基于 XML 矢量图形 SVG 应用的软件体系结构研究

袁家政[1),2)] 须 德[1)] 鲍 泓[2)]

[1)](北京交通大学计算机与信息技术学院，北京 100044) [2)](北京联合大学信息技术研究所，北京 100101)

摘 要 在单个 SVG 文档的模块化与结构化的基础上，分析了 SVG 实体的特征和组织方式，描述了 SVG 实体、对象、类的结构组成，提出了基于 SVG 应用的软件开发的体系结构 SSA 和系统框架设计。在 SSA 中，设计了一个转换算法，算法针对大量的 SVG 实体进行分类、提取 SVG 图元，并将 SVG 图元存储在关系数据库中，解决了包含大量 SVG 实体的应用系统存在的运行速度的问题、SVG 组件的数据重用问题和基于数据库系统的 SVG 开发问题。设计了一个基于 SSA 的数字文物导航平台系统，通过该系统与传统的基于 Html/Image 的文物展现模式的比较，验证了 SSA 开发模式的有效性。

关键词 可伸缩矢量图形 软件体系结构 图元 数据模型

中图法分类号：TP391.41 **文献标识码**：A **文章编号**：1006-8961(2007)04-0718-08

Study of SVG Software Architecture Based on XML

YUAN Jia-zheng[1),2)], XU De[2)], BAO Hong[2)]

[1)](*School of Computer & Information Technology, Beijing Jiaotong University, Beijing* 100044)

[2)](*Institute of Information Technology, Beijing Union University, Beijing* 100101)

Abstract According to the modularized SVG Entity feature and framework, we described a relationship among SVG entity, SVG class and SVG object, and presented a software architecture for a software development based on SVG application(called SSA). In SSA, we designed a transformation algorithm to implement very large SVG entity classification; then some SVG meta-graphs were extracted from SVG entity, SVG class and SVG object and saved to RDBMS. The algorithm has solved three problems: slow execution speed in SVG application system including very large SVG entities, data reuse issue for SVG component and SVG development issue for based on RDBMS. We designed an explorer platform system of digital museum based on SSA, and compared with traditional digital museum development model based on Html/Image. Experimental results showed the effectiveness of our SSA and goods performance for SVG application development.

Keywords scalable vector graphics, software architecture, meta-graph, data model

1 引 言

在 Internet 中，SVG[1](scalable vector graphics，可伸缩矢量图形)的出现为 WWW 提供了一个全新的基于 XML(eXtensible Markup Language，可扩展的标识语言)特性的 2 维网络图像格式。但是在实际应用中，SVG 的模块设计功能很差，需要用户自己来组织和设计。目前 SVG 的理论体系研究文献较少，大多数研究人员对 SVG 的应用和研究还处于起步阶段，文献[2]~[4]仅仅介绍了 SVG 的特点和应用方向，而文献[5]、[6]提到了 SVG 在 WebGis 和数字文物中的应用，但并未解决结构化的问题，文献[3]、[7]、[8]提到了 SVG 的网络支撑平台的体

基金项目：教育部科学技术研究重点项目(2002KJ124)；北京市优秀人才培养资助项目(20051D0502206)

收稿日期：2006-09-15；**改回日期**：2006-12-21

第一作者简介：袁家政(1971~)，男，副教授，CCF 会员。现为北京交通大学计算机应用技术专业博士研究生。主要研究领域为 XML、多媒体数据库、网络与信息安全。E-mail：xxtjiazheng@buu.com.cn；jzyuan@sohu.com

系结构问题,但未解决 SVG 应用的软件开发体系结构[9]问题。基于 SVG 应用的软件体系结构和数据模型的提取,在当前的国内外研究还是个空白。为此,在文献[10]、[11]中,提出了基于单个 SVG 图形文档的一种模块化和结构化组成格式,解决了单个文档结构化问题,本文继续结合现实世界的物体、模块化的 SVG 图形图像结构和一般软件体系结构的特点,挖掘 SVG 的优点,提出了一种实现基于 SVG 图形应用软件体系结构 SSA(SVG-based software architecture)和数据模型,并利用数字文物博物平台的开发得到了验证和应用。

2　SVG 实体的数据模型与表示

通常为了更好地处理客观实体,必须将实体进行形式化描述,并将实体按照有利于 SVG 的表述方式加以组织。在文献[10]中,将一个实体 E 看作一个对象 *Object*,描述为对于 $\forall E \in Object$,则 $E=(A, O)$,其中,$A=(a_1, a_2, \cdots, a_m)$,$O=(o_1, o_2, \cdots, o_n)$,$A$ 代表属性集合,O 代表对象 *Object* 集合,m、n 为整数。为了更好地使用 SVG 文档来描述客观实体,必须将 SVG 结构化,了解 SVG 实体中各个对象的关系,使其适应层次化的客观实体。

