

Aesthetic and Practical Concerns in the Drawing of Euler and Venn Diagrams: Case Studies using SVG

David Dailey

Computer Science Department, Slippery Rock University, Slippery Rock, PA, USA
david.dailey@sru.edu

Abstract. SVG is a graphical language standardized by the World Wide Web Consortium for the display of vector graphics on the WWW. It would appear to be precisely appropriate for drawing Euler and Venn diagrams so that those drawings would scale properly to different display devices. This paper presents techniques for using SVG for doing exactly this as well as investigating a variety of aesthetic considerations that might improve the “explanatory power” of such diagrams. Such features as the smoothness and convexity of curves, the angles and distances between their intersections, the areas of resultant regions, colors and other markings of the curves, as well as how to bring about these effects are considered.

1 Introduction

SVG or Scalable Vector Graphics has been the World Wide Web Consortium’s (W3C) recommended standard for web-based vector graphics since 2001 [1]. Its implementation by web browsers, however, has been somewhat slow, with Opera, Firefox, Safari and Chrome all providing some support during the years 2005 to 2008, but Microsoft not providing support, finally, until 2011, with the release of Internet Explorer 9 [2].

SVG is an XML language that provides formulaic descriptions of graphics that allow quicker download times than bitmapped formats (such as GIF, JPEG and PNG), rescaling to the device resolution (so that clarity may be preserved as an image is zoomed in upon or printed), and interactivity (meaning that images may be modified dynamically through JavaScript). By being a W3C standard, SVG ensures interoperability with the other web standards such as HTML, CSS, DOM, JavaScript and AJAX.

It is a language particularly well-suited to the creation of mathematical illustrations such as graphs [3], Voronoi diagrams [4], and, as will be argued in this paper, Euler and Venn diagrams. In part, this advantage owes to the portability and scalability of this form of graphics (supported by all modern web browsers and about 1 billion cell phones [5]), but SVG also brings advantages toward addressing certain aesthetic concerns, as these diagrams are used as explanatory constructs in teaching disciplines

ranging from Computer Science to Psychology and Philosophy [6] (<http://www.jfsowa.com/ontology/ontoschar.htm>). This paper will also present some of these aesthetic concerns and will critically examine SVG's suitability for addressing them.

Overall, the paper will conclude that while SVG is very appropriate and, in fact, easy to use to create illustrations that will be of value to the diagramming community, some of the more advanced features of the language (like filters and animation) still suffer from inconsistent support across browsers, while other aspects of the language appear to suffer from under-specification within the W3C specification. It is hoped that some of these concerns can be addressed within the nascent SVG 2.0 specification currently under development by the W3C.

2 Several Aesthetic Concerns for the Presentation of Set Theoretic Diagrams

Intrinsic to the study of Venn and Euler diagrams are several terminological distinctions and a few presuppositions that seem to be motivated by aesthetic considerations. For example, the concept of a *simple Venn diagram* [7] is based on the notion that no more than two curves should intersect at a given point. The very definition of Venn diagrams begins with the concept of the *simple closed curve*, despite the fact that certain set relations might be better represented by complex closed curves such as shown in figure 1. And included in the basic definition of Venn diagrams is another primarily aesthetic concern: the notion that all distinct regions in the plane should have differing inclusion relations (i.e., that no two subregions should be subregions of exactly the same collection of regions). We also, generally speaking, prefer smooth closed curves rather than ones with sharp edges, and prefer the regions each to be "large enough" (certainly to be visible).

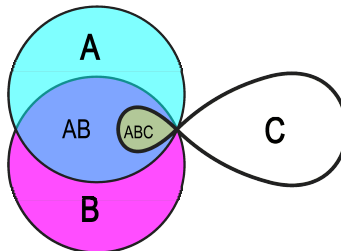


Fig. 1. An Euler diagram in which the absence of certain regions (AC and BC are null) motivates the use of closed curves that are not simple. This image is available in vector format at <http://cs.sru.edu/~ddailey/svg/NotSimple.svg>.

This section will discuss several aesthetic concerns that can be, in some sense of “perceptual distinguishability,” be added naturally to these diagrams to enhance their “readability.”

