文章编号:1671-251X(2012)12-0033-04
王丽丽. SVG 的 3D 实现[J]. 工矿自动化,2012(12):33-36.

SVG 的 3D 实现

王丽丽

(天地(常州)自动化股份有限公司,江苏 常州 213015)

摘要:针对 SVG 在 3D 图形的描述与绘制上存在局限性的问题,提出了一种 SVG 的 3D 实现方法;详细介绍了 SVG 的 2D 坐标系到 3D 坐标系的转换原理,给出了转换到 3D 坐标系后的 SVG-3D 图形的绘制与操控流程,并以在 3D 场景中绘制一个正立方体和圆球体为例,介绍了在 HTML 页面中采用 JavaScript 创建 3D 场景和绘制与操控这 2 个图形的实现。

关键词:可缩放矢量图形;2D 坐标系;3D 坐标系;坐标转换;3D 场景;JavaScript
中图分类号:TD67 文献标志码:A 网络出版时间:2012-11-29 15:12
网络出版地址:http://www.cnki.net/kcms/detail/32.1627.TP.20121129.1512.010.html

3D Implementation of SVG

WANG Li-li
(Tiandi (Changzhou) Automation Co., Ltd., Changzhou 213015, China)

Abstract: In view of problem of limitation on description and drawing of 3D graph by SVG, the paper proposed a 3D implementing method of SVG. It introduced conversion principle of 2D coordinate to 3D coordinate of SVG in details, gave processes of drawing and operation control of SVG-3D graph on 3D coordinate. Taking drawing a cube and a spherosome in 3D scene as example, it expounded implementation of creation of 3D scene and drawing and operation control of the two graphs in HTML page by use of JavaScript.

Key words: SVG, 2D coordinate, 3D coordinate, conversion of coordinate, 3D scene, JavaScript

0 引言

SVG(Scalable Vector Graphics,可缩放矢量图形)是基于可扩展标记语言(XML),用于描述二维矢量图形的一种图形格式和标准规范,由 W3C(World Wide Web Consortium,国际互联网标准组织)制定[1]。随着 Internet 的迅速发展,人们对于 Web 环境下的地理信息获取需求也日益明显,SVG 矢量地图的 XML 格式十分有利于地理空间数据的描述和呈现,使得其在 Web 环境及手机应用的图形绘制等领域占有相当大的份额[2]

近年来,随着各种各样 3D(Three-dimensional,三维)技术的产生以及 3D 技术对人们视觉效果的冲击,2D 矢量图形已满足不了人们对立体直观的视觉效果的追求。然而,SVG 是用于描述二维矢量图形的,在 3D 图形的描述与绘制上存在局限与限制。为了在 Web 中实现 3D 的 SVG 绘制,就需要 SVG 结合并利用 JavaScript 实现 2D 坐标系到 3D 坐标系的转换,并实现矢量图形绘制从 2D 到 3D 的跨越。

1 SVG 的 3D 坐标系

可以把 SVG 的 3D 坐标系看作是在 2D 坐标系基础上的扩展,以下将详细介绍 SVG 的 2D 坐标系到 3D 坐标系的转换。

如图 1 所示,当站在坐标(800,0,0)位置,朝坐

收稿日期:2012-07-20
基金项目:天地科技技术创新基金项目(KJ-2012-TDCZ-03)
作者简介:王丽丽(1983-),女,湖北鄂州人,工程师,现主要从事煤矿相关系统的软件开发工作。E-mail:heblinlili@163.com

标原点(0,0,0)看过去,将看到所有 3D 对象都在原点附近,当把对象绕 Z 轴旋转一个 ϕ 角度时,那么,点(x', y',z')的新坐标位置(x'', y'',z'')是多少呢?显而易见,其相对于 Z 轴分量并没有改变,因此,可以得出,$z''=z'$,但是 x'和 y'又是多少呢?

从图 1 的顶部往下看时,看到的是如图 2 所示的二维坐标系(x,y)。

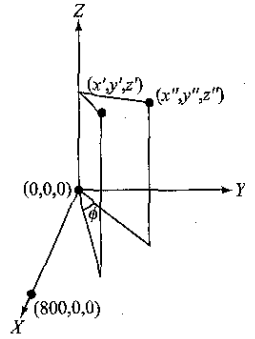


图 1 点(800,0,0)、(x',y', z')和(x'', y'', z'')在 3D 坐标系中的位置

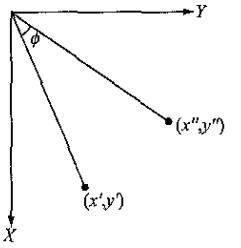
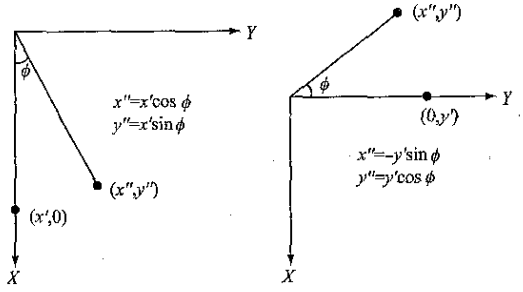


