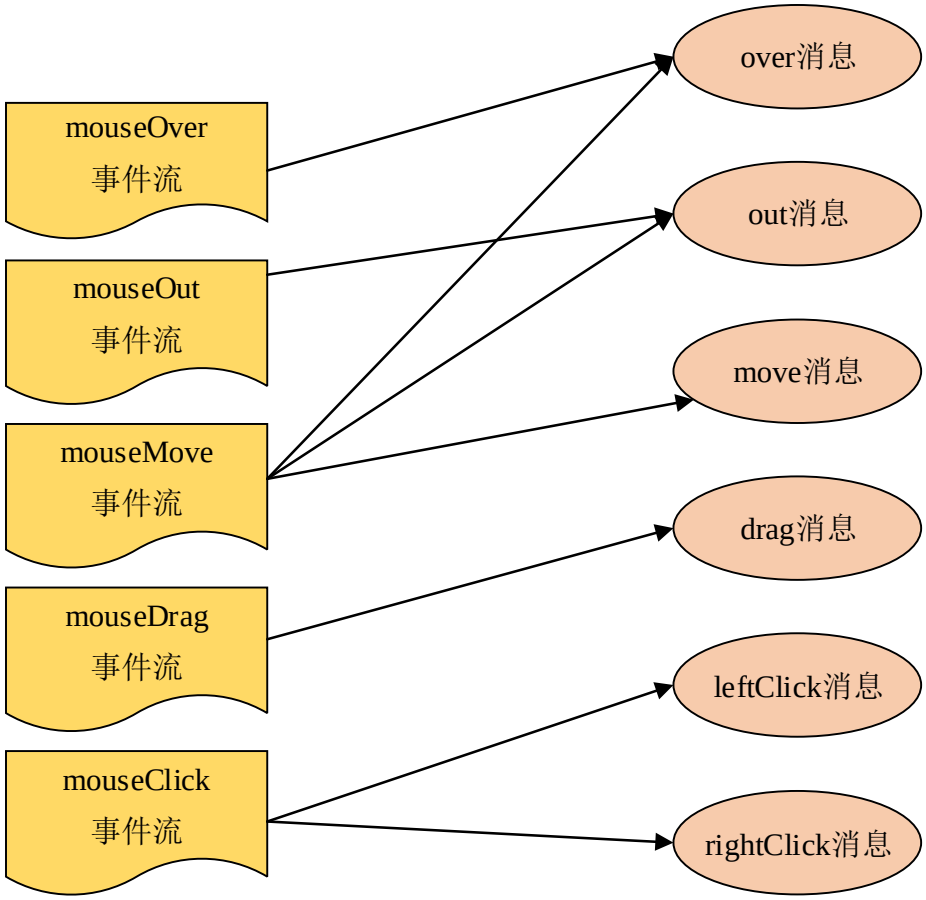

图 4.9 事件流到操作消息转换图

由图 4.9 可以看出，事件分发器完成了 mouseOver 事件流到 over 操作消息的转换；mouseOut 事件流到 out 操作消息的转换；mouseMove 事件流到 over 操作消息、out 操作消息，以及 move 操作消息的转换；mouseDrag 事件流到 drag 操作消息的转换；mouseClick 事件流到 leftClick 操作消息、rightClick 操作消息的转换。某种事件流可以转换成一种操作消息，例如 mouseOver 事件流；也可以转换成多种操作消息，例如 mouseMove 事件流。

下面将详细阐述各种事件流转换成多种操作消息的逻辑处理流程。

（1） 处理 mouseOver 事件流

当事件分发器收到 mouseOver 事件流时，将其转换成 over 操作消息，通过图元选择器选出发生交互的图元精灵节点，然后向发生交互的图表分区单元发送 over 操作消息，完成 mouseOver 事件流转换成操作消息。

（2） 处理 mouseOut 事件流

当事件分发器收到 mouseOut 事件流时，将其转换成 out 操作消息，通过图元选择器选出发生交互的图元精灵节点，然后向发生交互的图表分区单元发送 out 操作消息，完成 mouseOut 事件流转换成操作消息。

（3） 处理 mouseMove 事件流

当事件分发器收到 mouseMove 事件流时，在不同的逻辑情况下会转换成不同的操纵消息，这些操作消息分别是 over、out、move 消息。具体的逻辑判断流程如图 4.10 所示。

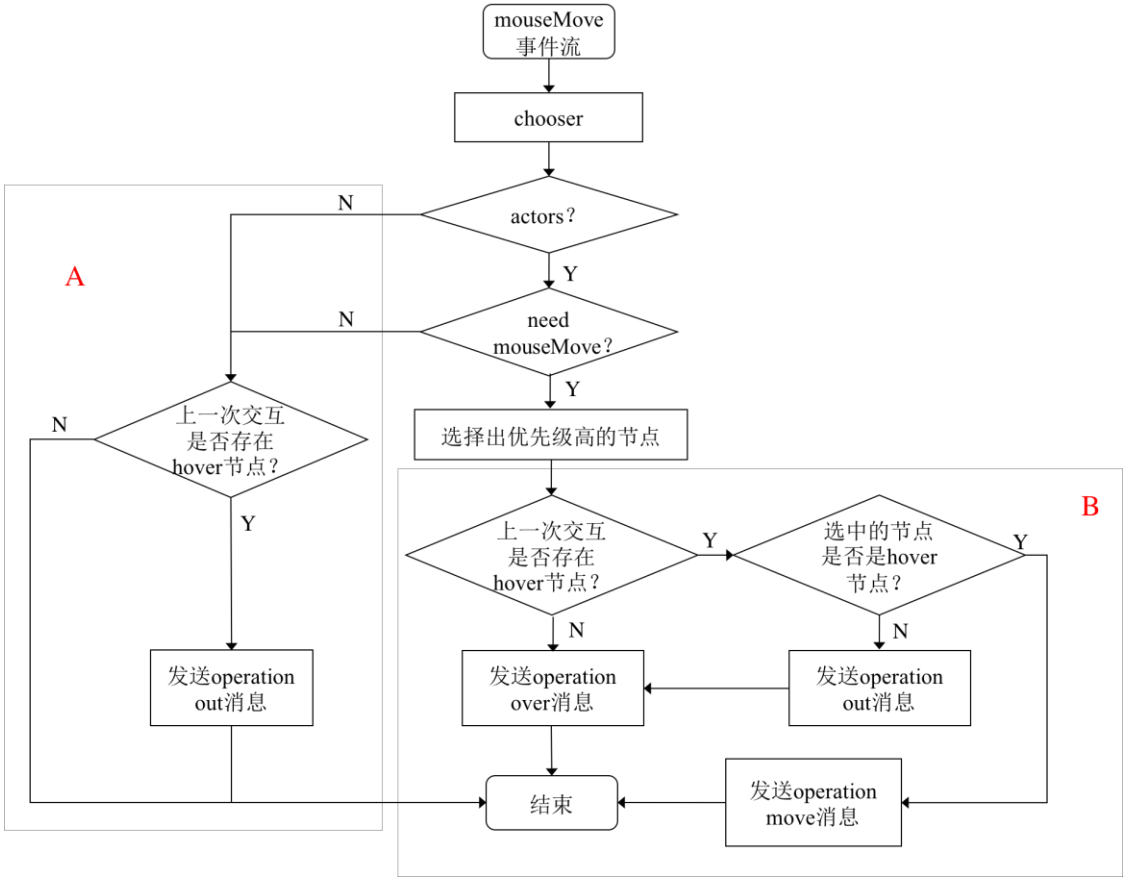

图 4.10mouseMove 逻辑判断流程图

当 mouseMove 事件流到达事件分发器中，通过图元选择器选择出当前发生交互的图元精灵节点，接下来要判断这些节点存在或者不存在，另外还要区分这些

选择出来的精灵节点是否需要 mouseOver 事件流，由此整个流程分为 A、B 两个子流程。

A 子流程在两种情况下发生，分别是图元选择器没有找到符合条件的精灵节点，或者找到的精灵节点并不需要当前的 mouseOver 事件流。此时需要判断上一次交互事件流中是否存在满足条件的节点，如果不存在则本次 mouseMove 事件流处理完毕后，不发送任何操作消息；如果存在，那么事件分发器发出 out 操作消息，以改变上一次交互事件流结束后相关节点的状态。

B 流程是在选择出了发生交互的精灵，并且这些精灵需要当前的 mouseOver 事件流的情况下产生的流程。通过对选择出来的节点的优先级进行判断，找出优先级最高的节点作为 mouseMove 事件流的响应节点；确定了节点之后，需要关注上一次交互事件流中是否存在满足条件的节点，如果不存在，则事件派发器对当前选中的节点发送 over 操作消息；如果存在，则需要判断上一次选中的节点和本次选中的节点是否是相同的节点，如果不是相同的节点，则对上一次选择的节点发送 out 操作消息，然后再对当前选择的节点发送 over 操作消息；若这两次 mouseMove 事件流针对的是同一个节点，那么对这个节点发送 move 操作消息。此时，mouseMove 事件流处理完毕。

由此可知，在 mouseMove 事件流发生后，会根据各种逻辑情况来生成不同的操作消息，这些操作消息被分发给相应的图元精灵节点进行响应。

（4） 处理 mouseDrag 事件流

当事件分发器收到 mouseDrag 事件流时，将其转换成 drag 操作消息，具体的逻辑判断流程如图 4.11 所示：

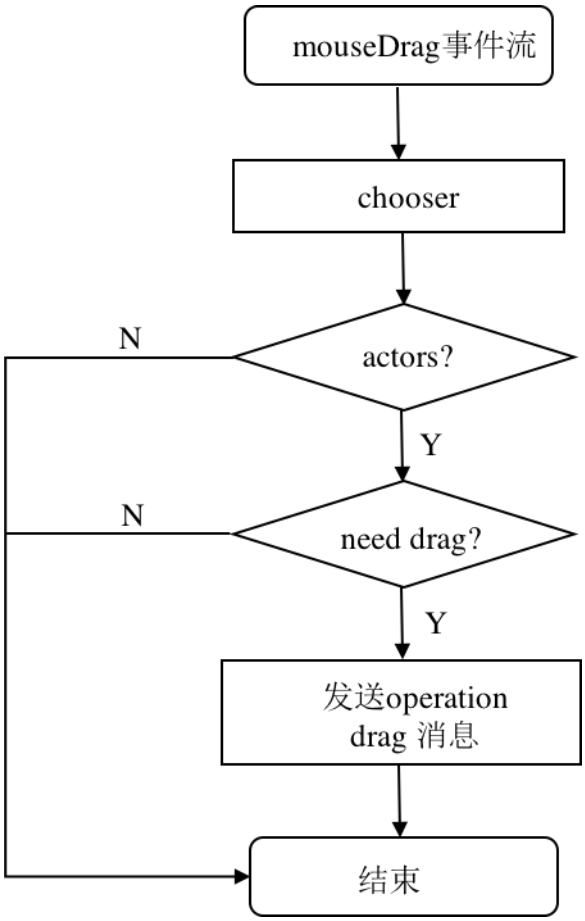

图 4.11mouseDrag 逻辑判断流程图

当 mouseDrag 事件流到达事件分发器时，通过图元选择器选择出当前发生交互的精灵节点的集合，只有这些精灵存在并且需要当前 drag 消息，事件分发器才会进行 mouseDrag 事件流的分发；否则对这个 mouseDrag 事件流不进行任何处理。