2.1 Aesthetic Criteria for Venn and Euler Diagrams

1. Curves should be, when possible, simple closed curves. The example of Figure 1 illustrates a situation in which, for sake of accurate adherence to the set relations, this may not be possible.
2. Curves should be smooth. Certainly the investigation of polygonal diagrams is of interest to combinatorists, but smooth curves are easier for the eye to follow.
3. When possible, curves should be convex. This appears not always to be possible, and the constructions of n -Venn diagrams, for any n , by both Venn and Edwards [7] rely on curves that have numerous concavities. But curves that do not have frequent changes of direction (from convex to concave) are intuitively easier for the human eye to follow.
4. The number of intersections between any pair of curves should be not only finite, as required by the standard definition, but small. Once again, this relates to the perceptual ease of understanding a diagram, though two completely convex curves may intersect at arbitrarily high numbers of points (as, for example, two regular n -gons each centered at the same point, with one slightly rotated about the center).
5. The number of curves that meet at a given point should be minimal. While many inquiries of Venn diagrams seek to minimize the number of intersections in the entire diagram (to make the graph of the diagram have fewer nodes), such diagrams are harder for the eye to traverse and must often rely on additional visual cues such as color to make the various curves distinguishable. According to this criterion, diagrams whose graphs are four-regular are preferred.
6. The points of intersection between regions should be relatively far apart, when possible. Clearly, if more than two curves intersect at a given point, then the distance has been eliminated altogether, but again, for ease of perception, it is best to ensure that these intersections are not so close together that confusion should result.
7. When two or more curves intersect at a given node, the angles of intersection should be “relatively high.” For example, in the four diagrams in Figure 2, diagram 1, is the relatively “classical” presentation, with each circle passing through the center of the other, producing an angle of intersection of sixty degrees. In Figure 2, the angles between the curves is maximized at ninety degrees. While it is arguable that diagram 1 may be preferable to diagram 2, this would probably stem from the relative sizes of the regions, a factor addressed in criterion 8 which follows.
8. The regions of the Venn diagram should be relatively “large.” While in certain contexts, we may wish to ensure that the areas of regions is proportional to the cardinality of the sets and intersections represented by the diagrams [8], absent this empirical constraint, it is expected that the human perceiver will more readily make sense of the subset and overlap relations when the regions are large enough to be labeled, shaded, or colored, and certainly, large enough to be seen.

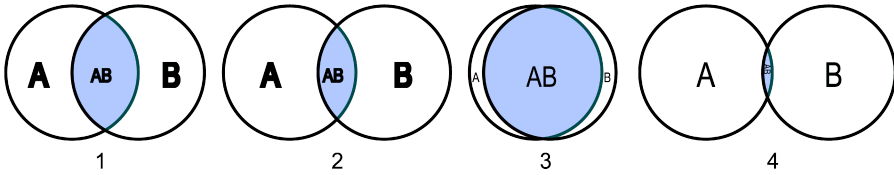


Fig. 2. Diagrams illustrating different sizes of regions and angles of intersection. This image is available in vector format at <http://cs.sru.edu/~ddailey/svg/intersection.svg>.

9. Regional markings (coloration, patterning or shading) should maintain “family traits.” That is, each of the subregions of a curve should bear some resemblance to one another, and certainly to the “undiluted” pure set as it appears outside the intersections with other sets. In Figure 3, the first two diagrams (at upper left) present the traditional additive and subtractive color models as rendered in SVG. And though, each subregion *inherits* its chroma from its parents, this fact is rather lost on many human perceivers who see no particular resemblance between black or white and the surrounding colors.
10. Notwithstanding criterion, 9, the markings (coloration, patterning or shading) of regions should, nevertheless, be maximally distinguishable, from one another, particularly when regions are adjacent. In the sixth diagram of Figure 3 (lower right) the four lowermost regions within it are not so perceptually distinguishable as we might prefer.

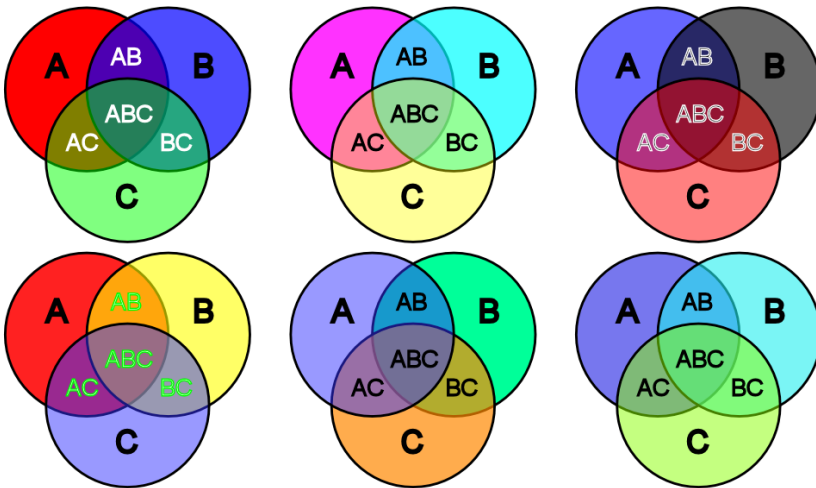


Fig. 3. Various Venn diagrams marked with different colors in an attempt to satisfy criteria 9 and 10. This image is available in vector format at cs.sru.edu/~ddailey/svg/V11.svg.