图 2 点(x',y', z')和(x'', y'', z'')在二维坐标系(x,y)上的投影

能否找出一个公式,根据 x'、y'和 ϕ 计算出 x''和 y''呢?似乎有点难,但是,当单独计算 x'和 y'时,就变得容易起来,如图 3 所示。



(a) 在 X 轴上的映射 (b) 在 Y 轴上的映射

图 3 点 (x', y')分别在 X 轴和 Y 轴上的映射

从图 3 可得出

$$\begin{cases} x'' = x'\cos\phi \\ y'' = x'\sin\phi \end{cases} \tag{1}$$

$$\begin{cases} x'' = -y'\sin\phi \\ y'' = y'\cos\phi \end{cases} \tag{2}$$

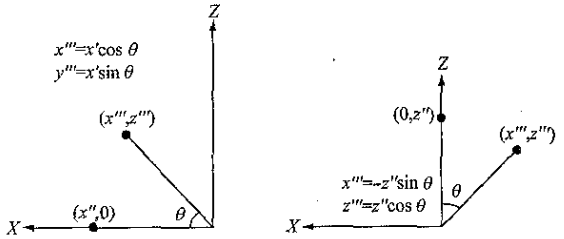
由(x',0)和(0,y')到(x',y'),可以把 x、y 相加并补充扩展得出式(3):

$$\begin{cases} x'' = x'\cos\phi - y'\sin\phi + z'\times 0 \\ y'' = x'\sin\phi + y'\cos\phi + z'\times 0 \\ z'' = x'\times 0 + y'\times 0 + z'\times 1 \end{cases} \tag{3}$$

对象绕 Z 轴旋转一个 ϕ 角度,通过观察和分析,可以通过矩阵和向量来描述上述 3 项:

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{4}$$

那么再次将所有对象绕 Y 轴旋转一个 θ 角度后的(x''',y''',z''')又将如何计算得出呢?从一个很高的位置往下看这些 3D 对象,Z 轴将保持垂直。而 Z 轴始终保持垂直并指向屏幕上方,是用于描述 3D 对象的基础过程。所以,参照对象绕 Z 轴旋转时的算法,从右侧朝 3D 对象看过去时,同理,简化得出的坐标系如图 4 所示。



(a) 在 X 轴上的映射 (b) 在 Z 轴上的映射

图 4 点 (x'', y'')分别在 X 轴和 Z 轴上的映射

对象绕 Y 轴旋转一个 θ 角度,通过分析也可得出

$$\begin{cases} x''' = x''\cos\theta - y''\times 0 - z''\sin\theta \\ y''' = x''\times 0 + y''\times 1 + z''\times 0 \\ z''' = x''\sin\theta + y''\times 0 + z''\cos\theta \end{cases}$$

用矩阵表示则如式(5)所示:

$$\begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} \tag{5}$$

将 2 个旋转一步到位,而不计算(x'',y'',z'')向量时,可得出如下关系:

$$\begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \times \left[\begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \right] \tag{6}$$

根据矩阵乘法简化式(6),得到

$$\begin{pmatrix} x''' \\ y''' \\ z''' \end{pmatrix} = \begin{pmatrix} \cos\phi\cos\theta & -\sin\phi\cos\theta & -\sin\theta \\ \sin\phi & \cos\phi & 0 \\ \cos\phi\sin\theta & -\sin\phi\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \tag{7}$$

最后，得出该点经过 2 次旋转后在屏幕坐标系的坐标(x^s, y^s, z^s)为

$$\begin{cases} x^s = y''' = x'\sin\phi + y'\cos\phi \\ y^s = z''' = x'\cos\phi\sin\theta - y'\sin\phi\sin\theta + z'\cos\theta \\ z^s = x''' = x'\cos\phi\cos\theta - y'\sin\phi\cos\theta + z'\sin\theta \end{cases} \tag{8}$$

同理，在 X 轴上的旋转也可以得出类似上述的公式，因此，由 2D 坐标系到 3D 坐标系的转换与绘制有规律及方法可循。

当 3D 对象在如下限定坐标范围框内$\{x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}\}$，其中心点相对 3D 坐标系原点的坐标为

$$\begin{cases} x_d = (x_{min} + x_{max})/2 \\ y_d = (y_{min} + y_{max})/2 \\ z_d = (z_{min} + z_{max})/2 \end{cases}$$