（5） 处理 mouseClick 事件流

当事件分发器收到 mouseClick 事件流时，将其转换成 leftClick 操作消息和 rightClick 操作消息，具体的逻辑判断流程如图 4.12 所示：

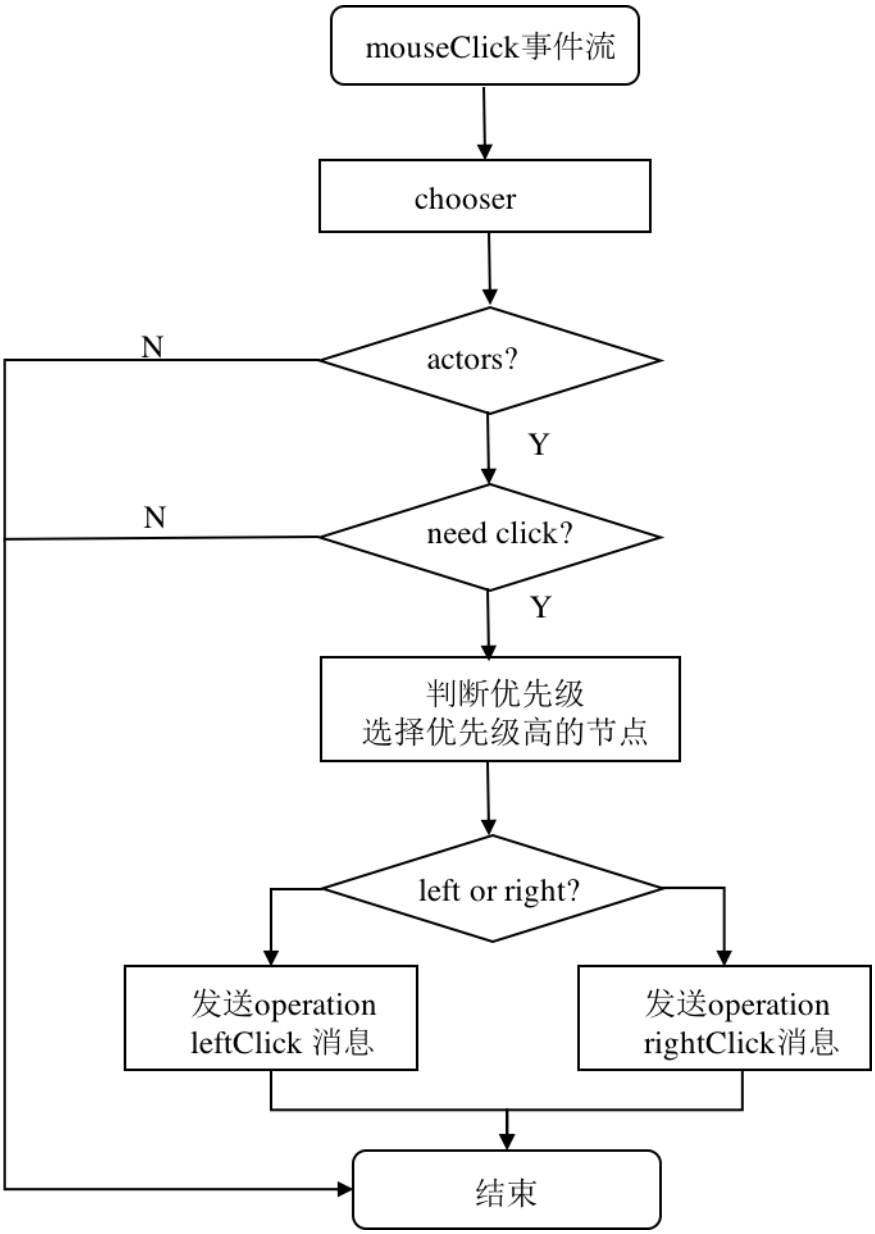

图 4.12 mouseClick 逻辑判断流程图

当 mouseClick 事件流到达事件分发器时，通过 chooser 选择器选择出当前发生交互的精灵节点的集合，只有这些精灵存在并且需要当前 click 消息，事件分发器才会进行 mouseClick 事件流的分发。在分发 click 事件之前，需要通过对选择出来的节点的优先级进行判断，找出优先级最高的节点作为 mouseClick 事件流的响应节点；同时需要通过对 mouseClick 事件流进行分析，区分开此次事件流是左

键 click 事件流还是右键 click 事件流，根据事件流的不同情况，事件分发器来分发相对应的 leftClick operation 消息和 rightClick operation 消息。

4.4.2 图元选择

在构建操作消息模型过程中，需要使用图元选择器。根据坐标、区域等信息，从空间树上将图表上发生了交互的节点精灵选择出来。空间树是图表中的所有图元精灵节点的逻辑组织结构，通过多叉树、二叉树以及混合树的树形结构来组织精灵节点的空间，从而加速了空间中场景精灵节点的选中操作。

4.4.3 事件分发机制

事件分发器是根据事件流的类型，将事件流转换而得到的操作消息和发生交互的精灵节点，通过管道机制分发到事件发生的图表分区单元（morph）中，每一个图表分区单元控制各自交互的处理细节。

另外每个图表分区单元有两个管道，分别是输入管道和输出管道。通过自身的输出管道可以发送操作消息给其他相关的图表分区单元，以实现图表分区单元间的消息通信。通过输入管道不仅可以接收事件分发器派发的操作消息，还可以接收其他图表分区单元发来的操作逻辑消息，并对这些操作逻辑消息进行响应，将结果显示在图表上，反馈给用户。

4.5 行为处理

行为处理器将事件分发器生成的操作消息转换成行为消息，构建并处理行为（action）模型。行为模型用来描述用户的交互行为，从而实现各种用户的交互操作。

在行为模型中定义了若干的行为动作，其中可以分为两类：原子型动作和组合型动作。其中，原子型动作是不可再分的最小粒度的行为动作，包括 hover、dishover、select、unselected 等。组合型动作通过原子型动作的组合和逻辑控制从而生成的一套动作，其中涉及到的组合和逻辑控制主要由 then 和 compete 两种算子进行组合。then 是顺序算子，then（a1,a2） a1 与 a2 是顺序关系，执行完 a1 后

执行 a2。Compete 是分支算子，compete（cond,a1,a2） a1 与 a2 是分支关系，cond 为分支判断条件。组合算子通过 then 和 compete 两种方式将多个行为进行组合，产生逻辑上的一个行为的算子。

行为模型将操作消息转换成行为消息，这些行为消息用来描述用户的交互动作。over 操作消息是由鼠标悬浮（hover）动作、显示数据提示框（showTooltip）动作触发。out 操作消息由取消鼠标悬浮（dishover）动作、隐藏数据提示框（hideTooltip）动作触发。move 操作消息由隐藏数据提示框（hideTooltip）动作触发。leftClick 操作消息由鼠标点击（select）、取消点击（deselect）动作触发。drag 操作消息由鼠标点击（select）、取消点击（deselect）动作和平移（translation）动作触发。rightClick 操作消息由显示菜单（showMenu）动作触发。具体转换细节详见图 4.13。

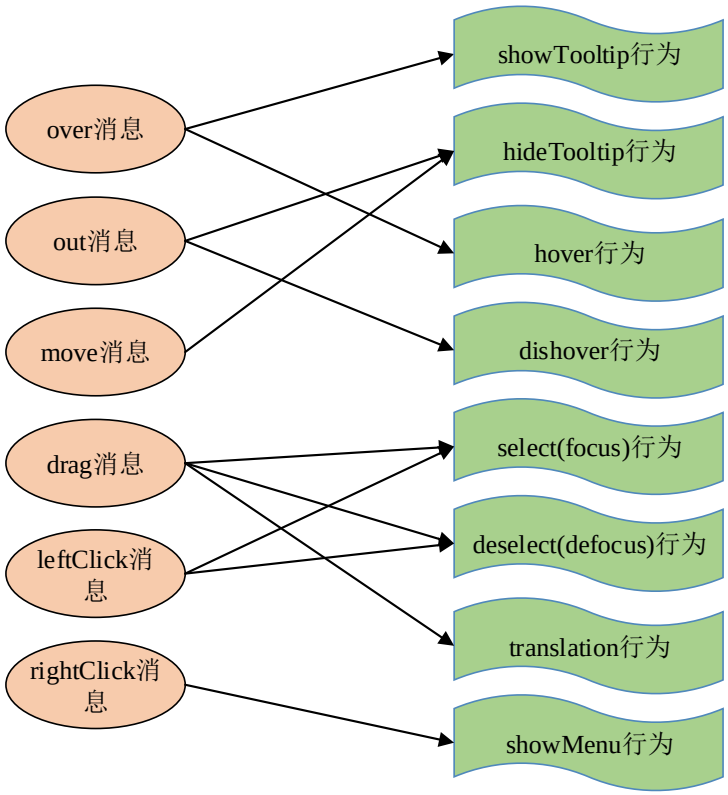

图 4.13 操作消息转换成行为消息

抽象出用户行为模型的好处在于不仅可以提高交互逻辑代码的复用率和执行效率，使代码易于修改和维护，还可以支持用户自定义的函数来描述用户想要的交互操作。并且行为消息之间可以依次传递，构建出更加复杂的用户交互操作，以满足用户的交互需求。另外，通过跨越不同的图表分区单元处理行为消息，系统可以实现图表分区单元间的交互逻辑操作。

4.6 本章小结

本章首先介绍了声明式交互的设计理念，抽象出事件流模型。其次，介绍了基于事件流模型的交互实现。事件仲裁器对事件流的转换、组合以生成系统内部需要的事件流结构。事件分发器处理新的事件流，生成操作消息，并把这些消息分发给交互发生的图表分区单元。每个图表分区单元控制着把操作消息转化成行为消息来完成用户的交互操作，最后将交互结果反馈在图表中。

第5章 系统应用与可视化实例展示

本章节介绍 NEV 数据可视化系统的应用。通过一系列的交互式可视化图表实例，本章展现了通过 NEV 绘制的图表的交互能力，评估了声明式交互的可表达性，验证了基于事件流模型实现交互的可行性。同时，本章展示了主题模型在系统中的应用。

5.1 系统应用

NEV 已经作为子系统外接入网易敏捷商业智能平台——网易有数[33]项目中。

5.1.1 网易有数系统架构

如图 5.1 所示，网易有数系统架构主要分为四部分，下面本小节对每一部分进行详细介绍。

展示层：包括系统界面和本文描述的数据可视化系统 NEV。系统界面展现了网易有数的使用入口。作为网易有数展示层重要的子系统，NEV 负责为用户提供与数据交互的接口，生成并渲染交互式可视化图表，帮助用户自由的探索数据，发现数据价值，对网易有数的用户体验起着至关重要的作用。

