Enriching the web with CSS Filters

Raul Hudea
Adobe Systems
Bucharest, Romania
rhudea@adobe.com

Rik Cabanier
Adobe Systems
San Francisco, CA 94103
cabanier@adobe.com

Vincent Hardy
Adobe Systems
San Francisco, CA 94103
vhardy@adobe.com

ABSTRACT

Filters in the web world are not a novelty, as they were already available for SVG content. The ability of applying filters on various SVG elements provided the developers with sophisticated rendered effects. With CSS Filters, it will be to be possible to style HTML elements by using filter effects. By default, the CSS Filters offers a good selection of predefined filters (like blur, sepia, grayscale) which allow developers to quickly make their HTML application more awesome. For the ones who want more control on how a filter should apply to a HTML element, CSS Shaders are a way as they provide the flexibility and expressivity needed to created arbitrary effects. This paper will look at how CSS Filters and CSS Shaders works and how can be used to create unique HTML content.

Categories and Subject Descriptors

I.3.6 [**Methodology and Techniques**]: Standards

General Terms

Design, Specification

Keywords

HTML5, CSS Filters, CSS Shaders, CSS Compositing

1. INTRODUCTION

The graphical and interactive richness of HTML has greatly improved in the recent years. CSS Transforms, transitions, animations, reflections, shadows, have all contributed to what we currently expect of a HTML5 application. The original SVG Filter effects specification is currently being transformed into a common specification covering CSS and HTML in addition to SVG.

2. CSS FILTERS

The filter effect is a way of applying that effect on an image buffer. The way this works in the HTML world is that an element (including its children) are rendered into a buffer. On that buffer, the filter effect is applied, and then that buffer is composited into the elements parent.

WWW2012 Developer Track, April 18-20, 2012, Lyon, France.

Figure 1: Predefined CSS filters

To apply a filtering effect on a HTML element is quite simple. Here is how to apply a *blur* effect on an HTML element:

```
<style>
#elem { filter: blur(1); }
</style>
```

The CSS *filter* property allows the use of one or a chain filters that should be applied. It is designed to be used with CSS Animations/Transitions.

The filter property can use a number of predefined filter functions: *blur*, *drop-shadow*, *gamma*, *grayscale*, *hue-rotate*, *invert*, *opacity*, *saturate*, *sepia* and *sharpen*. These filters can be used either as standalone (as in the example above) or can be combined as a chain of filters where the output of the previous filter is the input of the next one.

2.1 CSS Shaders

But even if the default filters can be chained, it is clear that they can not satisfy the need for creating unique ways of rendering HTML elements. In order to satisfy this need, *filter* accept a *custom* filter which allows the usage of custom written *vertex shaders* and *fragment shaders*.

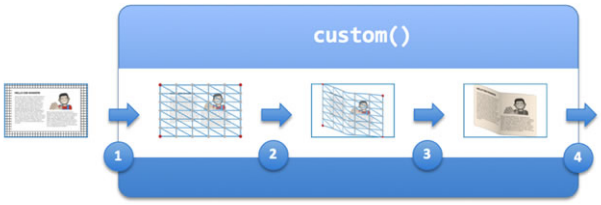

Figure 2: Custom CSS shaders processing model

A shader is a (typically) small program that process the vertices of 3D geometry *vertex shaders* and the color of pixels *fragment shaders*. For example, a *fragment shader*, also called *pixel shader*, can make arbitrary computations to determine the pixel's color, while a *vertex shader* can create a waving flag effect on a surface. By being used in 3D graphics, these shaders can be hardware accelerated and CSS shaders harness this power.

The custom filter are used similarly like the predefined ones:

```
filter: custom(url('vertex.fs'),
               url('fragment.fs'),
               vertexMesh,
               params);
```

2.2 Writing CSS Shaders

The language in which *vertex shaders* and *fragment shaders* are written, is the OpenGL ES shading language, which is also used for WebGL, another web related technology. The shading language is designed to make programming visual effects easy and exposing this to CSS offers a tie to this expressivity into the CSS syntax and make it easy to use and animate these effects.

It is not difficult to write shaders, but it takes some practice to do it and it is a lot of fun to author. Here is a sample of a *color inverter* shader:

```
precision mediump float;
// The 'original' content rendering in a texture.
uniform sampler2D s_texture;
// Shader parameters
uniform float amount;
varying vec2 v_texCoord;
void main() {
 vec4 color = texture2D(s_texture, v_texCoord);
 vec4 icolor = vec4(color);
 if (color.a > 0.0) {
   icolor.rgb /= icolor.a;
 }
 icolor = vec4(1.0 - icolor.r,
               1.0 - icolor.g,
               1.0 - icolor.b,
               icolor.a);
 icolor.rgb *= icolor.a;
 gl_FragColor = (1.0 - amount) * color + amount * icolor;
}
```

Figure 3: Custom shaders in action

2.3 Current status

The predefined CSS Filters (like blur, gamma...) are available in developer versions of Google Chrome browser and nightly versions of WebKit[1]. Since December 2011, these filters are hardware accelerated, taking advantage of the GPU power available on the machine. The hardware acceleration path is currently available only on WebKit for Mac and Google Chrome[2].

Support for CSS Shaders is currently available in both WebKit[3] and in nightly versions of Google Chrome (called Canary builds)[4]. Currently, there is no hardware acceleration path for any browser, but we are working to enable this for Google Chrome.

3. CSS COMPOSITING AND BLENDING

For years, designers have use advanced blending modes such as multiply and hard light and advanced to create visually compelling designs. The *CSS compositing and blending* specification seeks to brings this richness to web pages through a couple of simple CSS keywords.

Just as with CSS filters, this new specification can be used in HTML as well as SVG content and will act as a shorthand to the existing SVG filters that deal with compositing and blending.

The intent is to match the names, algorithms and visual representation of existing blending modes so people can reuse their existing skill set.

Another goal of the spec is to provide an easy to understand description on how blending, compositing and possibly color

[1]WebKit is the underlying HTML engine for Apple Safari and Google Chrome. See http://webkit.org
[2]If the `--enable-accelerated-filters` command line switch is used
[3]If WebGL support is enabled
[4]If the `--enable-css-custom-filter` command line switch is used

management are done so people that are not familiar with the subject matter can understand the design decisions.

4. CONCLUSIONS

As the CSS Filters and Shaders will become available in browsers, it will enable designers to come with idea that will surely blow our mind. By using custom CSS Shaders will allow developers to create new ways of interacting with the content, and, who knows, maybe a new ecosystem will appear around custom CSS Shaders.

5. REFERENCES

[1] V. Hardy. Introducing css shaders: Cinematic effects for the web. Tutorial, Adobe Systems, Oct. 2011. http://www.adobe.com/devnet/html5/articles/css-shaders.html.

[2] P. of the W3C CSS and S. W. Groups. Filter effects (FILTERS) 1.0 specification. W3C editor's draft, W3C, Dec. 2011. https://dvcs.w3.org/hg/FXTF/raw-file/tip/filters/index.html.

[3] The Kronos Group Inc. The OpenGL ES Shading Language http://www.khronos.org/files/opengles_shading_language.pdf

[4] J. Ferraiolo and D. Jackson. Scalable vector graphics (SVG) 1.1 specification. W3C recommendation, W3C, Jan. 2003. http://www.w3.org/TR/2003/REC-SVG11-20030114/.

[5] WebKit, HTML rendering engine http://www.webkit.org

[6] Chromium browser http://www.chromium.org/Home