当将 3D 对象的中心点视作坐标原点，则可视作坐标原点分别在 X、Y、Z 轴上作平移运动，则有 $x' = x - x_d$，$y' = y - y_d$，$z' = z - z_d$，因此，3D 对象将出现在以 3D 对象为中心即$\{x_{min} - x_d, x_{max} - x_d, y_{min} - y_d, y_{max} - y_d, z_{min} - z_d, z_{max} - z_d\}$坐标范围框内[3]。这种方式绘制对象使得 2D 坐标系到 3D 坐标系的绘制和操纵变得相对简单与实际可行。

2　SVG－3D 的绘制和操控

整个 3D 场景由许多相对独立的 3D 对象组成，以 3D 对象为元素单元，逐一绘制并构建成一个整体的 3D 场景。

在 3D 场景范围内的所有 3D 对象就像是绑定在一起的一个更大的 3D 对象。因此，3D 对象的操作如缩放、平移、旋转等将围绕 3D 场景中心点进行，这使得其有更好的空间想象力和视觉感受效果。

3D 对象的操控主要有放大、缩小、变焦、光束移动、视角改变、水平垂直位移、动态旋转等。

(1) 矢量缩放

SVG 是 XML 格式的数据，实现 3D 对象的矢量放大与缩小，图形显示清晰且不失真。

(2) 3D 对象的变焦

通过焦距的调整，可将 3D 对象全方位呈现在屏幕上。

(3) 光束与阴影

通过调节光束发散的位置，实现对象在 3D 场景中显示的立体效果与阴影，还原真实的图形世界，显现更为直观的视觉效果。

(4) 俯视视角变化

通过俯视角度的调整变化，可以从不同角度观看 3D 对象，改变了 2D 图形只能观测到对象的一个平面的局限，犹如真实场景中的人与实物。

(5) 3D 对象的水平、垂直位移

可对 3D 对象进行水平方向、垂直方向的移动。

(6) 动态旋转

可针对整个 3D 场景进行动态旋转，使得 3D 对象的所有立面都可以在旋转过程中呈现在眼前，对 3D 对象的观测从上到下、从左到右均可 360°一览无余。

3　SVG 的 3D 实现

一个复杂形状的 3D 图形，都是由 3 种基本的简单图形元素(点、线、面)组成，因此，在实现复杂的 3D 图形绘制时就需要将 3D 图形先分类分割，然后先绘制出简单图形，再组合起来以形成复杂形状的 3D 图形。一个完整的 3D 场景就相当于一个很大的容器，其中包含许多复杂形状的 3D 图形。因此，SVG 的 3D 图形绘制应先建立 3D 场景这个容器，然后再实施各种 3D 对象的绘制。

3D 对象在 HTML 页面中的绘制与操作是利用 JavaScript 与 HTML 进行交互与操控的，将所有这些交互与操控方法都封装成一个 JavaScript 库。在 HTML 页面中定义 3D 场景时，只需引用该库，并通过该库中相关的类和函数就可以简单地实现 3D 对象的绘制和操控[4-5]。该库封装有 3D 场景的创建、基本图形元素绘制方法类和 3D 对象的缩放、变焦、光束移动、视角改变、水平垂直位移、动态旋转等操作方法类。

通过 3 种基本的简单图形元素(点、线、面)绘制方法类，可以绘制出任何 3D 效果的立体几何图形，这将使 SVG 突破在 3D 图形绘制上的局限与限制。

以在 3D 场景中绘制一个正立方体和圆球体为例，在 HTML 页面中实现的 3D 场景的创建和正立方体和圆球体的绘制与操控如图 5 所示。

在 3D 场景中绘制圆球体的方法和算法如图 6 和图 7 所示，图 6 中，$r = R\sin\alpha_{Th1}$，$a_Z = R\cos\alpha_{Th1}$；图 7 中，$a_X = r\cos\alpha_{Th2}$，$a_Y = r\sin\alpha_{Th2}$；将一个圆球体经过水平切割成 N 个平面圆，在 Z 轴方向上先从上

