SVG Extensions for 3D displays

Enabling SVG on auto-stereosciopic displays

Jean Le Feuvre

Researcher
Telecom ParisTech Multimedia Lab

Paris
46 rue Barrault
75013
France
+33-1-45-81-71-69
+33-1-45-81-71-69

Jean Le Feuvre is researcher and teacher at Telecom ParisTech. His research focuses on multimedia systems architecture, including authoring, transport and rendering. He is the project leader of the GPAC open-source project, a framework offering various tools for delivery and playback of multimedia content, including W3C SVG and MPEG-4 BIFS.

Abstract

In the past few years, several technologies have been commercially deployed for viewing video and games on television, computer or mobile displays, and standardization bodies such as MPEG or ITU have produced several technologies to deliver and compress video content for such systems. With the increasing adoption of 3D displays in the consumer market, the question arises whether standard multimedia technologies, as used on Internet, are fitted for stereoscopic displays. In this paper, we present a stereoscopic-aware multimedia system handling 2D and 3D content with stereoscopic effects and discuss the extensions introduced in SVG to properly address depth effects.

Table of Contents

Bibliography

Introduction

3D displays, allowing natural perception of depth by the viewer, are increasingly gaining acceptance, on mobile as well as on digital TV markets. 3D display technologies rely on stereopsis, the ability for the brain to reconstruct depth information from different views of an object on the two retinas. This effect enhances the depth perception given by other cues such as perspective (further objects look smaller) or motion parallax (further objects seem to move slower). In these displays, each view is redirected to the target eye through an optical system, such as lenticular or polarized sheets on top of the screen, or mechanical system such as liquid crystal shutter glasses ref1. While most displays are stereoscopic only systems (e.g., only two different views are presented), some systems, called auto-stereoscopic, may present several views of the content, providing horizontal parallax to the viewer, which allows looking around the objects on the screen. Both types of displays enable placing objects in front or behind the screen plane. The range of the perceived depth position of objects depends on the display physical characteristics: while quite large in 3D movie theater system, this effect is reduced with TV displays, even more with mobile displays. We refer to this volume as the "stereoscopic box".

Extensive research has been done in 3D video compression, and several approaches exist to represent 3D video data ref2:

- Stereo pair for each frame,
- Texture and depth information for each frame,
- A combination of the above.

The depth information, or depth map, gives the position of each pixel of the video on the z-axis. Depth information simplifies the process of generating views from different angles, for example when using auto-stereoscopic displays or when moving the 3D video further or closer from the viewer. The generation of different images based on depth and color information is called DIBR ref3 for depth-image-based rendering.

Most synthetic 3D representation languages, such as Collada or X3D, are already suitable candidates for usage with 3D displays, as they already carry depth information. This is however not the case with standard 2D multimedia used all over the web, especially (X)HTML or SVG: these language are depth agnostic and do not allow authors to use the stereoscopic nature of the screen.

In order to allow for conception of efficient and attractive 2D multimedia content for (auto-)stereoscopic displays, an author may want the multimedia language to be able to:

- display 3D video and 3D still images,
- display synthetic objects such as UI items or text labels at any depth,
- dynamically modify the depth range of 3D video and images as well as synthetic 2D graphics for visual effects,
- generate synthetic objects with arbitrary, non-uniform depth information.

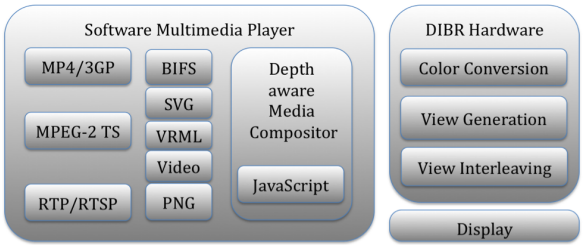
In this paper, we present a complete multimedia system handling stereo and auto-stereoscopic displays. The system is designed with low-end CE devices in mind, with a focus on 2D graphics and associated visual effects on stereoscopic displays. It furthermore integrates rendering of 3D data through software or a dedicated GPU. Our

approach is to re-use as much as possible existing technologies, without breaking the rendering or animation model of the underlying languages considered. Moreover, the porposed solution will have to be independent from the display type (stereoscopic or auto-stereoscopic).