2.1　SVG 实体类的关系和结构组成

在现实世界中,将实体分为简单实体和复杂实体,其中简单实体根据其几何特征进一步分为点状实体、线状实体和面状实体等 3 种类型,而复杂实体则由多个简单实体进行组合或者仿射变换构成。也就是说,矢量化了的图形整体作为对象,本身包含各种属性和特征;组成图形的各个部分,又可以看成独立的对象,也具有自身的属性和特征;同时这些对象之间具有一定的关系,所有这些属性和关系,就构成了完整的图形描述。SVG 矢量图形主要从物体的形状和颜色属性特征来表述实体,不同的图形实体,属性特征不同。

根据实体的描述和 SVG 文档标准[1]综合分析,实体转换为计算机中的 SVG 图形对象类时,图形对象的类结构可用下列集合表达式表达:

SVG 类 = {〈文本类〉,〈图形类〉,〈图像类〉};

图形类 = {〈基本图形类〉,〈组合图形类〉,〈复杂图形类〉};

基本图形类 = {〈Pixel〉,〈Line〉,〈Polyline〉,〈Rectangle〉,〈Circle〉,〈polygon〉,…};

组合图形类 = {〈图形类〉,[〈图形类〉]};

复杂图形类 = {〈图元类进行仿射变换〉};

图元类 = {〈基本图元类〉,〈组合图元类〉,〈属性类〉};

属性类 = {〈Color〉,〈stroke-width〉,〈fill〉},…}

SVG 图形对象的类结构及构成可用图 1 来表示。在图 1 中,SVG 图形对象最基本的要素是形状 $e(B)$、色彩 c、其他基本属性(attribute),构成了 SVG 基本图元类、组合图元类和属性类,图元类通过组合或仿射变换生成图形,基本图形类(Basic metaGraph_Class)、组合图形类(Combination metaGraph_Class)和复杂图形类(Complex metaGraph_Class)通过组合生成 SVG 类或实体。