Among these ten criteria are many principles that rather work against one another. Distinguishability of regions (criterion 10) and family resemblance (criterion 9) call for opposing solutions. Likewise, as seen in Figure 2, maximizing the angles of intersection of curves, is in contrast with maximizing the areas of the regions inside, particularly for the preferred convex curves. This, however, is the nature of aesthetics: a compromise, ultimately between several competing criteria. One seeks to minimize confusability at several different parts of a rather complex psychophysical realm. In subsequent sections, we'll look at how some approaches to working in this realm using SVG.

3 Basics of Using SVG

Those readers familiar with HTML will find SVG quite natural. It is a markup language complete with tags (inside angle brackets “<” and “>”) and attributes. The tags of SVG are typically geometric things like `<line>` `<circle>` `<rect>` (for rectangle), and the attributes are things like `r=20` that might specify the radius of a circle, or `width=200` that might specify the width of a rectangle. Sometimes, for sake of reusing code, we may group objects together using the group element `<g>`, and reuse them using the `<use>` element. SVG also has transforms (for rotation, translation and resizing), patterns (for specifying customized patterns for use in filling curves), and filters (for modifying the way that regions appear).

SVG is visible in all modern browsers (meaning that you will have to use Internet Explorer 9 or 10 to see it) and many mobile devices (including i-phones, Androids that run IceCreamSandwich, Blackberry Playbooks, and dozens of others) for a total number of devices estimated at 1 billion [5]. However the SVG specification [9] is complex and has taken several years for most browsers to implement. Currently, support is quite thorough and consistent for basic SVG, but is a bit spotty for more advanced features like filters and animation [10].

To get started, one may copy the following code (also available at <http://cs.sru.edu/~ddailey/svg/simplest.svg>) into a text editor and save it as *simplest.svg*. Once saved in a place that is visible to web browsers (typically, a web server, though one's own hard drive will generally work), then one may simply open the file with a (modern) browser to see.

```
<svg xmlns="http://www.w3.org/2000/svg">
<circle r="50"/>
</svg>
```

This should succeed in drawing a black circle centered at the extreme upper left corner of the web browser. To some extent, it is the simplest SVG document possible.

Much more about how to use SVG to draw far more elaborate illustrations, and even interactive web documents, can be seen in the W3C's SVG Primer [2].

4 Drawing Curves: using circles, ellipses and paths.

SVG has several drawing primitives, including polygons, rectangles, and the like, but for these purposes we will concentrate on how to draw smooth curves: circles, ellipses and paths.

4.1 Drawing circles

While the above *simplest* code adds only one attribute (*r* for radius) to the `<circle>`, more attributes may be added to vary its center, the colors of both its boundary and interior, and other aspects. As shown in Figure 4, the coordinate system places the origin (0,0) at the top left of the browser window, with *x* increasing to the right and *y* increasing downward.