The rest of this paper is organized as follows. Section 2 presents the architecture of the system. Section 3 details our contribution on depth effects in SVG content. Section 4 gives some notes on the implementation. We conclude this paper and present future work in Section 5.

System Architecture

Our system, described in Figure Figure 1, “Figure 1. Systems Architecture”, is divided in two core components: a software multimedia player and hardware DIBR renderer.



test caption.

Figure 1. Figure 1. Systems Architecture

Hardware Renderer

The BIBR hardware renderer is in charge of generating the different views from the different objects present in the multimedia scene. The choice for a DIBR hardware renderer was justified by its ability to handle stereoscopic as well as auto-stereoscopic displays easily, with a fixed computational cost independent from the number of views to generate. The hardware architecture derives from ref4.

The hardware manipulates texture objects decoded or generated by the software module. The texture color data can be in RGB or YUV formats. Each object can have its own depth or alpha 8-bit plane. It must be noted that the system does not currently allow for both depth and alpha planes at the same time, but instead may use a depth plane where the higher bit of each component acts as a shape mask. Each object has depth offset and depth gain coefficients, allowing redrawing of the 3D scene without modifying the textures and depth maps.

The hardware is also in charge of interleaving the pixels on the display e.g., placing (sub)-pixel of each view at the correct location depending on the optical system).

Multimedia Player

The multimedia player used is GPAC ref5. It can handle various multimedia languages such as SVG, VRML or BIFS, as well as video formats such as MPEG-4 SP or AVC/H264 and still images such as JPEG and PNG. Multimedia data can be read from various sources such as RTP, MPEG-2 TS and MP4/3GP files, commonly used in Digital TV and Mobile worlds.

The software is in charge of decoding video, images and multimedia scenes. The compositor task can be decomposed as follows:

- Compute spatial 2D and depth positioning of objects,
- If needed, generate synthetic graphics and depth maps,
- Send the objects to the DIBR hardware,
- Execute timers and user interactions.

Currently, the prototype hardware does not handle hardware accelerated vector graphics such as OpenVG, therefore the rasterization process is performed in software on off-screen surfaces which are then passed to the hardware. The rendering engine supports several declarative formats for interactive services(W3C SVG, MPEG-4 BIFS), which have all been extended to allow for depth information.

Although our prototype also supports 3D languages such as X3D, in the rest of this paper we will focus on the SVG language, especially the SVG 1.2 Tiny profile since our prototype is more oriented towards embedded systems.

Depth Effects

Objects in a typical depth-aware 2D interactive service can be separated in two categories:

- Objects with a constant depth where all pixels are at the same distance from the viewer. We refer to these objects as "flat objects"
- Objects with non-uniform depth, such as video, images or complex ynthetic graphics. We refer to these objects as "3D objects"

Depth Offset

The first depth effect to use on 2D content is the ability to position an object such as a menu item or a sprite in a game, at any depth. All pixels of this object will then share the same depth. This enables pushing 2D objects in front of the screen or behind the screen. This effect can also be used to adjust the depth of a set of objects with different depth offsets. This allows depth translation of some parts of the content, which is very useful in transitions and animations.

For the SVG case, we defined a new attribute "depth-offset" for that purpose. Although there is no restriction for the target element supporting this attribute, we restricted ourselves to the <g> element for implementation purposes.

Depth Scaling

Another useful depth effect is the ability to scale the depth values of an object. This is especially needed for objects with depth maps, as it allows stretching and shrinking effects on 3D objects without having to modify their associated depth map. It can be particularly

useful when UI elements get or loose focus, or combined with a color revealing/fading transition.For that purpose, we defined a new attribute "depth-scale", which is applied to the <g> element.

Following SVG transformation hierarchy design, "depth-offset" and "depth-scale" attributes of a <g> element are applied to its descendent. Nested <g> elements with depth properties can be used.