智能推导层：完成图表自动化推导，生成图形透视表的配置。

控制层：负责响应展示层的请求，合成智能推导层需要的数据结构。

数据层：存储各种数据源，包括 MySQL、Oracle 等。完成对各种数据源的请求，并对请求的数据进行处理，得到智能推导层需要的数据结构。

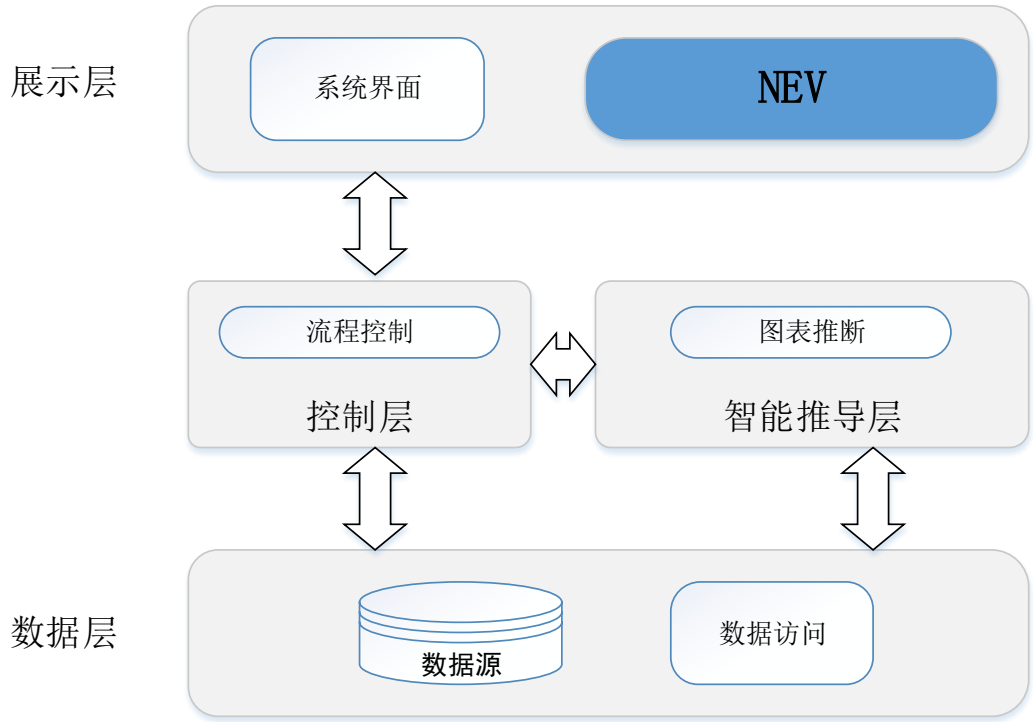

图 5.1 网易有数系统架构图

5.1.2 网易有数系统界面

如图 5.2 所示，网易有数的系统界面主要包括三个区域：数据描述区域、可视化查询输入区域和图表渲染区域。

红色框部分是数据描述区域，分成维度和度量两个类目。系统根据数据源的特性判断类别，将数据字段导入维度和度量。

黄色框部分是可视化查询输入区域，输入的内容被智能推导层使用，包括筛选器和行列字段。

蓝色框部分是图表渲染区域，包括 NEV 的图表绘制区域和可视化属性面板，可视化属性面板是用户自定义图表样式的入口。

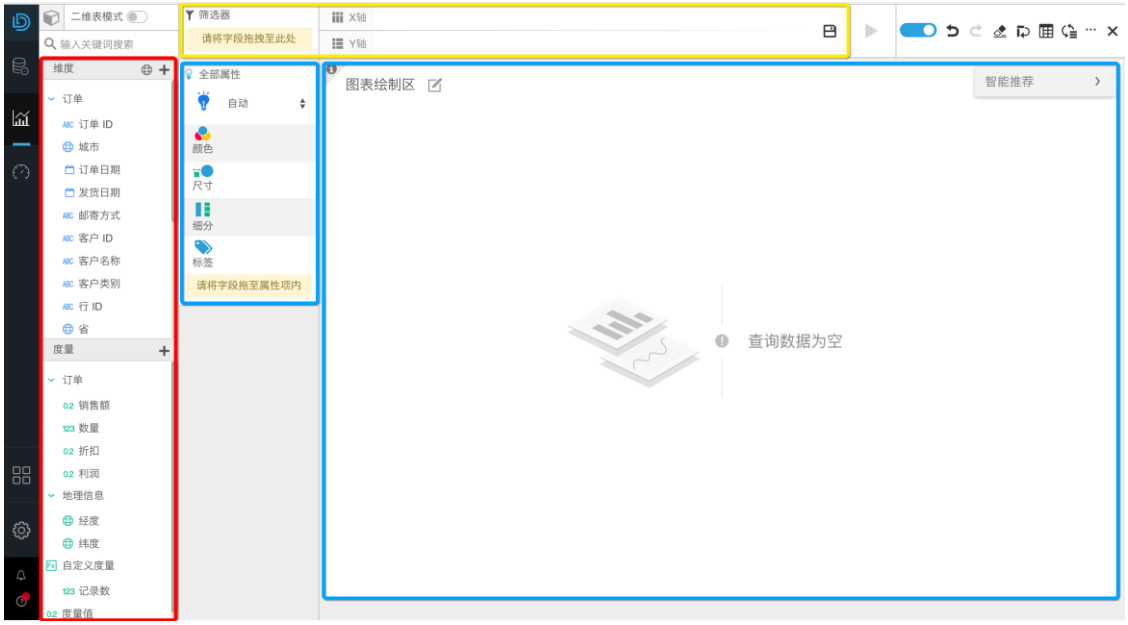

图 5.2 网易有数用户界面

5.2 数据描述

本章节使用的测试数据来自经典的 Superstore 数据集，主要使用的字段包括离散维度字段和连续度量字段，具体的字段内容如表 5.1 所示。

表 5.1 使用的数据集字段表

离散维度字段	地区
	客户类别
	产品类别
	发货日期
	省
连续度量字段	销售额
	折扣
	利润
	经度、维度

数据在维度属性上呈现出离散状态，包含了地区、客户类别、产品类别、发货日期和省这五种维度属性。数据在度量属性上呈现连续状态，包含了销售额、折扣、利润、经度和纬度这五种度量属性。用户可以在这些数据属性上自由地探索数据，发现数据中有价值的信息。

5.3 选择操作实例

通过选择交互，用户可以选中感兴趣的数据点进行观察，系统支持三种方式来进行选择，分别是：悬浮操作、点击操作和框选操作。

为了更好地展示交互过程中图表形态的变化，本小节介绍未进行交互时图表形态，如图 5.3 所示。

图 5.3 未发生交互操作示意图

图 5.3 主要展示了在不同地区的各种客户类别中，每种产品的销售额。条状图的颜色标识了不同的产品类别，条状图的宽度标识了销售额的多少。

5.3.1 悬浮操作

从图 5.3 可以看出，华东地区的消费者对办公用品的需求很高。此时，如果用户想探究其销售额的定量描述，可以通过鼠标悬浮操作来高亮显示感兴趣的图

元。如图 5.4 所示，当鼠标悬浮在图中的柱形上时，选中的柱形会高亮显示，该柱形还会产生偏移标签来显示数据的详细信息。可以看出华东地区的消费者对于办公用品的购买数额达到 740876。

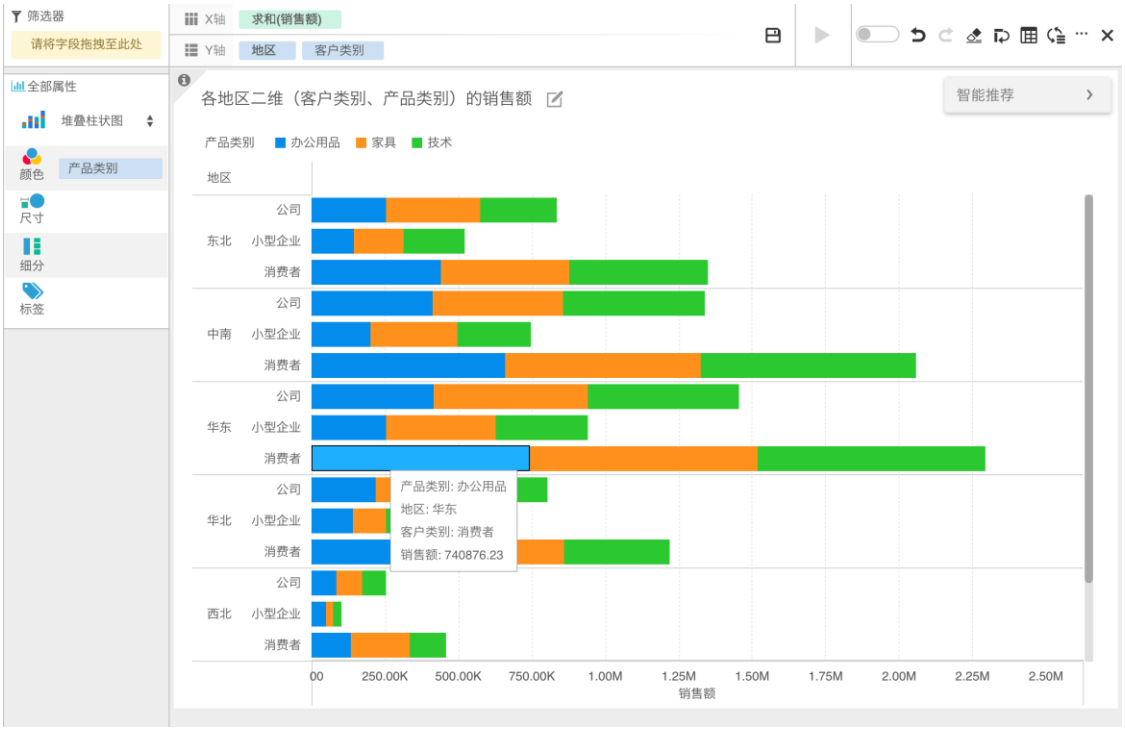

图 5.4 鼠标悬浮选中图元

在本文描述的数据可视化系统中，实现悬浮操作的声明式表达式为：

```
"streams": [
    {"type": "mouseover", "in": "graph"},
    {"type": "mouseout", "in": "graph"}
]
```

在图表的图形区，本系统通过对 mouseover 事件流和 mouseout 事件流的捕捉，来表达鼠标移入图元和移出图元的操作。另外还在 x 轴区、y 轴区和透视图表的标题区添加了对鼠标移入/移出事件流的监控。事件处理模型处理这些事件流，并将处理结果反馈给用户，在图 5.4 中是通过高亮图元和显示偏移标签来实现反馈的，并且其他的图元表现形态没有发生变化。当鼠标移走后，被高亮显示的图元

会恢复到原始状态，并且提示框也会随之消失。如图 5.5 所示，鼠标从高亮显示的柱形向左移动进入 y 轴区后，被高亮显示的柱形恢复到原始形态，相关的偏移标签随之消失，同时 y 轴区被高亮显示。