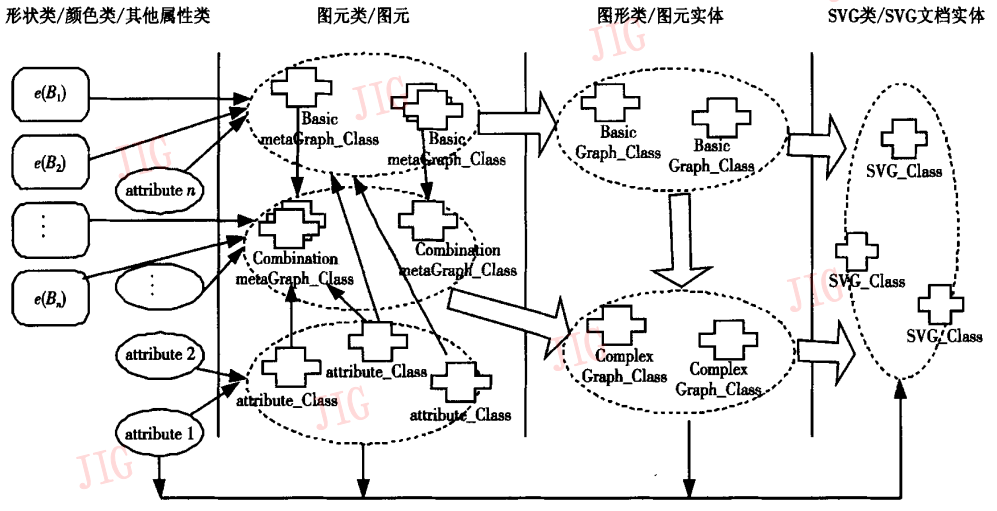

图 1　SVG 实体关系的结构图

Fig. 1　A structure model about relations of SVG entities

2.2 SVG 应用系统中的数据模型与数据表示

为了使 SVG 的应用开发更为简洁和便捷，必须充分利用 SVG 图形图像类的结构特点和 SVG 文档的优点，将其模型化。使用文档模型 DOM 将单个的 SVG 文档模型化，DOM 包含两个关键的抽象概念：一个是树状的层次结构；另一个是用来表示文档内容和结构的节点集合。在 SVG 文档的树状层次中包括了所有节点，这些节点是 SVG 文档的各个元素及属性名，节点本身也可以包含其他的节点。这样的好处是可以通过这个层次结构找到并修改某一特定节点的信息，具体实例如图 2 的一个 SVG 文档实例，描述了一幅喀麦隆国旗，图 3 是图 2 文档的层次树结构模型。

```
〈? xml version = "1.1"?〉
〈! DOCTYPE svg PUBLIC " -//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"〉
〈svg width = "800" height = "800"〉
  〈title〉flag〈/title〉
  〈desc〉This is a flag of Cameroon.〈/desc〉
  〈defs〉〈rect id = "ComRect" width = "50" height = "100"/〉〈/defs〉
  〈text x = "25" y = "190" fill = "blue" font - family = "Verdana" font - size =
"10"〉This is a flag of Cameroon.〈/text〉
  〈g id = "flagobject"〉
    〈title〉Red Rectangle〈/title〉
    〈desc〉This is a Rectangle Shape and Red color of flag.〈/desc〉
    〈use x = "68" y = "70" fill = "Red"  xlink:href = "#ComRect" /〉
  〈g id = "pentagram"〉
      〈title〉Yellow pentagram〈/title〉
      〈desc〉This is a Pentagram Shape and Yellow color of flagobject.〈/desc〉
      〈polygon fill = "Yellow" points = "94,109 96,115 102,115 97,119 99,
125 94,121 89,125 91,119 86,115 92,115"/〉
    〈/g〉
  〈/g〉
  ……  ……
〈/svg〉
```

图 2 SVG 文档实例

Fig. 2 A cameroon flag with SVG entity description

在基于 SVG 实际应用中，SVG 图形对象是 SVG 应用系统数据结构中最基本的对象，是使用面向对象方法设计 SVG 图形应用系统的基础，在软件开发体系结构 SSA 中，可以将 SVG 图形实体及对象划分为 SVG 文档、Entity 实体、SVG 类、SVG 复杂对象、SVG 基本对象等。由于 SVG 实体大多由基本图元（metaGraph）信息来描述，图元的描述方法是 SVG 应用系统中进行矢量图形描述的重要部分，因为图元代表着实际应用中的具体实体，是矢量图形应用

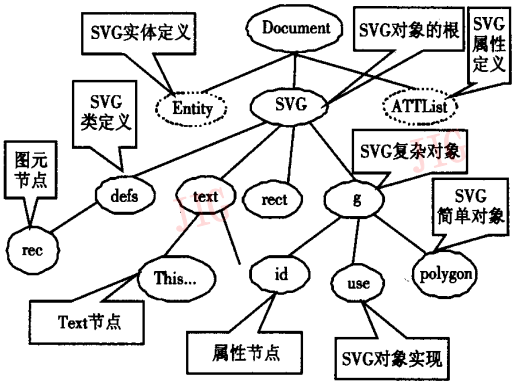

图 3 利用 DOM 将 SVG 文档模型化

Fig. 3 A DOM for fig. 2

系统中最为关键的对象。图元在 SVG 中的表示必须既包含几何形状的信息，又要包含实际的应用属性信息。根据这些要求，在文献[9]的基础上进行修改和总结，图元可以用 BNF 描述表示如下：

〈SVG 图元〉::=〈SVG 形状类定义〉[〈颜色属性定义〉〈应用属性定义〉〈交互事件定义〉]

〈SVG 形状类定义〉::=〈SVG 实体形状〉[〈color〉〈fill〉〈 stroke-width〉〈transform〉〈animation〉]

〈SVG 实体形状〉::=〈SVG 基本图形〉[〈 SVG 基本图形〉]

〈SVG 基本图形〉::=〈SVG 基本图形类〉[〈coordinate〉〈width〉〈height〉]

〈应用属性定义〉::=〈color〉〈 应用属性〉[〈应用属性〉]

〈应用属性〉::={〈name〉=〈value〉}

〈交互事件定义〉::=〈交互事件〉[〈交互事件〉]

〈交互事件〉::={〈name〉=〈函数响应〉}

其中，color、fill、stroke-width、transform、animation、coordinate、width、height 分别表示 SVG 实体中的颜色、填充样式、线形宽度样式、仿射变换参数、动画动作参数、实体坐标、实体宽度和实体高度等属性，而 name、value 表示属性名和值。

根据前面的描述，图元的 SVG 表示由几何形状、颜色属性、应用属性和交互事件组成。图元的几何形状可以表示为图元所包含的基本图形的组合，即基本的图形通过组合、仿射变换组成复杂的图元。

3 基于 SVG 的软件体系结构的设计

在基于 SVG 的实际应用系统开发中，SVG 主要的优势在于其基于文本的矢量图形图像格式和有无

限精度的分辨率,主要用于开发基于 Web 的动态、可缩放与平台无关的图像图形内容表现和交互的应用系统。然而大型的应用系统不是一个 SVG 文档所能解决的,往往需要无数个 SVG 的组合和按照一定方式协同、继承才能完成;或者至少是大数据集,甚至是海量信息发布。为此,基于 SVG 的应用系统开发,必须解决以下几个问题:

