Introduction to dojox.gfx3d library

Kun Xi

June 27, 2007

Abstract

dojox.gfx3d is proposed to render 3D objects in web browser, this introduction states the reasoning, desigh philosophy and unleashes some implementation details for the users to have a better understanding how this library works.

1 History of Graphic Library

dojo.gfx library has been developed since the Dojo Summer of Code 2006, This 2D graphics library provides a unified interface to manipulate the vector graphics, with 95% behavior consistence[1] for SVG and VML render engines, across Mozilla Firefox, Safari, Opera and Microsoft Internet Explorer. Chart developers show great interest in dojo.gfx for data visualization and they encourage us to extend the library to support three dimension objects.

dojox.gfx3d is the new library for this feature request. Careful readers may notice the subtle name change: dojo.gfx is developed in dojotoolkit 0.4 code base, using *dojo* namespace; and dojox.gfx3d is proposed to work in dojo 0.9 code base only, using *dojox* namespace. To make things even worse, we have migrated dojo.gfx to the 0.9 code base and rename it as dojox.gfx. In short, dojo.gfx and dojox.gfx refer to 2D graphics library, with the same functionality but in different code base, dojox.gfx3d is the 3D graphics library.

Notice: all the names are subject to change since we are in quite a early stage.

2 Fundamental of Computer Graphics

Mathematically, three dimensional object is represented by sets of points (x, y, z). The coordination can be translated, rotated and scaled, i.e *world transform*[2] in 3D domain; then 3D objects are projected to 2D shapes for presentation, aka *Camera transform*.

2.1 World Transform

World transform is quite useful. Consider the the problem of a rotated cube. There are two approaches:

[1] Due to the limitation of VML, the behavior of radiantGradience is not the same.
[2] 3D Projection: http://en.wikipedia.org/wiki/3d_projection

1. Calculate the coordinates manually; which is a troublesome, tedious.
2. Create an "normal" cube, then apply a set of world transformations that move it to the right position and orientation.

Let's examine the second approach:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{- translation matrix}$$

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha & 0 \\ 0 & sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{- rotation about the } x\text{-axis}$$

$$\mathbf{R_y} = \begin{bmatrix} cos\beta & 0 & sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\beta & 0 & cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{- rotation about the } y\text{-axis}$$

$$\mathbf{R_z} = \begin{bmatrix} cos\gamma & -sin\gamma & 0 & 0 \\ sin\gamma & cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{- rotation about the } z\text{-axis}$$

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{- scale matrix}$$

dojox.gfx3d uses OpenGL convention for the world transform, the world transformation matrix is defined as:

$$\mathbf{W} = \mathbf{T} \times \mathbf{R_x} \times \mathbf{R_y} \times \mathbf{R_z} \times \mathbf{S} \tag{1}$$

We apply the world transform matrix to the left side of the corordination (x, y, z) to get the new coordination(x', y', z'):

$$\begin{bmatrix} x' \\ y' \\ z' \\ c \end{bmatrix} = \mathbf{W} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2}$$

2.2 Camera Transform

The viewer's coordinates determine the mapping of a 3D object to a 2D space. These coordinations describe the position and angle of the viewer.

Isometric projection is used in dojox.gfx3d to project the transformed 3D objects to 2D shapes, which is much simpler than *persepctive projection* and still widely used in the pseudo 3D games. The identity camera transform casts (x, y, z) to (x, y), imagine parallel lights from the Z-axis casting a shadow on $X - Y$ plate. As we transform, rotate the plate, the coordinations will change. In other words, we change the scene's position in space, while the position of the illumnation source does not change.

$$\mathbf{CT} = \begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{- inverse object translation}$$

$$\mathbf{CR_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & sin\alpha & 0 \\ 0 & -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{- inverse rotation about the } x\text{-axis}$$

$$\mathbf{CR_y} = \begin{bmatrix} cos\beta & 0 & -sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ sin\beta & 0 & cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{- inverse rotation about the } y\text{-axis}$$

$$\mathbf{CR_z} = \begin{bmatrix} cos\gamma & sin\gamma & 0 & 0 \\ sin\gamma & cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{- inverse rotation about the } z\text{-axis}$$

The camera transform matrix is defined as

$$\mathbf{C} = \mathbf{CR_x} \times \mathbf{CR_y} \times \mathbf{CR_z} \times \mathbf{CT} \tag{3}$$

Note: The order of transform matrices matters.
The overall 3D to 2D projection matrix is:

$$\mathbf{M} = \mathbf{C} \times \mathbf{T} \tag{4}$$

3 Design Philosophy

There are no native 3D render engines available in the main stream web browsers so far, therefore, dojox.gfx3d will project 3D objects to 2D shapes, then render surfaces using dojox.gfx to mimic the lighting and texture. dojox.gfx3d is built upon dojox.gfx, so it is possible for us to manipulate the underlying 2D render engines via dojox.gfx interface. We would like to deliver the functionalities in one implementation instead of differentiating SVG and VML.

3.1 Viewport

dojox.gfx3d.Viewport (*Viewport* in the following) is the container for all 3D objects. The Viewport is logically another dojox.gfx.Group object. This inter-operation helps to integrate dojox.gfx3d into dojox.gfx and improve the code reuse. Pragmatically, users may aggregate more than one Viewport into Surface, a good candidate for subgraphics.