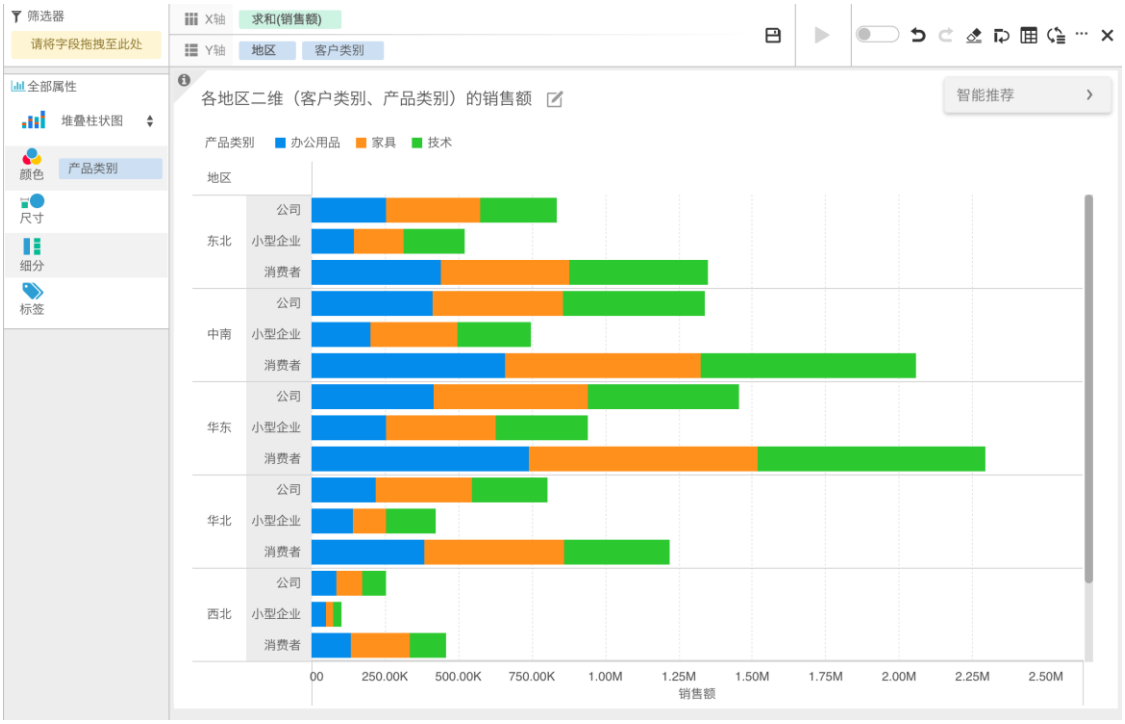

图 5.5 鼠标悬浮选中轴区

5.3.2 点击操作

当用户想要突出显示华东地区消费者对家具的购买数值时，可以通过鼠标点击操作来完成。如图 5.6 所示，选中的柱形被高亮显示，同时产生了对数据信息描述的偏移标签。其他未被选中的柱形均被置灰。系统通过颜色上的反差来凸显用户关注的数据信息。

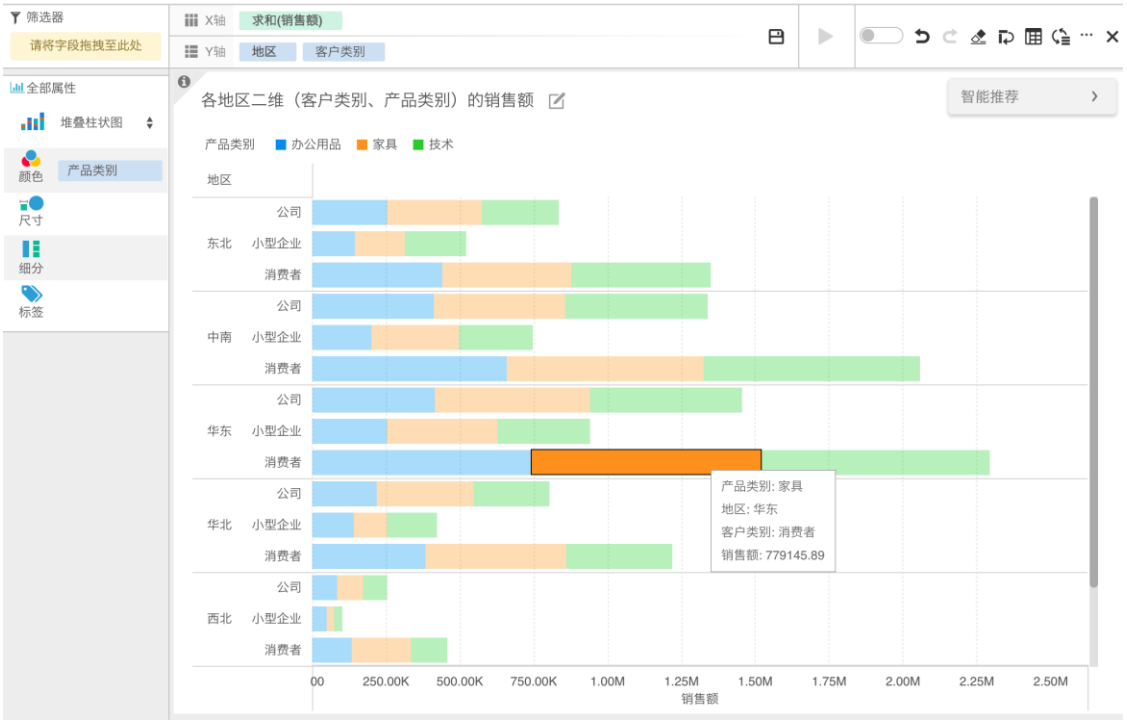

图 5.6 鼠标点击图元

本文描述的数据可视化系统支持通过鼠标点击来高亮显示关注的数据点，描述这个操作的声明式表达式为：

```
"streams": [
    {"type": "mousedown", "in": "graph"}
]
```

在图表的图形区通过捕捉 mousedown 事件流来描述用户的点击选中操作和取消选中操作。当用户想要取消点击操作时，可以点击当前选中的图元。如图 5.7 所示，再次点击选中的图元，置灰的图元会恢复成原始形态，被选中的图元变为鼠标悬浮状态。

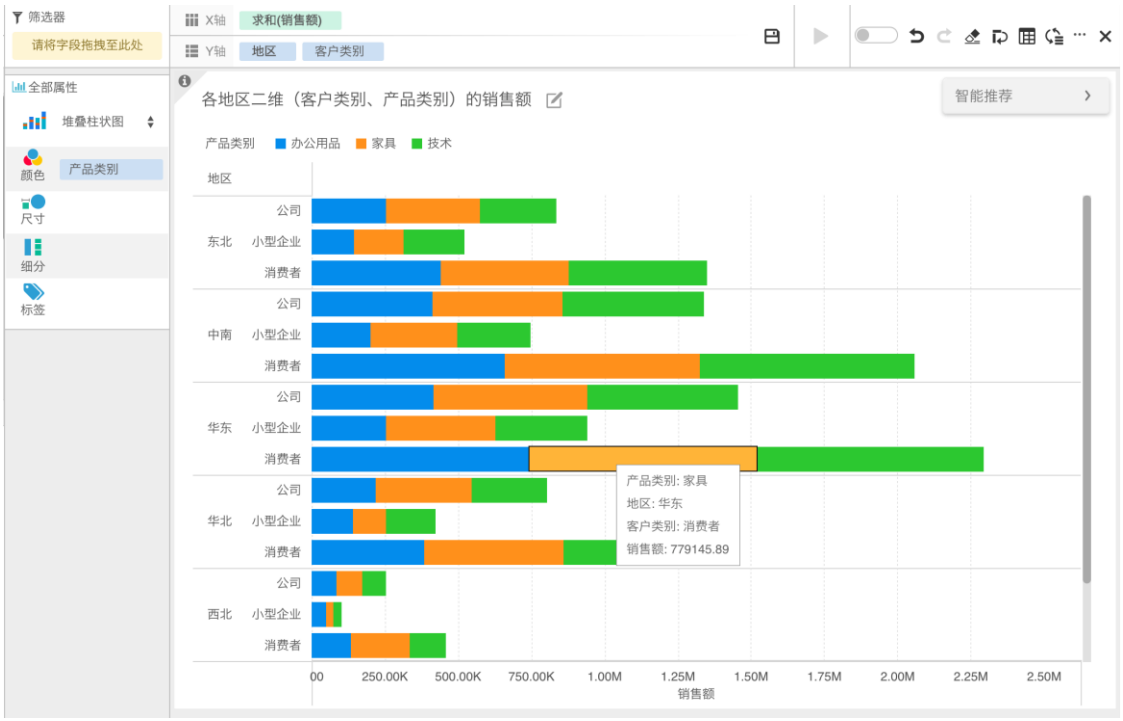

图 5.7 恢复鼠标点击

用户还可以点击图表的空白区域来将整个图表恢复到原始形态。

5.3.3 框选操作

有时用户并不关心某一个数据点的情况，而是关注某一范围内数据点的集合，此时可以通过框选操作来选择。图 5.8 描述了从 2011 年第 1 季度到 2015 年第 1 季度，各个地区的销售额。

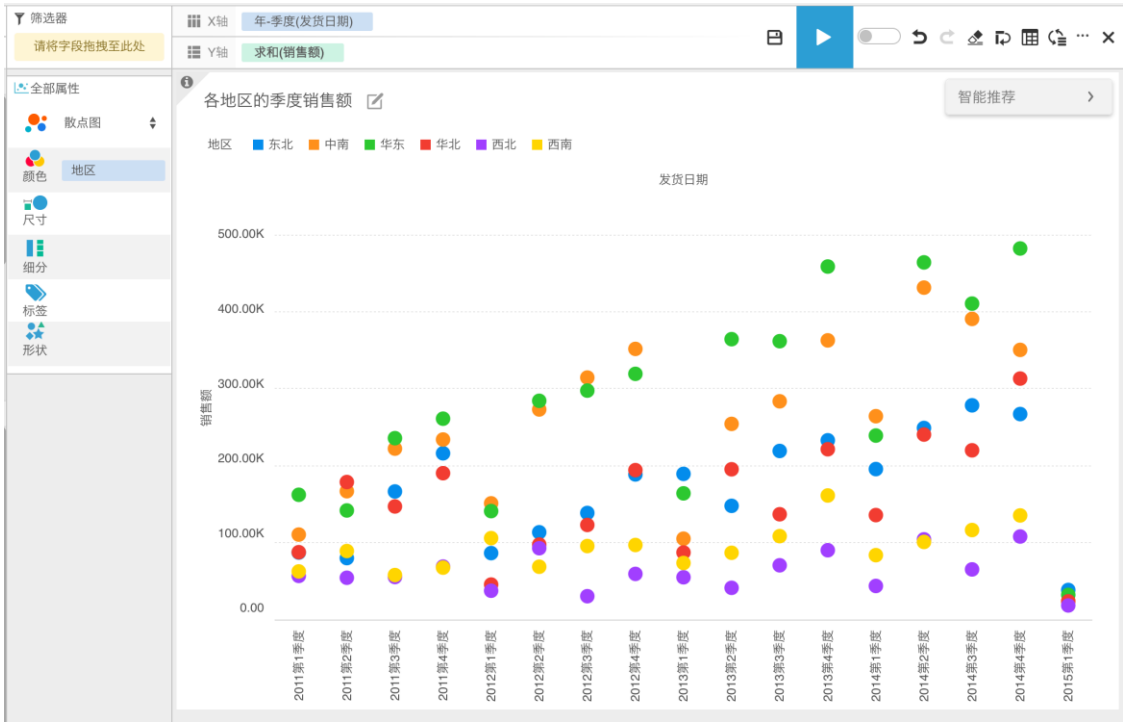

图 5.8 各地区的季度销售图

如果用户想了解 2013 年各个地区销售额的变化趋势，那么用户可以采用框选操作。框选操作示意图如图 5.9 所示，通过鼠标刷动，选择出 2013 年各地区的销售额。

