

Fractal CSS Design

An Introduction to Iterated Layout Patterns with Applications to Web Page Design

Dr. E. Garcia
admin@miislita.com

Published: December 28, 2009; Last Update: March 1, 2010

Copyright © E. Garcia

Keywords: fractals, css, design, iterations, layout, patterns, web pages, motifs

Abstract: This paper introduces a technique for generating CSS layout patterns based on fractal theory. Applications to the design of Web pages are presented.

Introduction

As mentioned in our series of articles on fractals, starting with *The Fractal Nature of Semantics* and subsequent articles (1 – 8), these patterns are found everywhere: in Mathematics, Chemistry, Information Retrieval, Literature, Economics, and dissimilar fields.

To revisit the topic, a fractal is something whose parts resemble the whole thing in some way when observed at different length scales or time scales. This is called self-similarity. A multifractal is something that exhibits several different fractal patterns. This is termed self-affinity.

Unlike in Mathematics where we can find all kinds of fractals and multifractals, in Nature we often find that these patterns display self-similarity or self-affinity at limited scales of length or time.

This is often the result of imposing some limits to the rules that generate the patterns. Such rules account for the number of iterations, scaling ratios, and type of affine transformations used. Sometimes, rules to account for randomness are included.

The basic unit of a fractal is called a motif. If the rules that define a motif are known, generating fractals reduces to recursively applying these rules back to the motif. By recursion (iteration) we don't mean making a mere copy of a motif, but that a modified version of it is found within itself. In Fractal Geometry we refer to this as a graphical iteration.

If fractals are everywhere, how could we find them in something apparently off-topic like the design of Web pages? Well, page layouts too are graphical shapes rendered by a special software called a browser. They can be visualized as the product of iterating a basic layout unit using few CSS rules. Let's have an example.

Assumption and Conventions

First, some assumptions and conventions:

For the sake of clarity, we will iterate a motif just once. All relevant markup will be discussed at a later time. I assume you are familiar with CSS and HTML. I also assume that:

- you know why we need to declare W3C xml and doctype instructions.
- title and metadata information goes in the head section of an HTML document.
- markup for the content of a document goes in the body section.
- whenever possible, CSS rules are declared in an external file and referenced through a link element placed inside the head section.

Fractal CSS Design Example

Step 1: Let's first delimit a working space.

Row 1 Content
Row 2 Content
Row 3 Content

Figure 1. Three rows.

Step 2: We also define a motif.

Left Content Content Content Content Content	Center Content Content Content Content Content Content Content Content Content Content Content Content	Right Content Content Content Content Content
---	---	--

Figure 2. Motif.

Step 3: We then place two instances of the motif between rows.

Row 1 Content		
Left Content Content Content Content Content	Center Content Content Content Content Content Content Content Content Content Content Content Content	Right Content Content Content Content Content
Row 2 Content		
Left Content Content Content Content Content	Center Content Content Content Content Content Content Content Content Content Content Content Content	Right Content Content Content Content Content
Row 3 Content		

Figure 3. Preliminary layout.

Step 4: Next, we iterate once by placing a scaled-down version of the motif into any section of itself. Let's place a reduced motif in the middle cell, between Rows 2 and 3.

Row 1 Content				
Left Content Content Content Content Content	Center Content Content Content Content Content Content Content Content Content Content Content Content			Right Content Content Content Content Content
Row 2 Content				
Left Content Content Content Content Content	Left Content Content Content Content	Center Content Content Content Content Content Content Content Content Content Content Content Content	Right Content Content Content Content	Right Content Content Content Content Content
Row 3 Content				

Figure 4. Final layout.

For the grand finale, we have the option of keeping the underlying motif away from the view by hiding its borders. More on this, later. At this point you might realize that all kind of tables can be generated with this approach. After all, a table (a grid) is just the result of iterating cells within a given space; no more, no less.

The tricky part is to find the set of CSS rules that goes with the intended motif. The rest is a matter of doing some monkey work (copy/paste) of HTML markup and some additional CSS tweaking. If your HTML and CSS are correct, any glitch in the rendering is probably due to the browser. Browsers do have glitches to work around.

Let us now describe the instructions that go with the above figures.

CSS and HTML Instructions

We first declare the W3C xml and doctype instructions and all basic markup for the head and body sections. This should not be optional. The CSS instructions will be coded in an external file, arbitrarily named fracstyle.css.

We also need to agree on the font-size units to use. In the example, em units were used and computed by setting the body font-size to 100% and dividing by 16 all pixel units, so that 1em = 16px.

To simplify unit conversions, some Web designers like to use a body font-size of 62.5%. In this way, 1em = 10px, 1.6em = 16px, 1.8em = 18px, and so forth. Use the font-sizes and units that suits your design needs.

Once we have settled on the above, we are ready to declare all relevant CSS and HTML instructions.

Step 1 The source code for the working space is given below.