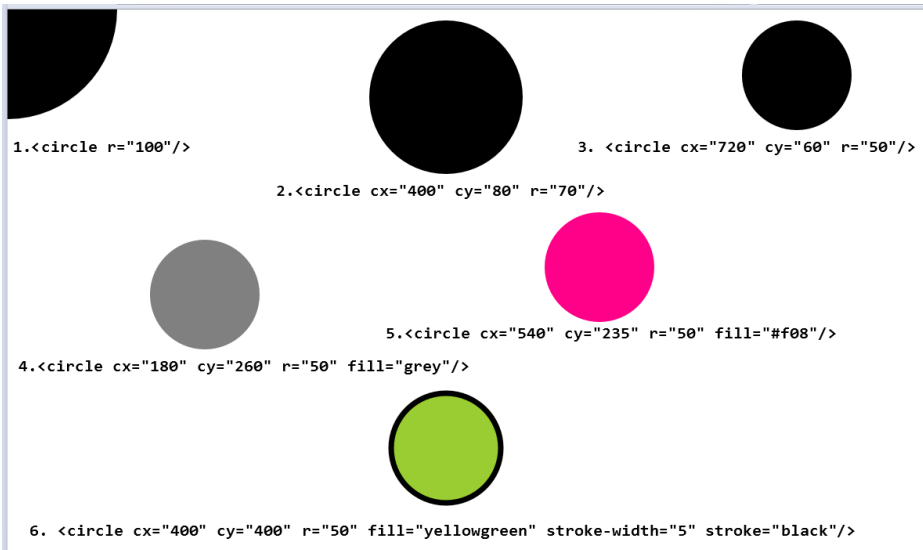


Fig. 4. Circles drawn with increasingly more complexity and attributes. This image is available in vector format at cs.sru.edu/~ddailey/svg/circles.svg .

As can also be verified from the above, *fill* controls the color (or gradient or pattern) that is placed into a shape; its default value is *black*. If the center of the circle (*cx,cy*) is not specified, then it is assumed to be (0,0).

Two other things about circles (and by extension the more complex shapes like ellipses and paths) in SVG should be known. We may make the interior of a circle invisible by setting `fill="none"` and we may vary the opacity of the stroke and the fill, independently as shown in Figure 5. Both of these will prove useful in drawing of set theoretic diagrams.

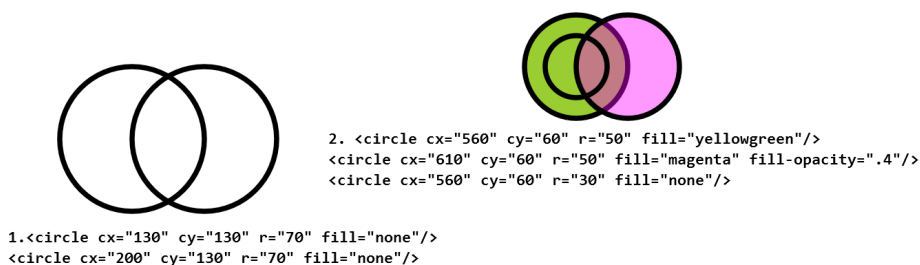


Fig. 5. Making a circle empty or partly transparent. This image is available in vector format at cs.sru.edu/~ddailey/svg/opacity.svg.

The magenta circle in the second diagram of Figure 5 has its opacity set to .4. Hence it will be 40% opaque and 60% transparent. The stroke-opacity attribute has not been adjusted, meaning that its value remains the default: 1.0, or 100% .

4.2 Ellipses

Ellipses are drawn in SVG much like circles except for having two radii: r_x , and r_y representing the distances from the center to the horizontal and vertical extremities. And unlike circles, rotation is relevant for ellipses, allowing us, for example, to draw, Venn's symmetric construction of one of the two 4-Venn diagrams. This example shows some of SVG's power for creating these sorts of illustrations.

In Figure 6, an initial ellipse is drawn. Because other ellipses will share the same stroke-width (3), stroke color (*black*) and the same fill (*none*) , it is nestled inside a group (a `<g>` tag) so that it, and all other versions of it, may inherit these shared attribute values. So that the drawing will have vertical symmetry, the ellipse is then rotated 60 degrees, about its center (250,150), using the *transform* attribute:

```

<g stroke="black" fill="none" stroke-width="3" >
  <ellipse id="E" ry="55" rx="100" cy="150" cx="250"
transform="rotate(60, 250,150)"/>
</g>

```

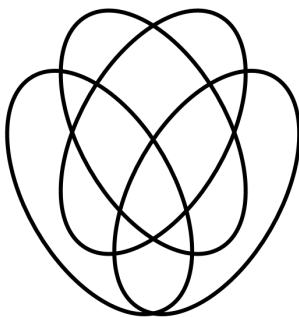


Fig. 6. The sixteen regions of a 4-Venn diagram drawn with congruent ellipses. The SVG version can be seen at cs.sru.edu/~ddailey/svg/fourVenn1.svg.

Next the ellipse is cloned using the `<use>` element, with the resulting code appearing as this:

```
<g stroke="black" fill="none" stroke-width="3" >
  <ellipse id="E" ry="55" rx="100" cy="150" cx="250"
transform="rotate(60, 250,150)"/>
  <use xlink:href="#E" transform="translate(-40, 45)"/>
  <use xlink:href="#E" transform="rotate(60, 250,150)"/>
  <use xlink:href="#E" transform="translate(40, 45) ro-
tate(60, 250,150)"/>
</g>
```

The initial ellipse has been given an id (“E”) which can be referred to later (as #E in the *use* elements). There are three reuses of the ellipse named E: one is simply translated down 45 pixels and leftward 40 pixels. Another one is rotated an additional 60 degrees about its center. The final one is both rotated and translated. In SVG multiple transforms are performed from right to left, so that this last ellipse is first rotated 60 degrees from the original, and then repositioned into alignment. Another way of efficiently reusing code can be used in the construction of the radially symmetric 5-Venn diagram, shown in Figure 7, using the `<replicate>` tag, that is being considered for addition to SVG 2.0.