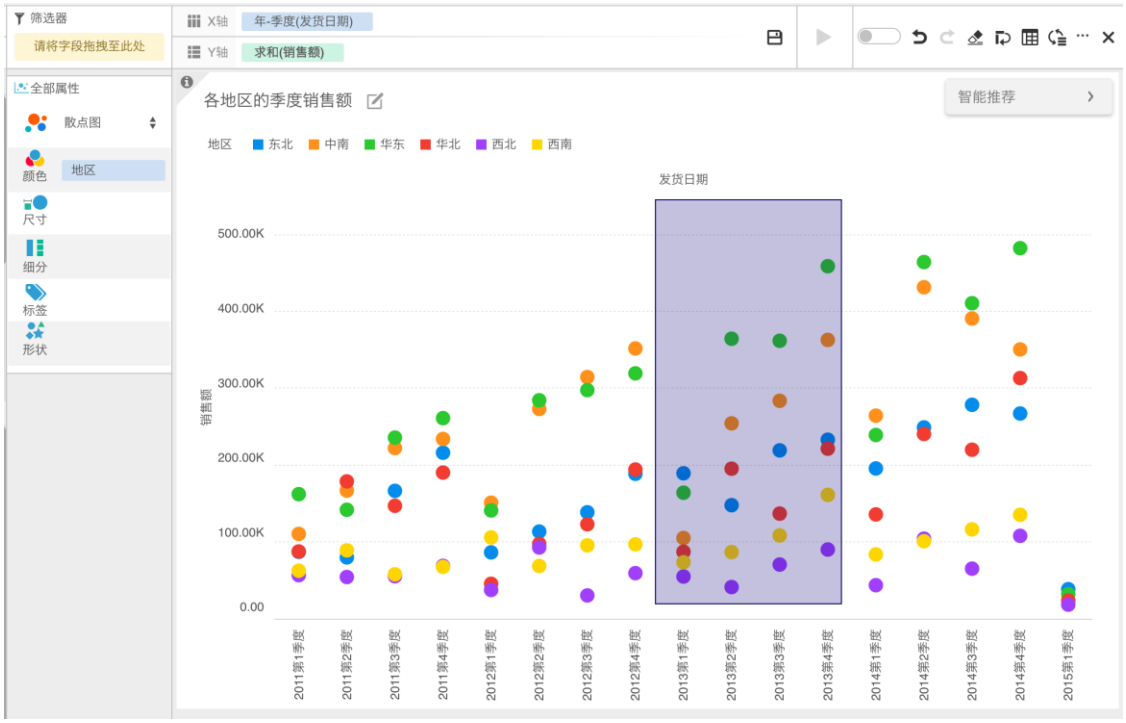

图 5.9 框选操作过程

当框选操作结束后，选中的数据节点集合被高亮显示，其余被置灰，如图 5.10 所示。可以清晰地看出，西北地区在 2013 年中销售额一直最少，销售额在 2013 年中呈现增长趋势的是中南地区、华东地区和西南地区，但是西南地区销售额的增长最慢。

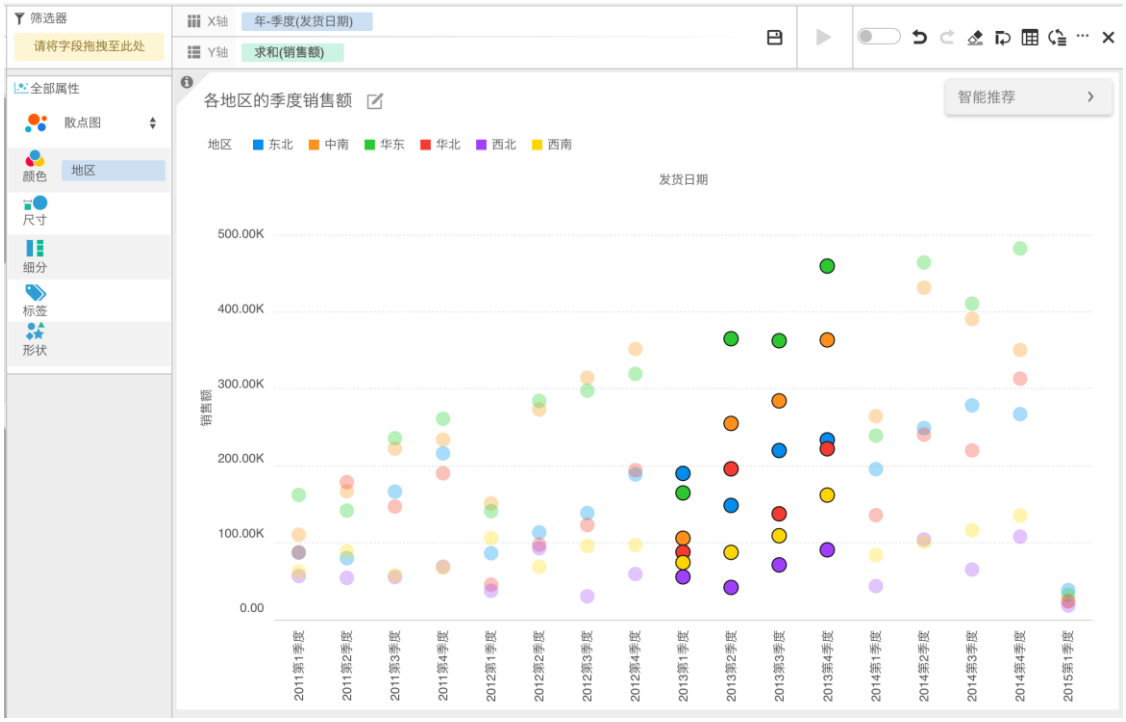

图 5.10 完成框选操作

本系统在实现上述的框选操作时，将框选操作的事件流抽象如下：

```
"streams": [
    {"type": "mousedown", "in": "graph"},
    {"type": "mousedown, mouseup, [mousedown, mouseup] >mousemove",
        "in": "graph"}
 ]
```

mousedown, mouseup, [mousedown, mouseup] > mousemove 这个事件流是在 mousedown 事件、mouseup 事件之间过滤出 mousemove 事件，其含义是在图表的图形区中捕获用户的拖动行为。框选操作是在图表的图形区发生 mousedown 事件流的基础上，再与图形区产生的拖动事件流进行组合而产生的新事件流。

用户通过点击图形区任何位置能够取消框选操作。

5.4 过滤操作实例

5.4.1 通过图例过滤

若用户想要观察东北地区从 2011 年到 2015 年时间内，销量额的变化趋势，此时可以通过图例来完成对某一类目的过滤。如图 5.11 所示，通过点击东北地区图例，东北地区对应时间范围内的销量额变化趋势被高亮显示，此时其他地区的数据节点被置灰。

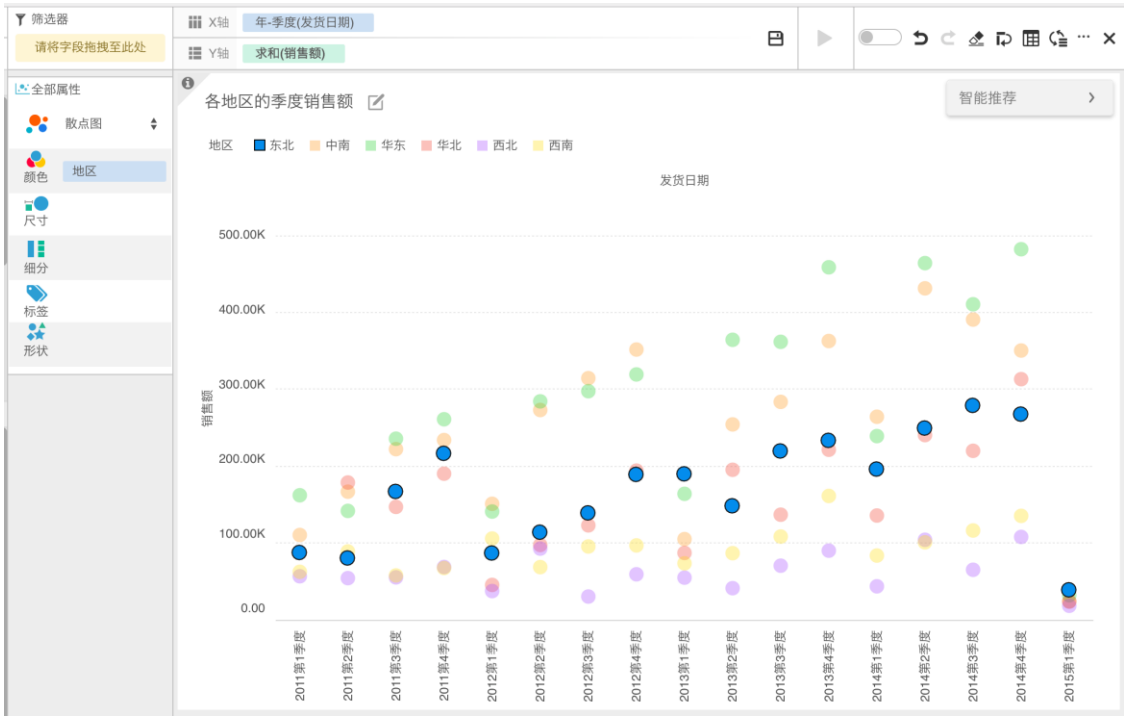

图 5.11 通过离散型图例过滤

由图 5.11 看出，通过选中离散型图例用户能够观察某一类数据集。在本系统中，点击离散型图例进行过滤的事件流表达式被描述为：

```
"streams": [
    {"type": "mousedown", "in": "legend"},
    {"type": "mousedown", "in": "graph"}
]
```

通过在图例中捕获 mousedown 事件流来得到过滤数据集的条件，事件处理模型针对过滤条件设置图元的状态，满足条件的数据集被高亮显示，其余被置灰。本系统可以在图例区或者图形区，通过捕获 mousedown 事件流来实现取消选中的操作。

用户还可以通过拖动连续型图例来设置数据范围。如图 5.12 所示，折扣值被映射为散点的半径。用户拖动图例设置折扣范围是 10%到 20%时，可以看出折扣值在该范围内的数据集被高亮显示，其余被置灰。

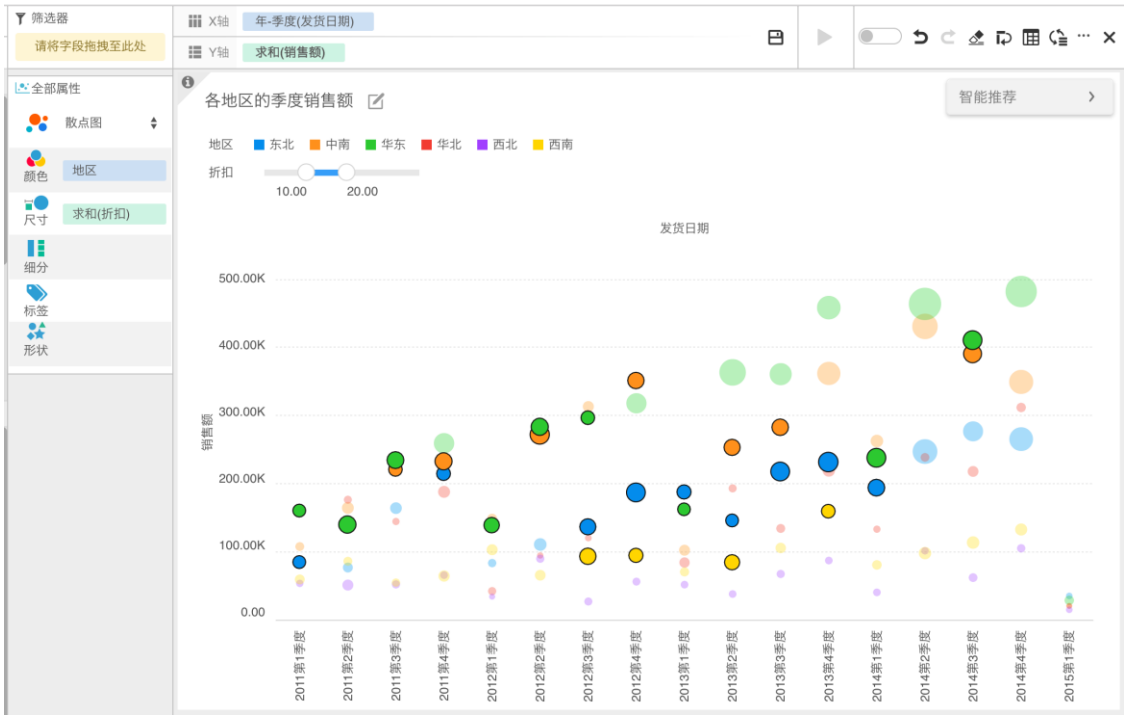

图 5.12 通过连续型图例过滤

通过连续型图例进行范围过滤的事件流表达式为：

```
"streams": [
    {"type": "mousedown", "in": "legend"},
    {"type": "mousedown, mouseup, [mousedown, mouseup] > mousemove",
        "in": "legend"},
    {"type": "mousedown", "in": "graph"},
]
```