(1) 单纯利用单个的 SVG 文档的特性或优势是不足以应对这些应用系统的网上发布及交互时所出现的运行速度问题;

(2) 如何解决 SVG 的数据重用问题;

(3) 就大型应用系统而言,文档形式肯定不是管理数据的理想解决方案,必须有数据库的支撑,如何将 SVG 存入数据库,也是必须解决的问题。

为了解决上述问题,同时使基于 SVG 的应用开发更为简洁和便捷,充分利用了 SVG 图形图像类的构成模型和 SVG 文档的优点,将其与关系数据库结合起来,结合一般软件体系结构的特点,设计了一种实现基于 SVG 图形应用的软件体系结构 SSA 和系统框架结构如图 4、图 5 所示。

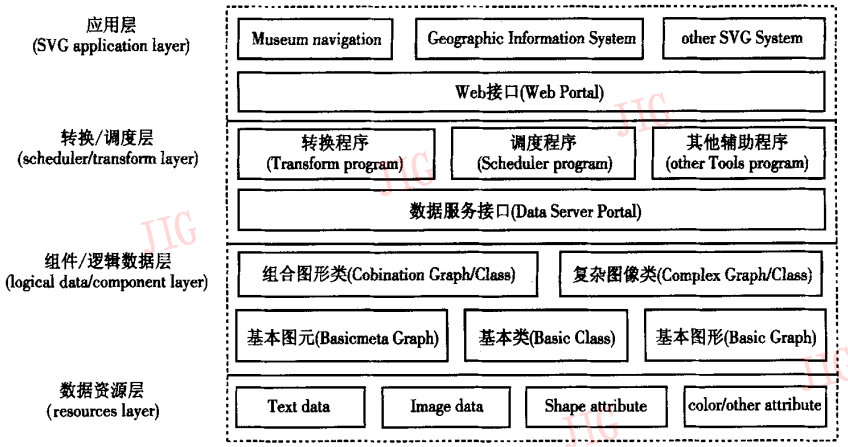

图 4　基于 SVG 应用的软件体系结构(SSA 的层次结构)

Fig. 4　SVG-based software layer architecture

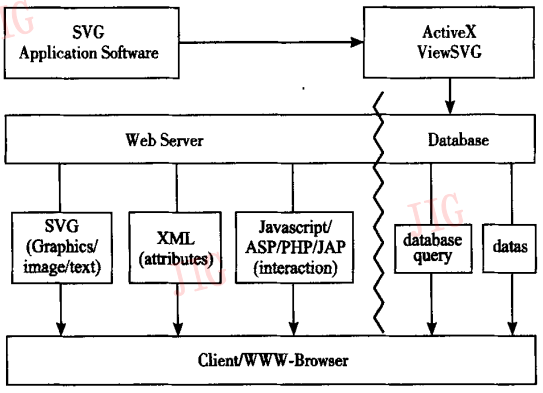

图 5　SVG 应用的系统框架

Fig. 5　A system framework of SVG application

在图 4 中,SSA 具体包含 4 个层次:应用层(SVG application layer)、转换/调度层(scheduler/transform layer)、组件/逻辑数据层(logical data/component layer)、数据资源层(resources layer),其中,应用层和转换/调度层是整个体系结构的核心。

如图 5 所示,SVG 应用的系统框架,是基于 Web 的 B/S 结构的扩展,运行在 TCP/IP 协议之上,应用在 Web 服务之中,增加了专门处理 SVG 文档的 ActiveX-View-SVG 控件,通过数据库 Database 和 Web Server 接口,支持了 SVG、XML、ASP/JSP/Java script 技术以及多种数据库数据等多种数据格式的处理和查询。

(1) 应用层(SVG application layer)　应用层主要在整个软件系统执行的生命周期中使用 SVG 进行内容表现和与用户进行交互,为用户提供各种基于 SVG 应用的各种服务程序,如:数字博物馆的平面导航系统、地理信息系统、电子线路板平面图形系统等,这些系统利用 Web 接口提供的基于 B/S 结构包括 SVG 数据库系统的各种数据,由于大型的应用系统是由若干个 SVG 文档的组合和按照一定方式协同、继承、进行内容表现和与用户进行交互,其速度、效率及结构的清晰性,是本层面临的主要问题。

定义 1　SVG 时间图形集($\{\Delta\}$,Δt),在基于

SVG 软件系统执行的生命周期中，把一个 2 维平面内的所有内容表现和与用户进行交互的信息看成一系列的 SVG 实体图形 Δ。如果一段时间 Δt，所有表现的 SVG 实体组成的集合（记为 $\{\Delta\}$），称为 SVG 时间图形集，记为（$\{\Delta\},\Delta t$）。