Viewport is derived from the dojox.gfx.Group, and created by *dojox.gfx.Surface.createViewport*.

```
dojo.declare("dojox.gfx3d.Viewport", dojox.gfx.Group,  ... ...

dojo.extend(dojo.gfx.Surface, {
    createViewport : function() {
        return this.createObject(dojox.gfx3d.Viewport);
    }
```

The following interfaces are exposed to create 3D objects[3]:

```
dojox.gfx3d.Viewport.createLine(line)
dojox.gfx3d.Viewport.createPath(path)
dojox.gfx3d.Viewport.createTriangle(triangle)
dojox.gfx3d.Viewport.createCube(cube)
dojox.gfx3d.Viewport.createCylinder(cylinder)
dojox.gfx3d.Viewport.createPipe(pipe)
dojox.gfx3d.Viewport.createCone(cone)
```

Viewport object manages the camera and lightings as well:

```
dojox.gfx3d.Viewport.setCameraTransform(matrix)
dojox.gfx3d.Viewport.applyCameraRightTransform(matrix)
dojox.gfx3d.Viewport.applyCameraLeftTransform(matrix)
dojox.gfx3d.Viewport.applyCameraTransform(matrix)

dojox.gfx3d.Viewport.addLighting(lighting)
```

Once the camera and/or lighting is chaned, all 3D objects inside the viewport have to be re-rendered. To eliminate unnecessary computation, the Viewport will **not** take action until the *render* is explicitly called. *render* will iterate all the children in the viewport and redraw them by calling each callback function *Object.render* with updated environment.

3.2 Object

dojox.gfx3d.Object(*Object* in the following) is the base class for concrete 3D objects. Object is derived from *dojox.gfx.Shape* to reuse the Group/Shape infrastructure from dojox.gfx. The following operations are added or overrided for the new functionalities:

setObject the equivalent of dojox.gfx.setShape, using this new name to avoid confusion.

[3]the default meta data of 3D objecs are discussed in 3.4

setTransform Using the dojox.gfx3d.matrix family to manipulate the transform matrix.

render a pure virtual function is used render itself using Viewport's environment, all concrete 3D objects should override it.

The following 3D objects are derived from Object:

- dojox.gfx3d.Line
- dojox.gfx3d.Path
- dojox.gfx3d.Triangle
- dojox.gfx3d.Cube
- dojox.gfx3d.Cylinder
- dojox.gfx3d.Cone
- dojox.gfx3d.Pipe

3.3 Scene

dojox.gfx3d.Scene(*Scene* in the following) is a container for Scene and other 3D objects, equivalent to *dojo.gfx.Group*. Users can group several 3D objects together and apply the world transformation to them in one shot. Scene also supports to create 3D objects.

3.4 Meta Object

Meta object[4] are the interface for user to create, set/get the attributes of 3D objects.

3.4.1 Line

Line is described by the two ends' coordinations: $(x1, y1, z1, x2, y2, z2)$.

3.4.2 Path

Path supports two mode as dojox.gfx.Path does, *Absolute* and *Relative*. In Absolute mode, the coordination is the absolute corordination in the 3D viewport mode, while in Relative mode, the coordination is the offset of current position. SVG convention is used as the drawing commands as dojox.gfx. The following drawing commands are proposed to be supported:

moveTo move current postion to the new position

lineTo draw a line from current position to a new position

curveTo draw a bezier curve in 3D space.

[4] Any suggestion for a new name of this?

3.4.3 Triangle

Triangle is the fundamental element to build more complicated 3D objects, described as a (x, y, z), b (x, y, z), c (x, y, z).

3.4.4 Cube

Cube[5] is defined by the two vertices in the diagram, uppper (x, y, z) and lower (x, y, z), each surface is perpendicular to one of axises.

3.4.5 Cylinder

Cylinder is defined as bottom (x, y, z), height and radius. *bottom* is the cororrdinations of the center of bottom surface.

3.4.6 Cone

Cone[6] is defined as top (x, y, z), height and radius. *top* is the coordination of the tip of cone.

3.4.7 Pipe

Pipe[7] may stir the interest of Chart developers. It is charactized by 3D Path with radius and texture. It could be a cornerstone to build professional pseudo chart.

3.5 Meta

3.5.1 Transform matrix

Transform matrix is defined as:

$$\begin{bmatrix} xx & xy & xz & dx \\ yx & yy & yz & dy \\ zx & zy & zz & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

so the 3D transform matrix in dojox.gfx3d is defined as a dictionary: $\{xx, xy, xz, yx, yy, yz, zx, zy, zz, dx, dy, d$

3.5.2 Lighting

TODO

3.5.3 Texture

The following textures are supported:

Solid Solid color $\{r, g, b, alpha\}$, alpha indicates the transparency.

Hollow Specialized Solid, aka $\{r : 255, g : 255, b : 255, alpha : 1.0\}$

[5] Or should we use *Box*?
[6] Can we render the surface using linearGradient to support Cone?
[7] Do we have problem to implement this?

4 Conclusion

dojox.gfx3d rocks.