CSS
<pre>html,body{font-family:sans-serif,arial,Helvetica;background:#fff;color:#000;vertical-align:top;margin:0em;padding:0em;font-size:100%;line-height:1.125em;height:100%;} div.wrapper{margin:0em;padding:0em;border:1px solid #000;} div.row{padding:0em;margin:0em;text-align:center;clear:both;}</pre>
HTML
<pre><?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"> <head> <title>Fractal CSS Design Example</title> <meta name="keywords" content="fractal,css,design,motif,iterations" /> <meta name="description" content="Example of a Fractal CSS Layout Design" /> <link rel="stylesheet" type="text/css" href="fracstyle.css" /> </head> <body> <div class="row"><div class="wrapper"> Row 1 Content </div></div> <div class="row"><div class="wrapper"> Row 2 Content </div></div> <div class="row"><div class="wrapper"> Row 3 Content </div></div> </body> </html></pre>

Note that we have defined a div wrapper to be embedded in subsequent div elements. Feel free to modify margins and paddings to satisfy your design requirements.

For the purpose of visualizing the emerging layout so we understand what we are doing, we declare a 1px (or 0.0625em) border for the wrapper using border:1px solid #000. After completing our layout, we can go back and remove this border in order to hide from view the underlying motif. If you wish to keep it, please do.

Later on, the three div rows can be made invisible or used as head, middle, and foot divisors. BTW in our example, all content goes inside the wrappers.

Step 2: We now declare the source code for the motif.

CSS
<pre>div.wrapper{margin:0em;padding:0em;border:1px solid #000;} div.row{padding:0em;margin:0em;text-align:center;clear:both;} div.container-left,div.container-right{margin:0em;padding:0em;} div.container-left{background:url(".gif") top left repeat-y;} div.container-right{background:url(".gif") top right repeat-y;} div.container-center{background:url(".gif") top center repeat-y;} div.left{float:left;text-align:left;margin:0em;padding:0em;height:100%;width:25%;} div.right{float:right;text-align:right;margin:0em;padding:0em;height:100%;width:25%;} div.center{text-align:center;margin:0em auto;padding:0em;height:100%;width:50%;}</pre>
HTML
<pre><div class="container-left"> <div class="container-right"> <div class="left"><div class="wrapper"> Left Content Content Content Content Content </div></div> <div class="right"><div class="wrapper"> Right Content Content Content Content Content </div></div> <div class="container-center"> <div class="center"><div class="wrapper"> Center Content Content Content Content Content Content Content Content Content Content Content Content </div></div> </div> </div> </div></pre>

In our example, we use a 3-column motif, with an arbitrary width ratio of 25%-50%-25%. We could have combined some of the rules for the left, right, and center div classes. This was not done, just in case you want to modify the width ratios or individual column attributes according to your design needs.

CSS rules for background images are also included, just in case you want to use some. These CSS and HTML instructions are now added to the previous instructions.

Step 3 Next, the code for placing the motif between the rows is declared.

CSS
<pre>html,body{font-family:sans-serif,arial,Helvetica;background:#fff;color:#000;vertical-align:top;margin:0em;padding:0em;font-size:100%;line-height:1.125em;height:100%;} div.wrapper{margin:0em;padding:0em;border:1px solid #000;} div.row{padding:0em;margin:0em;text-align:center;clear:both;} div.container-left,div.container-right{margin:0em;padding:0em;} div.container-left{background:url(".gif") top left repeat-y;} div.container-right{background:url(".gif") top right repeat-y;} div.container-center{background:url(".gif") top center repeat-y;} div.left{float:left;text-align:left;margin:0em;padding:0em;height:100%;width:25%;} div.right{float:right;text-align:right;margin:0em;padding:0em;height:100%;width:25%;} div.center{text-align:center;margin:0em auto;padding:0em;height:100%;width:50%;}</pre>
HTML
<pre><?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"> <head> <title>Fractal CSS Design Example</title></pre>

```

<meta name="keywords" content="fractal,css,design,motif,iterations" />
<meta name="description" content="Example of a Fractal CSS Layout Design" />
<link rel="stylesheet" type="text/css" href="fracstyle.css" />
</head>
<body>
<div class="row"><div class="wrapper">
Row 1 Content
</div></div>

<!-- First Motif Instance -->
<div class="container-left">
<div class="container-right">
<div class="left"><div class="wrapper">
Left Content Content
Content Content Content
</div></div>
<div class="right"><div class="wrapper">
Right Content Content
Content Content Content
</div></div>
<div class="container-center">
<div class="center"><div class="wrapper">
Center Content Content Content Content
Content Content Content Content Content Content Content Content
</div></div>
</div>
</div>
</div>

<div class="row"><div class="wrapper">
Row 2 Content
</div></div>

<!-- Second Motif Instance -->
<div class="container-left">
<div class="container-right">
<div class="left"><div class="wrapper">
Left Content Content
Content Content Content
</div></div>
<div class="right"><div class="wrapper">
Right Content Content
Content Content Content
</div></div>
<div class="container-center">
<div class="center"><div class="wrapper">
</div></div>
</div>
</div>
</div>

<div class="row"><div class="wrapper">
Row 3 Content
</div></div>
</body>
</html>

```

A closer look at Figure 3 indicates that the portion of the layout between Rows 1 and 2 is the well-known three-column liquid layout used by many commercial sites across the Web. We just added an additional motif instance between Rows 2 and 3.