Depth Calibration

The depth values in our architecture are by default in the range -1.0, which is the farthest point viewable on the display z axis, to 1.0, which is the nearest point. Such a range may not always be convenient for an author; especially, being able to work with depth units closer to pixel units might be desired. We added a depth calibration attribute to the <svg> element, "depth-viewbox"; allowing an author to specify the minimal and maximum depth value used in the content. As with "viewBox" attribute, this attribute can be modified (scripting, <animate> element).

The following example shows the compination of the different depth rendering attributes introduced. The result is a rectangle floating in front of the screen and a text in front of the rectangle. The "depth-scale" attribute of the 'button1' <g> element can typically be animated on mouse or focus in/out.

```
<?xml version="1.0" ?>
<svg xmlns="http://www.w3.org/2000/svg"   xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:ev="http://www.w3.org/2001/xml-events" width="160" height="160"
        depth-viewbox="-100 100">
 <title>Depth UI item</title>
 <desc>Shows usage of  depth extension tools.</desc>

 <g id="button1" transform="translate(20,20)" depth-offset="30" depth-scale="0.5">
  <rect width="100" height="30" fill="blue"/>
  <g depth-offset="30">
   <text x="20" y="20">Click!</text>
  </g>
 </g>
</svg>
```

Depth Maps

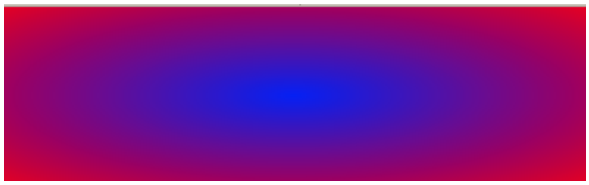
Objects with non-uniform depth are either bitmap-based objects coming from <video> or <image> elements, or synthetic graphics. In order to generate synthetic graphics with depth information, we defined a new SVG filter called "feDepthComponent".

This filter uses the RGBA source image and produces an RGBA+depth image to be used with the renderer. The depth image used is specified using a <feImage> as a secondary RGBA source, and one component of the secondary source is then transfered to the depth component of the filter result, using either <feFuncA>, <feFuncR>, <feFuncG>, <feFuncB>. This technique enables computing complex synthetic depth maps uncorrelated with the color information.

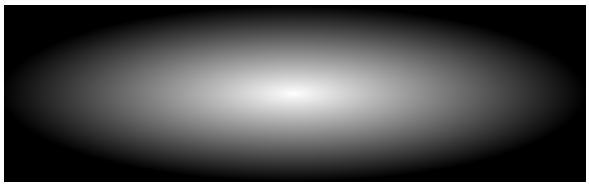
```
<?xml version="1.0" ?>
```

```
<svg xmlns="http://www.w3.org/2000/svg"   xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:ev="http://www.w3.org/2001/xml-events" width="160" height="160">
 <title>Depth Image</title>
 <desc>Producing a depth image with feDepthComponent</desc>

 <defs>
  <radialGradient xml:id="MyGradient" r="0.1">
   <stop offset="0" stop-color="blue"/>
   <stop offset="1" stop-color="red"/>
  </radialGradient>
  <rect xml:id="depthRect" fill="url(#MyGradient)" width="100" height="30"/>
  <filter xml:id="depthFilter">
   <feImage xlink:href="url(#depthRect)" result="depth"/>
   <feDepthComponent in="SourceGraphic" in2="depth">
    <feFuncR type="linear" slope="-1" intercept="1"/>
   </feDepthComponent>
  </filter>
 </defs>

 <g filter="url(#depthFilter)" transform="translate(20,20)">
  <rect width="100" height="30" fill="blue"/>
  <text x="20" y="20">Click!</text>
 </g>
</svg>
```

This code shows an example of the proposed feDepthComponent filter wich uses a red and blue gradient and composes the depth map from the red component of the gradient. Such a gradient could be used to simulate waving of a surface. The gradient looks like this



and the resulting depth map looks like this:



Since the proposed filter work in color coordinates, the depth calibration introduced previously is ignored for depth map and the components are automatically mapped to the interval [0, 1.0] where 0 is the far plane of the stereoscopic box and 1 the near plane.