在连续型图例区捕获到 mousedown 事件流，然后捕获到拖动图例两端的选择器的拖动事件流，进而控制数据的筛选范围；在拖动事件发生过程中，选中范围内的数据节点被实时高亮显示，数据范围之外的数据节点被置灰。当想要取消这个范围过滤操作时，通过捕捉图例区的 mousedown 事件流或者图形区的 mousedown 事件流。

在地图中，也可以通过连续型图例的选择，来方便筛选出关心的数据集。如题 5.13 所示。

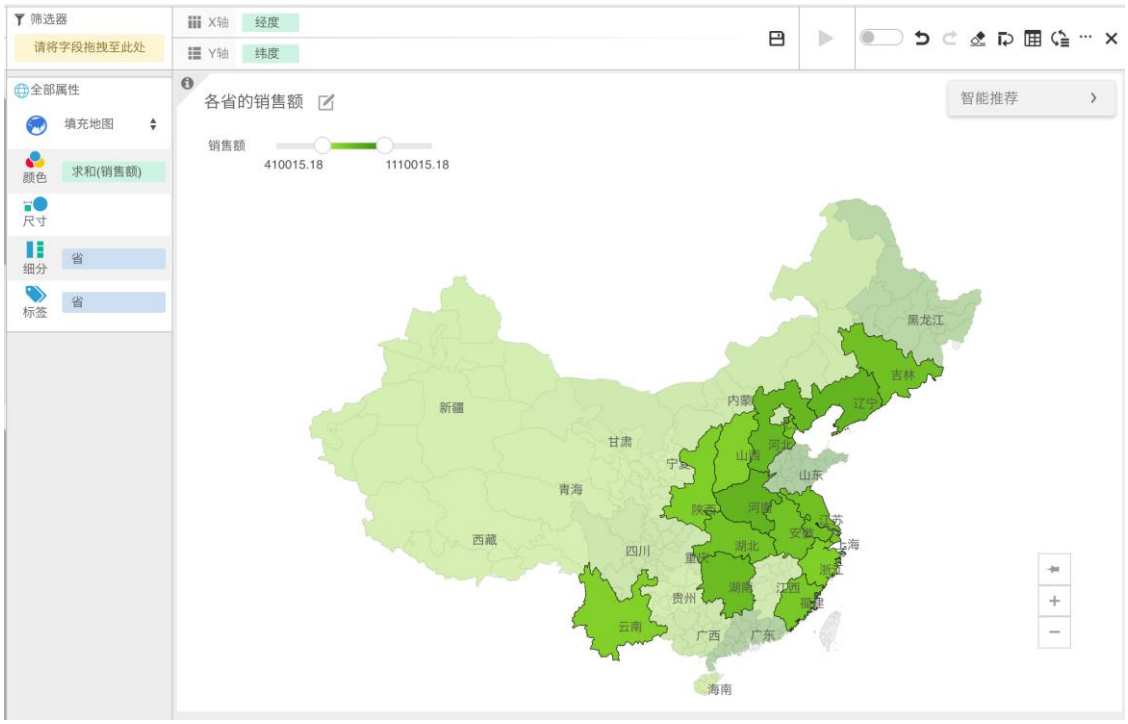

图 5.13 地图中的过滤操作

5.4.2 通过透视轴过滤

在透视图表中，还可以通过透视轴来对某一行或者某一列数据集合进行选择。这个选择操作的事件流表达式描述为：

```
"streams": [
    {"type": "mousedown", "in": "pivotAxis"},
    {"type": "mousedown", "in": "graph"},
```

]

通过在透视轴区捕获 mousedown 事件流来作为一类数据子集的筛选条件，并且在透视轴区以及图表区都可以取消这个筛选操作，也是通过捕捉 mousedown 事件流来实现的。

如图 5.14 所示，选择出中南地区所有客户类别的产品销售额，选中的数据集被高亮显示，其余的数据集通过置灰处理，来突出显示选中的数据子集。

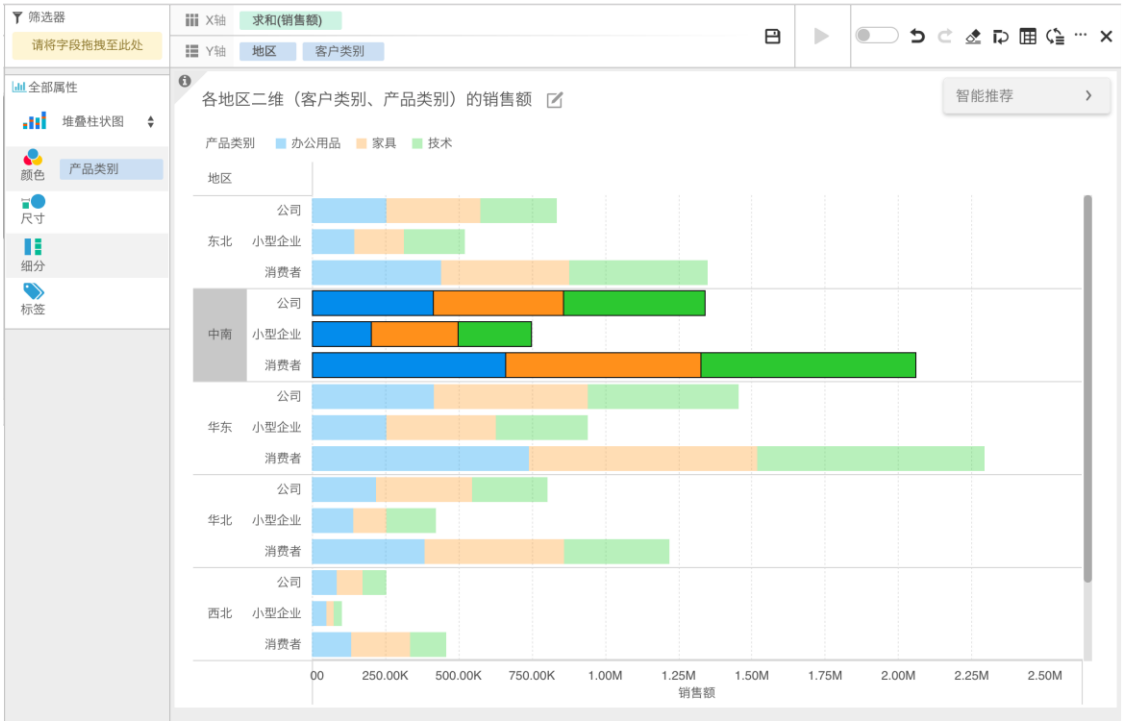

图 5.14 通过透视轴过滤

5.5 导航操作实例

本文描述的数据可视化系统主要实现了导航交互技术中的平移和缩放功能。平移和缩放的交互操作应用用在地图中。图 5.15 展现了没有发生导航交互时，地区分布图的显示效果。

图 5.15 中国地区分布图

5.5.1 平移操作

对于地图的平移交互操作，声明式的表达式如下：

```
"streams": [
    {"type": "mousedown", "in": "graph"},
    {"type": "mousedown, mouseup, [mousedown, mouseup] > mousemove",
        "in": "graph"}
]
```

在图形区捕捉拖动事件流，整个图形区会随着鼠标的移动而进行左右移动。图 5.16 描述了地区分布图随着鼠标向右方向平移。

图 5.16 平移交互图

5.5.2 缩放操作

对于缩放交互操作，事件流的描述表达式声明如下：

```
"streams": [
    {"type": "mousewheel", "in": "graph"}
]
```

在图形区捕捉鼠标滚动事件流，整个图形区会随着鼠标的滚动而放大或者缩小，滚动的方向决定了是放大操作还是缩小操作。地图放大交互示意图详见图 5.17，缩小交互示意图详见图 5.18。

图 5.17 放大交互图

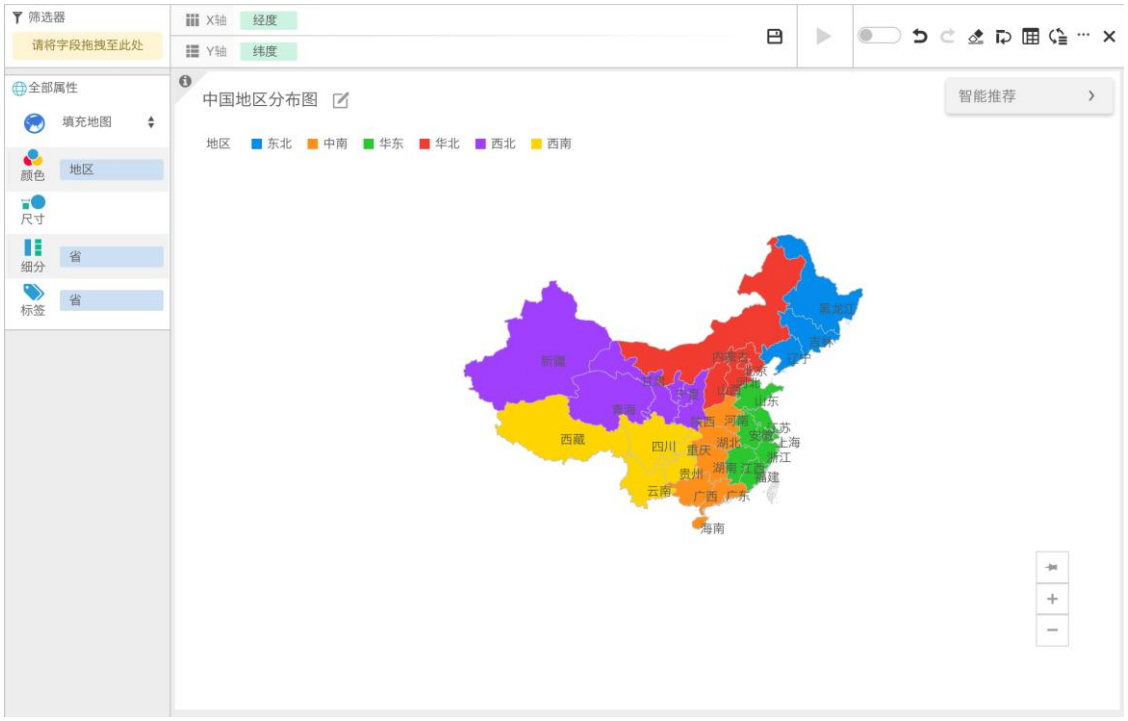

图 5.18 缩小交互图

5.6 用户自定义样式实例

在没有用户自定义样式的情况下，图表的样式来自于主题模型的默认值。图 5.19 描述了每个地区与客户类别之间的饼图，通过标签来显示销售额。图 5.19 中的文本、轴线的样式都来自于主题模型的设置。