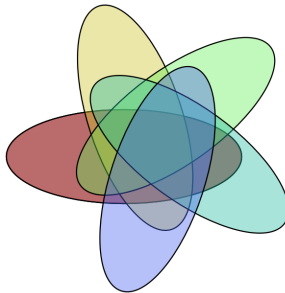


Fig. 7. A radially symmetric Venn diagram on 5 curves, drawn with only 10 lines of markup using `<replicate>`. The SVG version can be seen at <http://cs.sru.edu/~ddailey/svg/repVenn3.svg>

4.3 Paths

The use of SVG paths is particularly useful for drawing smooth curves with concavities. Using the `<path>` element, the diagram author has complete control over the slope of curves at the specified endpoints.

We will illustrate two uses of SVG paths, including the components for both quadratic and cubic Bézier curves, to draw a figure eight and a trefoil.

In a nutshell, SVG paths (see [11]) shape a curve using components based on lines, circular arcs, and Bézier components. The Bézier components allow us to draw, between any pair of points, a curve that visits both points and does so from any desired

angle. Let us consider just a figure-eight curve such as shown in Figure 8, below (and at <http://cs.sru.edu/~ddailey/svg/figureEight.svg>).

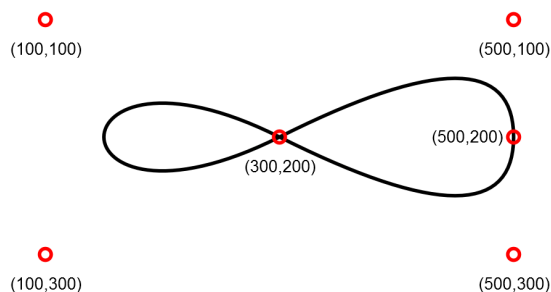


Fig. 8. Drawing a figure-eight using quadratic and cubic Bézier segments.

The code for this figure is as follows:

```
<path d="
  M 300,200
  Q 500,100 500,200
  Q 500,300 300,200
  C 100,100 100,300 300,200
"/>
```

Note that the curve begins at the point, 300,200, as marked and then progresses “in the direction” of the point 500, 100, finally ending up (after the first “Q”) at 500,200. From there, it now heads south in the direction of 500,300, ultimately returning to 300, 200. That is, the two quadratic components of the curve are controlled by the collinear points (500,100) and (500, 300). The slope of the curve as it begins at (300,200) is $\frac{1}{2}$ and, as it returns to (300,200) it is $-\frac{1}{2}$. At the midpoint of its double quadratic swoop, the path is momentarily vertical. The cubic component while similar, uses two control points (chosen to be collinear with those of the quadratic component) so as to assure that its slopes at (300,200) are likewise $\frac{1}{2}$ and $-\frac{1}{2}$. This demonstrates how we might stitch smooth segments together while controlling their slopes at key points such as intersections.

Circular arc components are conceptually a bit simpler, however, given two points and a radius of a circle that passes through both, there are, in fact, four circles that satisfy those constraints. Thus, the syntax for drawing arcs specifies with a few attributes which of those four curves is actually intended. (The interested reader is directed to the illustration at [12])

Without going into detail about this syntax, observe from Figure 9, that we may use circular arcs to adjoin line segments into smooth curves (in the sense of being continuously differentiable.)

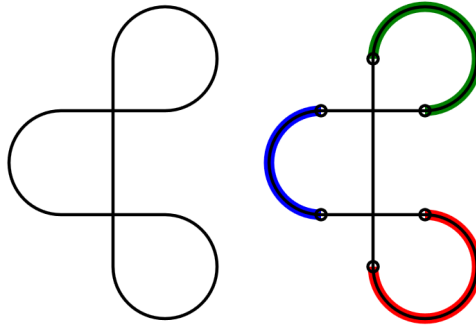


Fig. 9. A curve with three linear components and three circular components.