定义 2 视图展示时间窗体（viewbox time window） 在应用层中设定了一个视图展示时间窗体，该窗体指的是在单位时间 t，与用户交互的 SVG 实体集合。因此，时间段 Δt 在视图展示时间窗体中包含的 SVG 实体就是 SVG 时间图形集，如图 6 所示。视图展示时间窗体随着时间 t 滑动，代表着软件系统的执行过程。

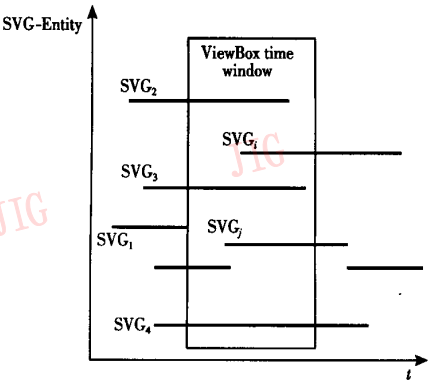

图 6 SVG 的视图窗口

Fig. 6 A viewbox time window of SVG application

SVG 应用层的实体有限性原则 一个（$\{\Delta\},\Delta t$）由一系列有限的 $\{SVG_i, i=1,2,\cdots,n\}$ 实体构成，也就是说，应用层的 SVG 实体可数，这些 SVG 实体构成一个 SVG 文档片断，其中，SVG_i 是 SVG 基本实体对象或复杂的实体对象。

SVG 应用满足的贴图原理 （$\{\Delta\},\Delta t$）根据相似距离[12]的定义，可以划分若干个相似图形类，记为 $\{SVG\_Class_i, i=1,2,\cdots,m\}$，$(\{\Delta\},\Delta t)=\sum_{i=1}^{m} SVG\_Class_i$，其中每一个 SVG_Class_i 可以由一个基本图元 $\{\langle B\rangle\}$ 生成。

应用层的实体有限性原则与贴图原理的作用是主要解决应用层的运行效率和数据重用问题，根据应用层的上述规则和定义，在基于 SVG 2 维平面中，可以通过转换/调度层将应用层中的 SVG 实体分类，提取 SVG 的公共实体，即基本图元 $\{\langle B\rangle\}$，由其通过仿射参数生成各种 SVG 的复杂实体，以高速和高效的方式来进行内容表现和与用户交互，同时也解决了 SVG 的组件重用问题。

（2）转换/调度层（scheduler/transform layer）主要作用：一是对 SVG 实体分类，提取 SVG 公共实体因子；二是生成复杂 SVG 实体；三是控制视图展示时间窗体，与用户交互。

转换层的核心是转换算法和调度算法。转换算法的作用是对 SVG 实体分类，提取 SVG 公共实体因子，它将整体软件周期中的 $\{SVG_i, i=1,2,\cdots,n\}$ 实体根据 SVG 实体形状 $e(B_i)$ 和色彩属性 c_i 使用分类算法[13]分别划分为 M 类和 K 类，对每一类提出公共实体因子生形状生成元 $\langle e(B_j)\rangle, j=1,2,\cdots,m$ 和颜色生成元 $\langle c_l\rangle, l=1,2,\cdots,k$，计算 SVG_i 实体的生成元对 $(\langle e(B_j)\rangle,\langle c_l\rangle)$，生成 SVG_i 的仿射参数 *transform*，这样只需存储公共实体因子生成元 $\langle e(B_j)\rangle$、$\langle c_l\rangle$ 和 SVG_i 的仿射参数 *transform*，就能压缩 SVG_i 实体的存储空间，提高 SVG_i 的运行效率，该转换算法过程为

输入：$\{SVG_i, i=1,2,\cdots,n\}$

输出：(1) $\{SVG_i(e(\langle B_j\rangle), c_l, transform), i=1,2,\cdots,n; j=1,2,\cdots,m; l=1,2,\cdots,k\}$

(2) $\{e(\langle B_j\rangle), c_l), j=1,2,\cdots,m; l=1,2,\cdots,k\}$

Step1 根据每一个 $\{SVG_i, i=1,2,\cdots,n\}$ 图形性质，将其划分为 $(e(B_i), c_i)$，其中 $(e(B_i))$ 代表 SVG_i 形状，c_i 代表色彩等属性。

Step2 根据分类算法[13]将所有 SVG_i 实体根据形状分类，划分为 m 类；同时根据分类算法将 SVG_i 实体根据色彩分类，划分为 k 类。

Step3 提取 SVG 实体形状集合 $\{e(B_j)\}$ 中的每一类的生成元 $\langle e(B_j)\rangle, j=1,2,\cdots,m$。