Depth filter values are however affected by "depth-scale" and "depth-offset" attributes, which are remaped by the implementation to [0 1] range according to the "depth-viewbox" attribute.

Note that the <feDepthComponent> can also be used to add depth information to raster image or video, or replace existing depth information.

Experimental Results

In order to display 3D still images, we had to define new formats for images:

- PNGD: PNG where alpha channel acts as depth channel
- PNGDS: PNG where alpha channel acts as depth channel with the high-order bit acting as a shape mask

We have also demonstrated playback of video sequences according to the MPEG-C standard, with dedicated video and depth streams both coded in MPEG-4 Simple Profile. All the effects proposed have been successfully tested. We have even demonstrated a combination of 3D models (using X3D language) with 2D user interface using SVG depth effect and <foreignObject> element.

One problematic we faced during this work was the respect of the SVG rendering model. Using depth effects will indeed change the rendering order of objects, which is no longer compatible with SVG. The impact on the SVG viewing model have been discussed in the elaboration of the SVG Transform specification. Several approaches are possible:

- let the author organise the elements as needed, strictly following the painter's algorithm.
- let the player organise the elements as needed, with a z-sorting of all elements present in the composition
- allow a mix of both cases, where only subparts of the composition are z-sorted.

Our first implementation only supported the first case, and the author is responsible for avoiding overlaps of object. This also reflects our hardware architecture which doesn't perform depth sorting of the various objects.

We are currently working on the mixed use-case to hide this complexity from the author. While this is clearly not problematic for flat objects or non-overlapping objects, the problem is more complex when mixing objects with at least one non-uniform depth map, since they can overlap one another along the depth axis and can no longer be sorted according to their depth.

Conclusion

In this paper, we have identified the lack of support for 3D displays in most 2D UI and rich-media standards. This situation is problematic as more and more 3D displays (steroscopic or auto-stereoscopic) are deing commercially deployed by the home entertainment industry. We have defined the minimal depth effects that multimedia authors would need to adress these displays. We have presented an SVG multimedia player running on such displays, and proposed extensions to the language supporting the proposed effects. We have shown intergation of simple UI effects and more complex depth-map based 3D effects, and shown the problematic of using objects with complex depth-map in the SVG rendering model. In future work, we will further investigate this problematic and extend our work to support SVG Transform. We will also conduct user trials to evaluate the visual quality of the proposed effects.

Bibliography

[ref1] Pastoor Siegmund. Matthias Wopking. “3-D Displays: A review of current technologies”. http://dx.doi.org/10.1016/S0141-9382(96)01040-2. Copyright © 1997. 100-110. *Displays*. 17. 2.

[ref2] A. Smolic. K. Mueller. P. Merkle. P. Kauff. T. Wiegand. “An overview of available and emerging 3D video formats and depth enhanced stereo as efficient generic solution”. http://dx.doi.org/10.1109/PCS.2009.5167358.. Copyright © 2009. *Picture Coding Symposium*.

[ref3] C. Fehn. K. Mueller. P. Merkle. P. Kauff. T. Wiegand. “Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV”. http://dx.doi.org /10.1109/PCS.2009.5167358.. Copyright © 2004. 93-104. *Proceedings of the SPIE Stereoscopic Displays and Virtual Reality Systems XI*.

[ref4] A. Bourges. J. Gobert. F. Bruls. “MPEG-C part 3: Enabling the introduction of video plus depth contents”. Copyright © 2006. *Proceedings of IEEE Workshop on Content Generation and Coding for 3D-TV*.

[ref5] J. Le Feuvre. C. Concolato. J.C. Moissinac. “GPAC: open source multimedia framework”. http://doi.acm.org/10.1145/1291233.1291452. Copyright © 2007. *Proceedings of the 15th international Conference on Multimedia*.