The curve at left in Figure 9 is drawn via the following code:

```
<path d="M 150,300    L 250 300
        A 50 50 0 1 1 200 350
        L 200 150
        A 50 50 0 1 1 250 200
        L 150 200
        A 50 50 0 0 0 150 300
"/>
```

while the curve at the right emphasizes the linear components (L) and the arc components (A) as well as the six connecting endpoints $\{(150,300), (250,300), (200,350), (200,150), (250,200), \text{ and } (150,200)\}$.

Without much difficulty, one can interrupt this curve at key points of overlap to create a roadway with bridges as shown at <http://cs.sru.edu/~ddailey/svg/notknot3.svg>.

By redirecting some of the arcs of Figure 9, we may “reattach” some of those arcs, as shown in the following code to produce the shape on the left in Figure 10:

```
<path d="M 150,300    L 250 300
        A 100 100 0 0 0 340 245
        A 100 100 0 0 0 250 100
        A  50  50 0 0 0 200 150
        L 200 350
        A  50  50 0 0 0 250 400
        A 100 100 0 1 0 250 200
        L 150 200
        A 50 50 0 0 0 150 300
" />
```

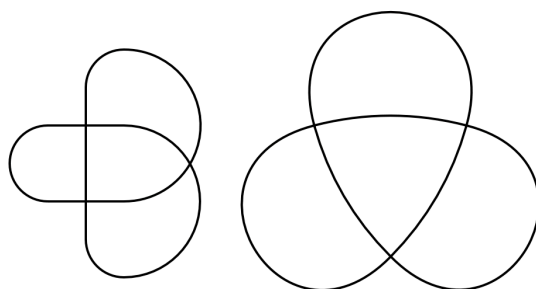


Fig. 10. Two versions of the trefoil. Left: using circular arcs; right: using Bézier curves.

The code for the diagram on the left of Figure 10 can then be generalized as shown in the diagram on the right, by using Bézier curves and making sure, as shown earlier in Figure 8, that the curve remains smooth.

```
<path d="M 200,200 Q 300, 174.12 400,200
C 600, 251.76 444.82,520.52 300,372.2
Q 227.59,299.54 200,200
C 145.82,0.82 455.18,0.82 400,200
Q 372.41,299.54 300,373.2
C 155.18,520.52 0,251.76 200,200
"/>
```

Traffic diagrams flowing along these curves, when drawn as knots, can be seen at cs.sru.edu/~ddailey/svg/knot1.svg and cs.sru.edu/~ddailey/svg/knot2.svg, while a similar illustration with Borromean rings can be seen at cs.sru.edu/~ddailey/svg/rings.svg.

5 Marking territory: shades, colors and patterns.

A cursory examination of Euler diagrams sampled in situ from the web, suggests that color (either as what SVG would call *fill* or *stroke*) is used to aid the viewer in “reading” the diagram. The concepts that such diagrams are used for (for example differentiating between the British Isles and the British Islands¹ [13], are frequently fairly complex (why else would we need a picture?), and as such helping the reader to parse the visual information seems to be one of the only uses of color (as articulated in Aesthetic Principles 9 and 10): namely to help recognize family traits (that subregions inherit from their regions) and to illustrate the individual differences (that distinguish subregions from their various containers, subsets and siblings).

However, color need not be the only way of doing this, and several centuries of printing with black ink have undoubtedly helped to inform the ways of “illustrating overlap” between geopolitical regions with and without the use of color. Techniques

¹ One might guess this to be a particularly difficult set of distinctions for American speakers of English.

of shading, coloring and patterning have been investigated by researchers in a broad array of fields (as instantiated by the visualization work of Ian McHarg and Edward Tufte). To what extent can some of these methods of conveying family resemblance and individual differences be expressed within SVG?

The simplest technique for combining markings of regions is to let their darkneses accumulate as more regions overlap, as shown in the code below and in Figure 11.

```
<g stroke-width="5" stroke="black" fill-opacity=".4"
fill="black">
<circle cx="240" cy="60" r="50" />
<circle cx="290" cy="60" r="50" />
<circle cx="265" cy="95" r="50" />
</g>
```

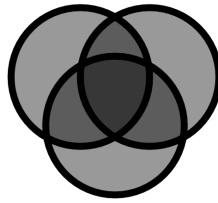
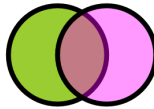
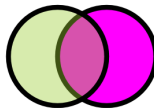


Fig. 11. Superimposition of partially opaque, simply shaded regions. The SVG version of this may be seen at <http://cs.sru.edu/~ddailey/svg/opacity3.svg>.