Step4 提取 SVG 实体色彩集合 $\{c_l\}$ 中的每一类的生成元 $\langle c_l\rangle, l=1,2,\cdots,k$。

Step5 根据 SVG_i 的 $(e(B_i), c_i)$ 与 $(\langle e(B_j)\rangle,\langle c_l\rangle)$ 的差异性计算 SVG 实体的转换参数，转换参数 $transform = \{(e(B_i)-\langle e(B_j)\rangle), (c_i-\langle c_l\rangle)\}$

Setp6 根据 Step3，Step4 输出 $\{e(\langle B_j\rangle), c_l)\}$，根据 Step5 输出 $SVG_i(\langle e(B_j)\rangle,\langle c_l\rangle, transform)$

调度算法，它包含就绪队列和消亡队列两个队列，就绪队列包含即将展示的 SVG 实体，消亡队列包含即将终结的 SVG 实体。调度算法主要为应用层实时提供需要交互的 SVG 实体，也就是说接收逻辑数据层传输来的相关数据源，根据不同的数据源的各种不同数据格式特点，生成 SVG 实体对象，使其进入就绪状态，进入就绪队列，在其生命周期 Δt 内控制视图展示时间窗体，使相关 SVG 实体进入激

活状态,当某个 SVG 执行终结后,撤销该实体,使其进入消亡队列。

(3) 组件/逻辑数据层(logical data/component layer)　在 SSA 中,主要存储和检索各种各样的 SVG 的基本组件、SVG 实体图形等,为转换层提供数据和接收转换层的数据保存为物理数据,其存储的数据格式可以是各种数据库中的 SVG 组件。

(4) 数据资源层(resources layer)主要组成 SVG 实体的各种数据资源和 SVG 的标识命令、属性、Image 实体图像和 Text 实体等,它们是 SSA 中最基本的原子元素。

SSA 能满足各种基于 SVG 的应用,通过层次的划分,使得基于 SSA 的 SVG 应用开发变得更为简单,其中,SVG 组件[14]的重用率、运行效率和速度,比起普通的 SVG 应用开发,有明显提高。

4　应用实例

为了验证 SSA 软件体系结构的有效性和使用关系型数据库 RDBMS 存取 SVG 实体/类的合理性,设计了一个基于 SSA 的虚拟数字博物馆 2 维平面导航和文物管理系统,该系统主要用于快速展示文物数字化图片、视频以及大容量博物馆平面导航展示图,系统结构如图 7 所示。

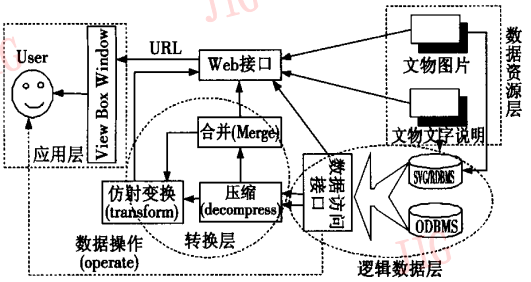

图 7　基于 SSA 虚拟数字博物馆的系统结构图

Fig. 7　A system framework of virtual digital museum based on SSA

(1) 在应用层,使用 SVG 格式文件描述了徐悲鸿纪念馆的 3 个文物展室的平面结构,显然由于需要展示给用户的信息很多,一个 SVG 图形是无法描述的,将其按图层划分若干个 SVG 个子图形实体,构成一个逻辑树形结构,通过视图展示时间窗体进行重载展示给用户。在 SVG 子图形实体中,随着用户的交互,通过 Web 服务接口调用了转换层提供的各种有关文物数字信息。

(2) 转换层主要将数据库的 SVG 描述的文物公共因子和文物分段图片通过解压(decompress),平移、旋转、放大等仿射变换(transform),合并等措施为应用层提供 SVG 实体。

(3) 逻辑数据层主要利用关系数据库和 XML-Natived 数据库存储的 SVG 文物公共图元和 SVG 文物分段图片查询和检索数据文件为转换层提供合成满足用户交互的文物展示信息。

(4) 有些无法直接使用 SVG 描述的文物信息和文字信息,一方面应用层可通过 Web 结构直接使用,另一方面通过在数据库中保存这些信息的地址来使用。

在基于 SSA 的虚拟数字博物馆 2 维平面导航和文物管理系统中,合成了12 173个 SVG 图片、存储了7 200 个 Imge 图片和13 205 段文字信息,其中12 173个 SVG 图片使用了3 518个 SVG 片断合成,并将3 518个 SVG 片断存储在关系数据库中。