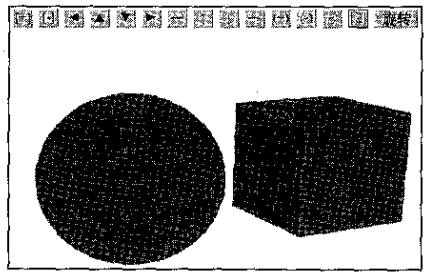


图 5 正立方体和圆球体的 3D 绘制与操控示例

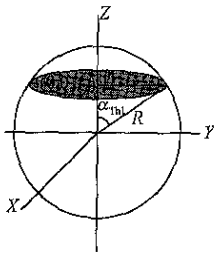


图 6 圆球体水平平面切割

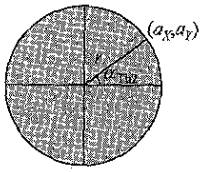


图 7 切割的平面圆

到下绘制平面圆，圆球体则由被切割成的这 N 个平面圆组成，角度 α_{TH1} 参数递增越小，切割得越多，N 值就越大，3D 球体绘制得就越真实。

在 3D 场景中绘制圆球体的实现代码：

```
//圆球体绘制方法类
function Circle3D ( aParentScene, aR , aFrontColor, aBackColor,
aStrokeColor, aStrokeWeight)
{ this. Parent=aParentScene;
  this. ClassName="Circle3D";
  this. Center=new Vector(0,0,0);
  ...
  this. Poly3D=new Array();//3D 对象数组
  var p1=0,P1=3. 141 592 65,αTh1=2,αTh2=10; // 角度 360 的完
  整圆弧按照 αTh2角度分割
  for ( var i=-90;i≤90;i+=αTh1 ) //球体按照 αTh1角度水平切割
  平面，αTh1越小，3D 球体绘制越真实
{
   var aX=0,aY=0,aZ=0,R=0;
   var cos=0;
   var sin=Math. sin(i * PI/180);
   if(i≥0)
{ cos=Math. cos(i * PI/180);}
  else
{ cos=-Math. cos(i * PI/180);}
  R=aR * Math. abs(sin);
  aZ=aR * cos;
  //平面圆绘制方法
  this. Poly3D [p1 ] = new Poly3D ( aParentScene, aFrontColor,
  aBackColor, aStrokeColor, aStrokeWeight);
  with (this. Poly3D[p1]) {
  for(var aFi=0;aFi≤360;aFi+=αTh2) {
  var ccos_Fi=Math. cos(aFi * PI/180);
  var ssin_Fi=Math. sin(aFi * PI/180);
  aX=R * ccos_Fi;
  aY=R * ssin_Fi;
  AddPoint(aX,aY,aZ);
}
   Update();
}
   p1++;
}
}
//正立方体绘制方法类
function Face4 ( aParentScene, aFrontColor, aBackColor,
aStrokeColor, aStrokeWeight);
```

正立方体仅由 6 个正方形平面组成，其 3D 绘制方法相对于球体的 3D 绘制方法较简单，这里就不再赘述。

4 结语

介绍了一种通过 2D 坐标系到 3D 坐标系的变换方法，并结合 JavaScript 与 HTML 易交互特性，使得在 Web 环境中 SVG 的 3D 矢量图形的绘制与操控变得简单且易实现。

当 SVG 从 2D 坐标系到 3D 坐标系的转换与跨越突破各种局限而成为可行与可实现时，基于 XML 格式的 SVG 在信息共享上的优势以及 SVG 的 3D 矢量图形给人们带来的直观的立体视觉感受效果和空间想象力，将使得其具有更为广泛的应用价值，因此，SVG 从 2D 坐标系到 3D 坐标系的转换与跨越将成为 SVG 发展道路上最大的推动力。

参考文献：

[1] W3C Working Draft. Scalable Vector Graphics (SVG) 1. 1[EB/OL]. (2011-08-16)[2012-07-04]. http://www. w3. org/TR/SVG11/.

[2] 刘啸，毕永年. 基于 XML 的 SVG 应用指南[M]. 北京：中国水利水电出版社，2001.

[3] WATT A. 3D 计算机图形学[M]. 包宏，译. 3 版. 北京：机械工业出版社，2005.

[4] 陈勇，李少华，史瑞芝. 基于 SVG 的地图网络发布技术研究[J]. 测绘科学技术学报，2007，24(2)：149-152.

[5] 纪陵，蒋衍君，施广德. 基于 SVG 的电力系统图形互操作研究[J]. 电力自动化设备，2011，31(7)：105-109，114.

[6] 曲锋. 基于 SVG 和 Ajax 的电网一次接线图实现[J]. 电网与清洁能源，2010，26(3)：41-44.

[7] 罗志伟，华庆一，李娟. 基于 SVG 的自适应用户界面工具开发[J]. 计算机工程，2010，36(5)：70-72.

[8] 王丽丽，白晓波. 基于 SVG 的 Web 煤矿地图应用[J]. 工矿自动化，2011，37(3)：9-12.