The problem with shading in this manner is that while it helps to differentiate “parent” regions from “child” regions, it does not help us distinguish things of the same generation. In SVG, objects maintain a stacking order from first to last declared. That is, two ovals, one magenta and one yellowgreen, and both 40% transparent, will not look the same, as in Figure 12. Thus, SVG imparts a bit of three dimensional asymmetry to its two dimensional objects.



```
1. <circle cx="560" cy="60" r="50" fill="yellowgreen"/>
<circle cx="610" cy="60" r="50" fill="magenta" fill-opacity=".4"/>
```



```
2.<circle cx="290" cy="270" r="50" fill="magenta"/>
<circle cx="240" cy="270" r="50" fill="yellowgreen" fill-opacity=".4"/>
```

Fig. 12. Effects of stacking order and opacity on appearance in SVG. SVG version may be seen at <http://cs.sru.edu/~ddailey/svg/opacity2.svg>.

Figure 13 investigates some combinations of color values and opacity values that seem to yield diagrams that are at least hint at satisfying principles 9 and 10. The obvious conclusion here is that the choice of colors and opacity values is somewhat critical toward producing a pleasant design, but it is not altogether obvious as to what might work. This is in part due to the fact that human color perception is complex. While most people have three differentially sensitive cones in the retina, some (mainly males) have only two, and some (mostly women) have recently been shown to have four. And while we are a predominantly trichromatic species at the level of the retina, we tend to be tetrachromats in the occipital lobe. Different cultures have differing numbers of primary color terms, though the boundaries between the primaries tend to coincide. Yellow is perceptually different than other primary or secondary colors and distance as calculated between two colors RGB values is not a good predictor of their perceptual distinctness. Couple all this with differing gamma curves for different monitors and the world of color is more complex than it might, at first glance, appear to be. Figure 13 also begins to raise and experiment, just a bit, with the interactions of labels and the regions underneath. Clearly, labels of regions must retain legibility, and that means that color contrasts must in some way be maximized, adding one more “readability” criterion to the Aesthetic Criteria already discussed.

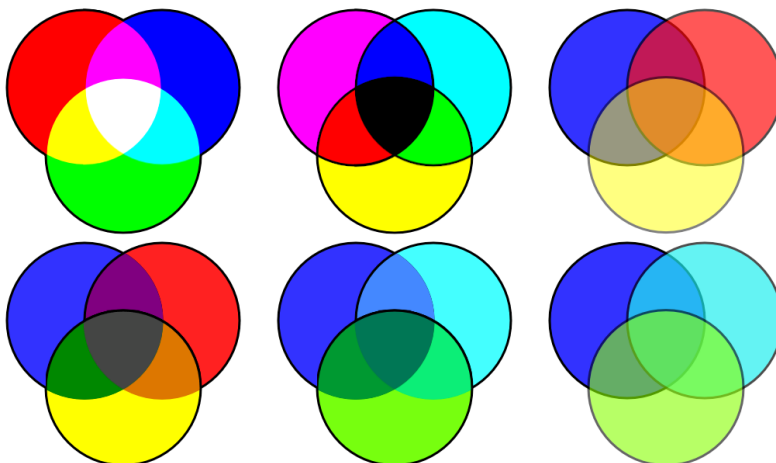


Fig. 13. Several differing color values offered to regions by varying the colors, opacities, and filtered combination of the original curves. (The original vector-based SVG file can be found at <http://cs.sru.edu/~ddailey/svg/V9.svg>).

As mentioned earlier, SVG also has the ability to use filters to allow, among other things, the interaction of colors to be more precisely controlled, or to allow the intersections of regions to be precisely targeted. In Figure 14, one can observe that higher inter-region color differences can be promoted by either using multiplicative or screening filters (top left and top center) or by using compositions filters to choose specific regions and, to it, apply any chosen color whatsoever. That is, in browsers that support advanced filters (currently only the Adobe plugin for Internet Explorer,

and Opera support these particular effects, though this is likely to change soon for both Firefox and Internet Explorer) we may exercise complete control over the coloration of the regions of a diagram, making the question of how to choose these colors all the more relevant.

While color is certainly a salient dimension of visual perception, it is not the only way to assign personality to shapes in SVG. Figure 14 shows several approaches using shapes and textures rather than just color to do so. These examples give rise quite naturally to a large set of complex questions about the distinguishability of various pattern glyphs, particularly in the context of multiple markings coalescing within regions that share features of two or more parent sets. That such information could be conveyed through tactile perception rather than through color, is a rather intriguing notion. While the meaningful primitives of color perception are relatively well studied, how might we most sensibly combine shapes in the manner? Is there a possibility of optimization is so complex a realm?