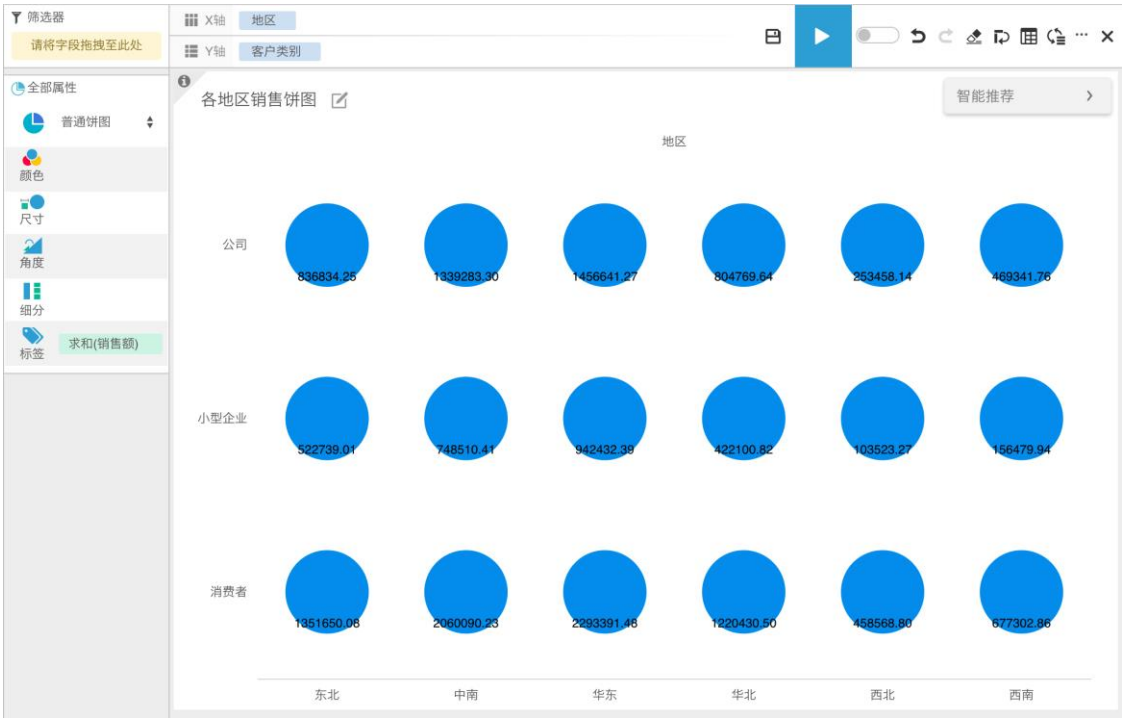

图 5.19 默认样式示意图

用户可以通过属性面板来自定义图表样式。图 5.20 显示用户选择了浅绿色（黑色框选中部分），用户也可以自定义标签的颜色、大小等样式，如图 5.21 所示。

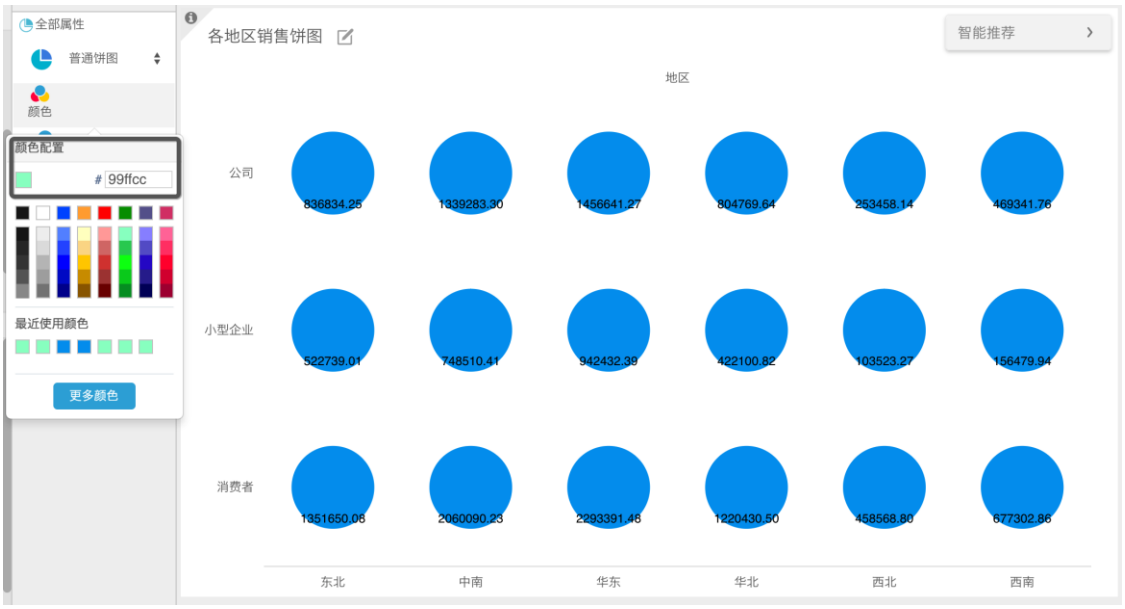

图 5.20 自定义颜色示意图

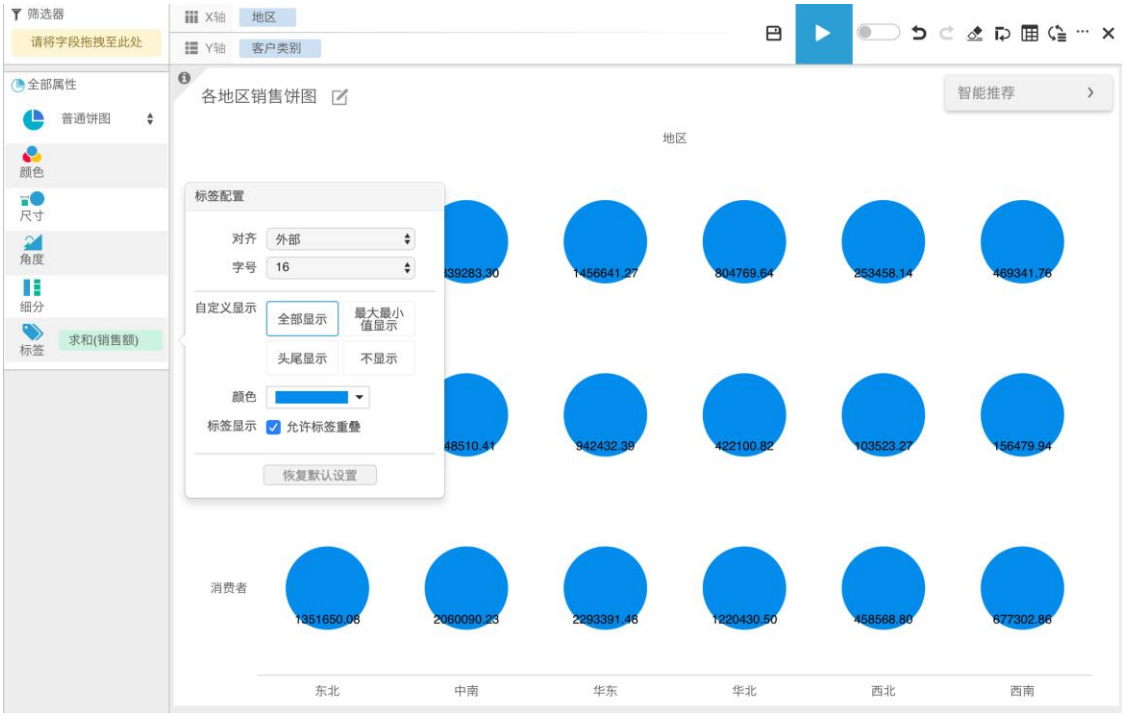

图 5.21 自定义标签属性示意图

通过属性面板，用户自定义了颜色以及标签的样式，生成用户期望的个性化图表详见图 5.22。

图 5.22 自定义颜色和标签的效果图

为了实现图 5.22 的自定义效果，主题模型中自定义样式的声明式描述如下：

```
"theme": {
    "visualScales": {
        "color0": {
            "visualType": "color",
            "domainType": "any",
            "range": {"r": 153,"g": 255,"b": 204,"a": 1}
        }
    },
    "classStyle": {
        "mark78": {
            "label": {
                "styles": [{
```

```
                    "font": {"size": 16},
                    "fillStyle": {"r": 13,"g": 161,"b": 237,"a": 1}
                }],
                "position": "outer"
            },
            "mark": {"color": "color0"}
        }
    }
}
```

系统通过"color0"字段指定用户自定义标记的颜色，通过标记的"label"字段自定义标签的样式。

用户可以自定义与数据相关的属性。如图 5.23 所示，用户通过颜色区别不同的产品类别，通过饼图半径的长短表示利润值的大小，通过角度的大小以反应销售额的多少。用户可以从图 5.23 看出中南地区的利润是最大的。

图 5.23 自定义与数据相关的视觉属性效果图

如图 5.24 所示，用户可以通过绑定颜色、尺寸和角度与数据字段之间的映射关系，以达到绘制玫瑰图的目的。其中，用户通过颜色区分不同的地区；把销售额映射为扇形的半径，半径的长短表示销售额的多少；将利润映射为扇形的角度，利润值越大，角度值也越大。另外，用户通过自定义标签以显示各地区的销售额与利润值。由图 5.24 可以看出，西北地区的销售额与利润值都是最少的；华东地区销售额是最多的。

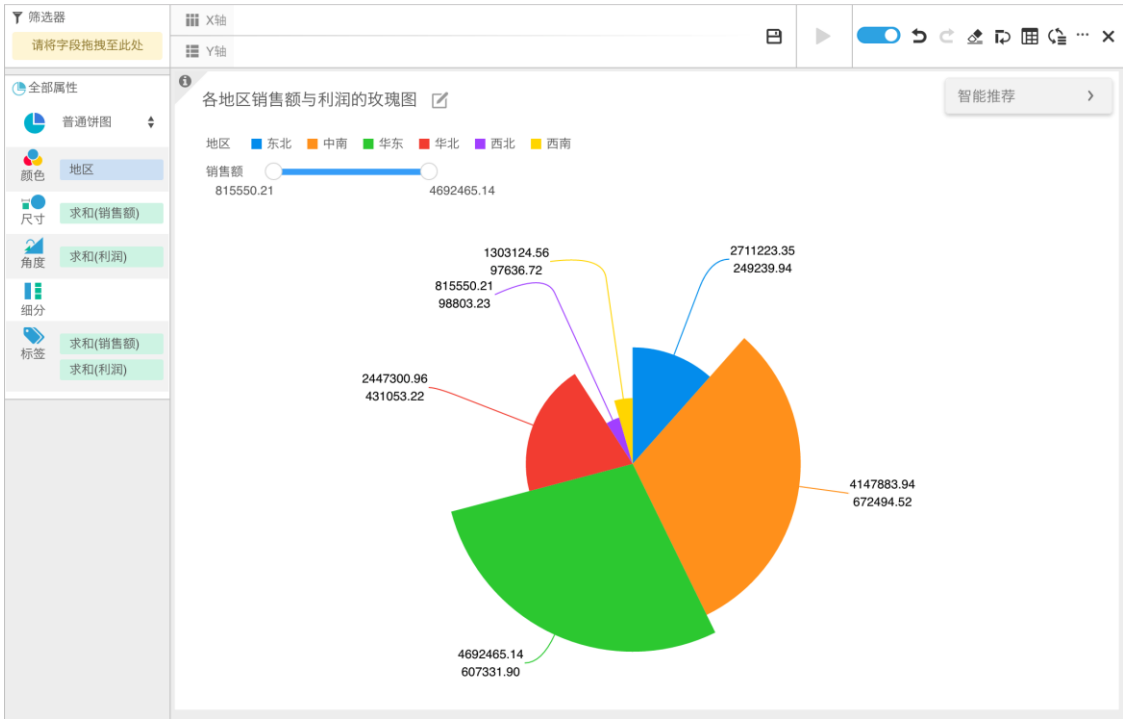

图 5.24 自定义视觉属性实现玫瑰图

为了实现图 5.24 呈现的玫瑰图，NEV 在声明标记时，标记核心字段的描述如下：

```
"mark": {
        "type": "pie",
        "color": "地区",
        "angle": "利润",
        "size": "销售额",
        "className": "mark2"
```

```
}
```