另外,为了验证 SSA 有效性和合理性,同时使用传统的基于 Html/Image 方式,建立 2 维平面导航和文物管理系统原型系统,其数据使用了 JPEG 图形格式表现的所有 SVG 图片的信息、Image 图片和文字信息不做改变,并从以下几个方面比较了这两个系统:

(1) 在博物馆导航系统 2 维导航平面地图,由于要表达的信息量很大,如果使用一个 SVG 文件或 Image 文件由于文件大,其图片载入速度比较慢。为解决这个问题,采取通过划分多个子 SVG 文件或 Image 文件,进行有效组合,构成逻辑树形结构。两种开发方式的载入效率比较如表 1 所示,基于 SSA 的开发效果要比传统方式快了 20 多倍。

表 1　导航系统用户交互界面载入效率比较

Tab. 1　The load efficiency of user interface of explorer system

开发方式	View Box Windows	图片载入平均时间(s/每幅)	图片显示效果
基于 Html/Image	多个 Image 组合,每幅图片 50K ~ 630K 字节	5	放大变形
基于 SSA	多个 SVG 构成树形结构,每幅图片 10K ~ 30K 字节	0.18	放大不变形

(2) 在载入文物图片信息时,定义文物图片的重载率

$$CL = \frac{Number\_of\_Graph\_Class}{Number\_of\_Graph}$$

表示在软件执行周期中展示文物图片实体的总数除以实际保存图元的总数(如果使用 SVG 描述图片,由于 SVG 本身是可解析执行的代码,因而 *CL* 就是 SVG 组件的重载率),比较两种开发方式的重载率 *CL*,如图 8 所示。

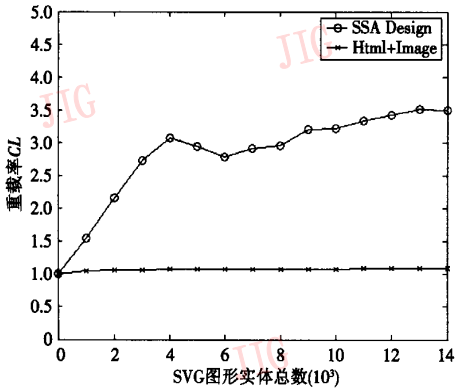

图 8 重载率 *CL*

Fig. 8 A reload ratio of SVG component

(3) 由于每个文物实体比较大,需要划分更小的实体,进一步生成图元,保存图元所需存储空间 $\sum_{i=1}^{m} SVG\_Class_i$ 比保存实体存储空间 $\sum_{j=1}^{n} SVG\_Entity_j$ 会更小、执行效率也会更高,令压缩率

$$C = 1 - \frac{\left(\sum_{i=1}^{m} SVG\_Class_i\right)}{\left(\sum_{j=1}^{n} SVG\_Entity_j\right)}$$

比较两种开发方式的压缩率 *C*,如图 9 所示。

通过以上方式的比较,充分证明了基于 SSA 开发的良好性能。

5 结 论

SVG 的出现为图形化网页设计带来了新的革命,同时也使 Web 技术的交互界面的开发进入了矢量图形图像的时代。但是基于 XML 的 SVG 应用的系统设计时,由于 XML 是半结构化的置标语言,有其天生模块化较差的弱点,并且大规模数据处理很难由单个 SVG 文档来完成,因此在实际应用时,可将客观世界的实体转换成 SVG 实体,将每个 SVG 实体模块化和结构化[10,11],研究这些结构化的 SVG

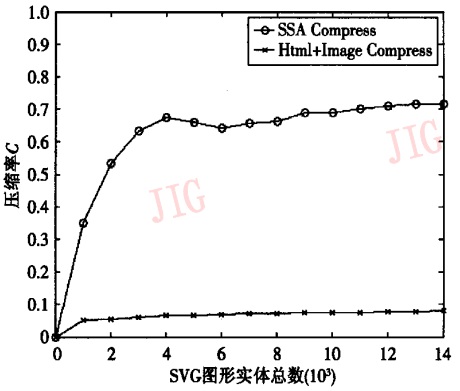

图 9 压缩率 *C*

Fig. 9 A compress ratio of SVG

实体,对其进行分类[13],提取 SVG 实体的公共形状图元和公共颜色特征生成元,存储在数据库中,通过 SSA 进行开发和应用,就能克服 SVG 的弱点,为 SVG 的结构化应用和软件开发便捷性,提供良好的理论基础。

但是本文的转换算法仍有一定的局限性,比如转换算法的最优性和复杂度,另外,基于 SVG 数据库的快速检索算法仍在研究中。考虑到 SVG 具有广泛的应用前景,且其基于 XML,能够继承 XML 的所有特点和优点,未来我们将重点改进转换算法、研究基于 SVG 图形的数据压缩算法以及研究开发图形搜索引擎的可能性,同时基于 SVG 图形网页人机交互也是研究的重点。

参考文献(References)

1 Scalable Vector Graphics (SVG) 1. 2 Specification [EB/OL]. http://www.w3.org/TR/2004/WD-SVG12,2004-10-27.

2 Introduction to Scalable Vector Graphics [EB/OL]. http://www-900. ibm. com/developerWorks/cn/education/xml/x-svg/tutorial-eng/index. Html, 2002-08.