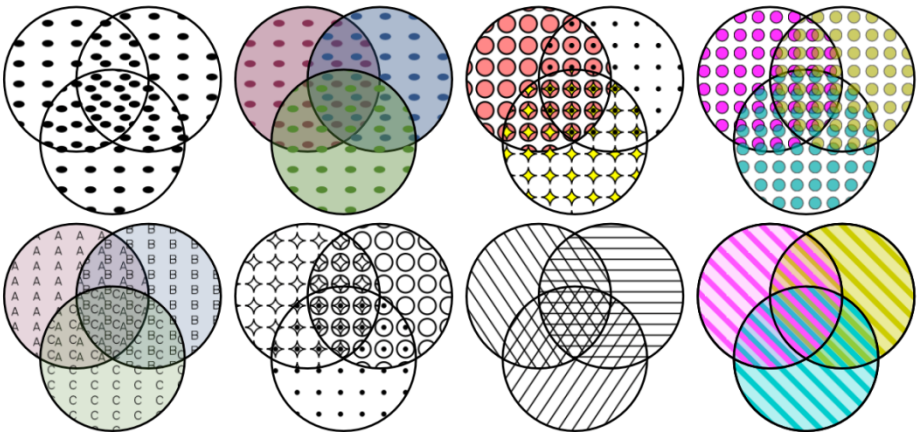


Fig. 14. Using SVG `<pattern>` to fill overlapping regions so as to preserve family traits. The SVG example can be studied at <http://cs.sru.edu/~ddailey/svg/V12.svg>.

6 Conclusion

Interrelations between the technology of drawing and the aesthetics of diagrams have certainly occurred since the early days of printing. The progression from engraving to half-tones to photography, color printing and web-based vector graphics has been a fascinating one. An interdisciplinary topic such as how best to portray the fundamental inclusion relations among a set of concepts is a topic as old, no doubt, as philosophy. SVG offers promise to those engaged in these studies, across many disciplines since the language offers both succinctness of expression, broad and growing standards support, and access to high level tools such as CSS, JavaScript and server-side web tools. At the same time, maximizing control over our diagrams can, in some cases, require some of the most advanced features the language has to offer. Browser support for some of these features remains spotty, but shows continual improvement

with new energy coming from major computing corporations as more parts of the SVG specification are implemented and as the specification itself shows very active growth and development.

References

1. Lilley, C.: W3C Scalable Vector Graphics (SVG) – History. W3C Interaction Domain, <http://www.w3.org/Graphics/SVG/History>
2. Dailey, D: An SVG Primer for Today's Browsers. W3C Working Draft — September 2010, <http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html> (2010)
3. Dailey, D., Elder, E., Perri, R.: A Browser-based Graphical User Interface for Designing and Manipulating Graphs. In: 7th Annual Conference, SVGOpen. W3C and SVG Working Group, Mountain View, CA <http://srufaculty.sru.edu/david.dailey/grapher/> (2009)
4. Bostock, M.: Voronoi Diagram, <http://mbostock.github.com/d3/ex/voronoi.html>
5. Dailey, D., Frost, J., Strazzullo, D.: Building Web Applications with SVG, Microsoft Press (2012)
6. Sowa, J.: Building, Sharing, and Merging Ontologies <http://www.jfsowa.com/ontology/ontoshar.htm>
7. Venn Diagram Survey: What is a Venn Diagram? The Electronic Journal Of Combinatorics (ed. June 2005), DS #5. <http://theory.cs.uvic.ca/~cos/venn/VennWhatEJC.html>
8. Micallef, L., Rodgers, P.: Drawing Area-Proportional Euler and Venn Diagrams using Ellipses, <http://www.eulerdiagrams.org/eulerAPE>
9. Dahlström, E., Dengler, P., et al.: Scalable Vector Graphics (SVG) 1.1 (Second Edition), W3C <http://www.w3.org/TR/SVG/> (2011)
10. Devaria, A.: When can I use...Compatibility tables for support of HTML5, CSS3, SVG and more in desktop and mobile browsers <http://caniuse.com/#cats=SVG>
11. Dailey D.: <path> in An SVG Primer for Today's Browsers, W3C (2010) <http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html#path>
12. Dailey, D.: <http://srufaculty.sru.edu/david.dailey/svg/newstuff/arcs.svg>
13. File:British Isles Euler diagram 15.svg, Wikipedia, http://en.wikipedia.org/wiki/File:British_Isles_Euler_diagram_15.svg