主题模型的声明式描述如下：

```
"theme": {
    "visualScales": {
        "color0": {
            "visualType": "color",
            "domainType": "discrete"
        },
        "size1": {
            "visualType": "size",
            "domainType": "continuous"
        }
    },
    "legends": {
        "color0": {},
        "size1": {}
      },
      "classStyle": {
        "mark2": {
          "mark": {
            "color": "color0",
            "size": "size1"
          }
        }
    }
}
```

用户自定义样式的细节越多，主题模型的声明式描述会越复杂。为了免除用户描述冗长的样式定制细节，网易有数提供了可视化属性面板，为用户自定义样

式提供了友好的入口，方便用户通过自定义视觉属性以探索数据中蕴含的信息，保证用户能够从多个维度观察数据。

5.7 本章小结

本章节介绍了 NEV 数据可视化系统在网易有数项目中的应用，展示了系统中实现的交互技术。另外，本章还展示了在主题模型的支持下，通过可视化属性面板的配置，以实现高度定制化的可视化图表。

第6章 总结与展望

6.1 本文的工作总结

本文在研究和分析了数据可视化技术的基础上，以数据到可视化图表的呈现和描述并实现用户的交互两个方面作为切入点，设计并实现了数据可视化系统，主要有以下几点工作与贡献：

（1） 在数据到可视化图表的呈现方面，本文基于图形语法提出了一套图形语言来描述图表的结构，同时完成了图表构建与渲染，有效地呈现了数据中几何、拓扑和形状特征，实现了数据到可视化图形的构建。

（2） 在图表样式设计方面，本文提出了主题模型来统一描述图表的样式，并且提供了简单方便、用户友好的操作接口来完成用户自定义样式的定制，来满足用户对于图表样式的定制化需求，为用户探索数据信息做好准备。

（3） 在声明用户交互方面，本文针对当前图表库对于用户交互描述的空缺，提出了声明式式交互语法来描述交互操作。这种描述方式与图形语法描述图表结构类似，通过使用类似于自然语言的交互语法，使用户直观、方便地表达期望的交互操作，极大地减轻了用户描述交互的压力。并且用户不需要关注交互操作的具体实现，从而摆脱了事件处理过程中的状态维护和细节处理，因此改善了用户体验。同时声明式交互语法非常强大，用户可以定义其个性化的交互操作，来探索可视化图表中数据背后的意义。

（4） 在实现用户交互方面，本文提出了事件流模型来建模发生在图表中的各种事件，通过事件流的组合能力来构建满足用户交互需求的事件流。同时，本文提出了事件处理模型来处理各种事件流，来实现用户的交互操作，并将交互结果反馈给用户，极大地方便用户观察探究数据的含义。

6.2 未来研究工作

在 NEV 数据可视化系统中，已经实现的用户交互模型在以下几个方面还存在改进的空间。

（1） 如何动态地改变可视化图形背后的图形结构、数据流结构来作为用户交互的基础，是本文交互模型在实现方面需要继续探索努力的方向。

（2） 通过声明式的交互语法如何友好地描述用户期望的交互结果，仍然需要探索研究。

（3） 本文在实现交互过程中，在交互延迟方面没有进行相关的优化处理，当数据量非常大时，本文提出的交互模型产生的交互延迟时间会超过用户可以接受的预期，因此提高交互响应的性能仍需要继续探索。

在信息时代，数据的重要性日益凸显出来，对这些数据的可视化探索越来越重要，用户交互作为探索的重要手段，其在未来还有大量研究需要继续深入下去。希望在不久将来，用户交互的描述与实现在学术界以及工业界都会有质的飞跃。

参考文献

[1] 三维数据场可视化. 北京: 清华大学出版社，1999.

[2] Cleveland W S. Visualizing data[M]. Hobart Press, 1993.

[3] Haber R B, McNabb D A. Visualization idioms: A conceptual model for scientific visualization systems[J]. Visualization in scientific computing, 1990, 74: 93.

[4] Keim D A, Mansmann F, Schneidewind J, et al. Challenges in visual data analysis[C]//Tenth International Conference on Information Visualisation (IV'06). IEEE, 2006: 9-16.

[5] Cammarano M, Dong X, Chan B, et al. Visualization of heterogeneous data[J]. IEEE Transactions on Visualization and Computer Graphics, 2007, 13(6): 1200-1207.

[6] Pike W A, Stasko J, Chang R, et al. The science of interaction[J]. Information Visualization, 2009, 8(4): 263-274.

[7] Heer J, Shneiderman B. Interactive dynamics for visual analysis[J]. Queue, 2012, 10(2): 30.

[8] Heer J, Card S K, Landay J A. Prefuse: a toolkit for interactive information visualization[C]//Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 2005: 421-430.

[9] Wilkinson L. The grammar of graphics[M]. Springer Science & Business Media, 2006.

[10] Vega: A Visualization Grammar. http://trifacta.github.io/Vega, April 2014.

[11] Bostock M, Ogievetsky V, Heer J. D^3 data-driven documents[J]. IEEE transactions on visualization and computer graphics, 2011, 17(12): 2301-2309.

[12] Bostock M, Heer J. Protovis: A graphical toolkit for visualization[J]. IEEE transactions on visualization and computer graphics, 2009, 15(6): 1121-1128.

[13] Yi J S, ah Kang Y, Stasko J, et al. Toward a deeper understanding of the role of interaction in information visualization[J]. IEEE transactions on visualization and computer graphics, 2007, 13(6): 1224-1231.

[14] Myers B A. Separating application code from toolkits: eliminating the spaghetti of call-backs[C]//Proceedings of the 4th annual ACM symposium on User interface software and technology. ACM, 1991: 211-220.

[15] Liu Z, Stasko J. Mental models, visual reasoning and interaction in information visualization: A top-down perspective[J]. IEEE transactions on visualization and computer graphics, 2010, 16(6): 999-1008.

[16] Alipay G2[EB/OL]. http://g2.alipay.com/.

[17] Stolte C, Tang D, Hanrahan P. Polaris: A system for query, analysis, and visualization of multidimensional relational databases[J]. IEEE Transactions on Visualization and Computer Graphics, 2002, 8(1): 52-65.

[18] Wickham H. A layered grammar of graphics[J]. Journal of Computational and Graphical Statistics, 2010, 19(1): 3-28.

[19] Bainomugisha E, Carreton A L, Cutsem T, et al. A survey on reactive programming[J]. ACM Computing Surveys (CSUR), 2013, 45(4): 52.

[20] Elliott C, Hudak P. Functional reactive animation[C]//ACM SIGPLAN Notices. ACM, 1997, 32(8): 263-273.

[21] Hudak P. Functional reactive programming[C]//European Symposium on Programming. Springer Berlin Heidelberg, 1999: 1-1.

[22] Czaplicki E, Chong S. Asynchronous functional reactive programming for GUIs[C]//ACM SIGPLAN Notices. ACM, 2013, 48(6): 411-422.

[23] Heer J, Bostock M. Declarative language design for interactive visualization[J]. IEEE Transactions on Visualization and Computer Graphics, 2010, 16(6): 1149-1156.

[24] Satyanarayan A, Wongsuphasawat K, Heer J. Declarative interaction design for data visualization[C]//Proceedings of the 27th annual ACM symposium on User interface software and technology. ACM, 2014: 669-678.

[25] Satyanarayan A, Russell R, Hoffswell J, et al. Reactive vega: A streaming dataflow architecture for declarative interactive visualization[J]. IEEE transactions on visualization and computer graphics, 2016, 22(1): 659-668.

[26] Amar R, Eagan J, Stasko J. Low-level components of analytic activity in information visualization[C]//IEEE Symposium on Information Visualization, 2005. INFOVIS 2005. IEEE, 2005: 111-117.

[27] Fekete J D, Plaisant C. Excentric labeling: dynamic neighborhood labeling for data visualization[C]//Proceedings of the SIGCHI conference on Human Factors in Computing Systems. ACM, 1999: 512-519.

[28] Keim D A. Information visualization and visual data mining[J]. IEEE transactions on Visualization and Computer Graphics, 2002, 8(1): 1-8.

[29] Chen H. Compound brushing [dynamic data visualization][C]//Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on. IEEE, 2003: 181-188.

[30] Moscovich T, Chevalier F, Henry N, et al. Topology-aware navigation in large networks[C]//Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2009: 2319-2328.

[31] Yi J S, Melton R, Stasko J, et al. Dust & magnet: multivariate information visualization using a magnet metaphor[J]. Information Visualization, 2005, 4(4): 239-256.

[32] Wan Z, Taha W, Hudak P. Event-driven FRP[C]//International Symposium on Practical Aspects of Declarative Languages. Springer Berlin Heidelberg, 2002: 155-172.

[33] NetEase Youdata[EB/OL]. https://youdata.163.com/.

攻读硕士学位期间主要的研究成果

致谢

时光荏苒，浙大求是园两年半的研究生生涯即将结束，浙里有我留恋的青春和记忆。即将离开之际，感谢求学路上所有给予我帮助的老师、同学。

本论文从选题到完成是在实验室导师的悉心指导下完成的。导师渊博的专业知识，严谨的治学态度，精益求精的工作作风，诲人不倦的高尚师德，朴实无华、平易近人的人格魅力对我影响深远。不仅使我树立了远大的学术目标、掌握了基本的研究方法，还使我明白了许多待人接物与为人处事的道理。在此谨向导师组表示崇高的敬意和感谢。

感谢网易有数开发团队，使得 NEV 有一个方便简洁的使用平台。感谢 NEV 小组的同事，他们在系统实现过程中给予了很多帮助。

感谢朱一聪、陈鸿翔、李幸超、唐思、姚雨程、何平、刘博文、林秋霞等师兄师姐在学习、科研项目和生活等各方面为我提供的帮助。尤其感谢唐晓瑜师姐在学习生活中的指导与帮助。感谢同届的吴联坤、冯杰、任伟超、王伟迪、张静恬、王改革、李邦鹏、刘伟、周俊林、朱清华、朱华、钱宇和吴晓晓，感谢你们在学习生活中提供的帮助。十分感谢于志超、金明健和胡凡在梳理论文过程中的鼓励与支持。

特别感谢我的父母、兄长对我无怨无悔地付出，和一如既往鼓励与支持，是你们无私的爱让我无后顾之忧完成学业；感谢你们包容我的任性，是你们的宽容让我有底气自由翱翔。感谢我的室友郭凌子和蔡文心，你们的包容与陪伴，让我的研究生生活绚丽多彩。

最后，感谢评阅、评议论文和答辩委员会的各位专家学者在百忙的工作中能给予指导。

张也

2017 年 1 月 5 日