3 Sagar M S. An SVG browser for XML languages [A]. In: Proceedings of Theory and Practice of Computer Graphics 2003 Conference[C], Birmingham, USA, 2003: 42 ~ 48.

4 Quint. A Scalable vector graphics[J]. Multimedia, IEEE, 2003, **10**(3): 99 ~ 102.

5 Zhou Wen-sheng. Research of WebGIS based on SVG[J]. Journal of Image and Graphics, 2002, 7(7): 694 ~ 698. [周文生. 基于 SVG 的 WebGIS 研究[J]. 中国图象图形学报, 2002, 7(7): 694 ~ 698.]

6 Zhang Jun, Guan Ji-hong, Zhang Jian-hua, *et al*. Geographic information integration and publishing based on GML and SVG[A]. In: Proceedings of The 4th In' 1 Conference on Computer and

Information Technology[C], Wuhan, China, 2004: 764 ~ 769.

7 Marriott K, Meyer B, Tardif L. Fast and e _ cient client-side adaptivity for SVG[A]. In: Proceedings of 11th International World Wide Web Conference[C], Honolulu, Hawaii, USA, 2002: 496 ~ 507.

8 Gan Zao-bin, Li Zhi-xin, Peng Bin. Data description model of vector graphics editing system based on SVG[J]. Computer Engineering and Design, 2005, **26**(1): 270 ~ 273. [甘早斌, 李志欣, 彭彬. 基于 SVG 的矢量图形编辑系统的数据描述模型[J]. 计算机工程与设计, 2005, **26**(1): 270 ~ 273.]

9 Hu Chun-ming, Huai Jin-peng, Sun Hai-long. Web service-based grid architecture and its supporting environmen [J]. Journal of Software, 2004, **15**(7): 1064 ~ 1073. [胡春明, 怀进鹏, 孙海龙. 基于 Web 服务的网格体系结构及其支撑环境研究[J]. 软件学报, 2004, **15**(7): 1064 ~ 1073.]

10 Yuan Jia-zheng, Xu De, Bao Hong. Architecture design & implementation of XML-based SVG [J]. Journal of Computer Research and Development, 2005, **42**(Suppl A): 129 ~ 136. [袁家政, 须德, 鲍泓. 基于 XML 的矢量图形 SVG 的结构设计与实现[J]. 计算机研究与发展, 2005, **42**(Suppl A): 129 ~ 136.]

11 Yuan Jia-zheng, Xu De, Shen Hong, *et al*. Effective structure method for SVG[J]. Journal of Simulation, 2006, **18**(S1): 5 ~ 7, 9. [袁家政, 须德, 沈洪等. 一种有效的矢量图形 SVG 的结构化方法(英文)[J]. 系统仿真学报, 2006, **18**(S1): 5 ~ 7, 9.]

12 Yuan Jia-zheng, Xu De, Bao Hong. Study of SVG database model & storage based on XML [J]. Journal of Computer Research and Development, 2006, **43**(Suppl): 444 ~ 450. [袁家政, 须德, 鲍泓. 基于 XML 的矢量图形 SVG 的数据库模型与存储研究[J]. 计算机研究与发展, 2006, **43**(Suppl): 444 ~ 450.]

13 Yuan Jia-zheng, Xu De, Bao Hong. An efficient XML documents classification method based on structure and keywords frequency[J]. Journal of Computer Research and Development, 2006, **43**(8): 1361 ~ 1367. [袁家政, 须德, 鲍泓. 基于结构与文本关键词相关度的 XML 网页分类研究[J]. 计算机研究与发展, 2006, **43**(8): 1361 ~ 1367.]

14 Wang Zhong-jie, Xu Xiao-fei, Zhan De-chen. Reuse、cost optimization oriented component refactoring method[J]. Journal of Software, 2005, **16**(12): 2157 ~ 2165. [王忠杰, 徐晓飞, 战德臣. 面向复用成本优化的构件重构方法[J]. 软件学报, 2005, **16**(12): 2157 ~ 2165.]