The second instance of the motif is now iterated. To do this, we paste the HTML markup of the motif inside the middle cell of its second instance. This time we don't need to add any extra CSS instructions as these will be reused.

Step 4: Finally, we complete the source code by adding the code for the iteration.

CSS
<pre>html,body{font-family:sans-serif,arial,Helvetica;background:#fff;color:#000;vertical-align:top;margin:0em;padding:0em;font-size:100%;line-height:1.125em;height:100%;} div.wrapper{margin:0em;padding:0em;border:1px solid #000;} div.row{padding:0em;margin:0em;text-align:center;clear:both;} div.container-left,div.container-right{margin:0em;padding:0em;} div.container-left{background:url(".gif") top left repeat-y;} div.container-right{background:url(".gif") top right repeat-y;} div.container-center{background:url(".gif") top center repeat-y;} div.left{float:left;text-align:left;margin:0em;padding:0em;height:100%;width:25%;} div.right{float:right;text-align:right;margin:0em;padding:0em;height:100%;width:25%;} div.center{text-align:center;margin:0em auto;padding:0em;height:100%;width:50%;}</pre>
HTML
<pre><?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"> <head> <title>Fractal CSS Design Example</title> <meta name="keywords" content="fractal,css,design,motif,iterations" /> <meta name="description" content="Example of a Fractal CSS Layout Design" /> <link rel="stylesheet" type="text/css" href="fracstyle.css" /></head> <body> <div class="row"><div class="wrapper"> Row 1 Content </div></div> <!-- First Motif Instance --> <div class="container-left"> <div class="container-right"> <div class="left"><div class="wrapper"> Left Content Content Content Content Content </div></div> <div class="right"><div class="wrapper"> Right Content Content Content Content Content </div></div> <div class="container-center"> <div class="center"><div class="wrapper"> Center Content Content Content Content Content Content Content Content Content Content Content Content </div></div> </div> </div> </div> <div class="row"><div class="wrapper"> Row 2 Content </div></div> <!-- Second Motif Instance --> <div class="container-left"> <div class="container-right"> <div class="left"><div class="wrapper"> Left Content Content Content Content Content </div></div> <div class="right"><div class="wrapper"> Right Content Content Content Content Content </div></div> <div class="container-center"> <div class="center"><div class="wrapper"> <!-- Iteration --> <div class="container-left"> <div class="container-right"></pre>

```
<div class="left"><div class="wrapper">
Left Content Content
Content Content Content
</div></div>
<div class="right"><div class="wrapper">
Right Content Content
Content Content Content
</div></div>
<div class="container-center">
<div class="center"><div class="wrapper">
Center Content Content Content Content
Content Content Content Content Content Content Content Content
</div></div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<div class="row"><div class="wrapper">
Row 3 Content
</div></div>
</body>
</html>
```

We are done! If you plan to use this example as a *Faux Columns* layout, you need to supply your own background images.

Possible Modifications

In this basic example, we ended up with a table-like design that readjusts its size when the browser's window is resized.

If we want, we can:

- hide the rows by modifying their CSS rules.
- place a banner or breadcrumb menu inside Rows 1 and 3
- use Row 1 for a header and Row 3 for a footer
- place main content, graphics, or a search box in the middle cells.

For browsers that cannot interpret well the `<?xml version="1.0" encoding="UTF-8"?>` tag, charset information can be specified with `<meta name="content-type" http-equiv="content-type" content="text/html; charset=utf-8" />`.

Since the CSS rules equally affect different portions of the layout, one might want to incorporate some style diversity. To preferentially affect particular instances of a motif or particular regions of the design, one could rename classes and declare new CSS rules. One way of doing this consists in making new instances of the original CSS code itself and then using a naming convention for each instance.

We can also:

- rename classes or add new ones
- modify rules so changes will affect specific portions of the layout.
- place pre-styled elements like paragraphs, divisions, etc, to completely fill individual cells.

Again, hiding the wrappers' borders makes the underlying motif less obvious.

Additional Work

Although we could, we did not combine the iterative process with other transformations like translation and reflection.

Overall, the CSS code used was pretty rudimentary. No attempts were made at optimizing the CSS rules, making columns all of same height, or declaring rules for elements inside cells since this article is aimed at showing how to implement recursive layouts with a small set of CSS rules. We have built a tool, The Fractal CSS Design Studio, to automate the generation of optimized rules, motifs, and layouts.

Figure 5 depicts some motifs and the resultant layout patterns obtained after few iterations.

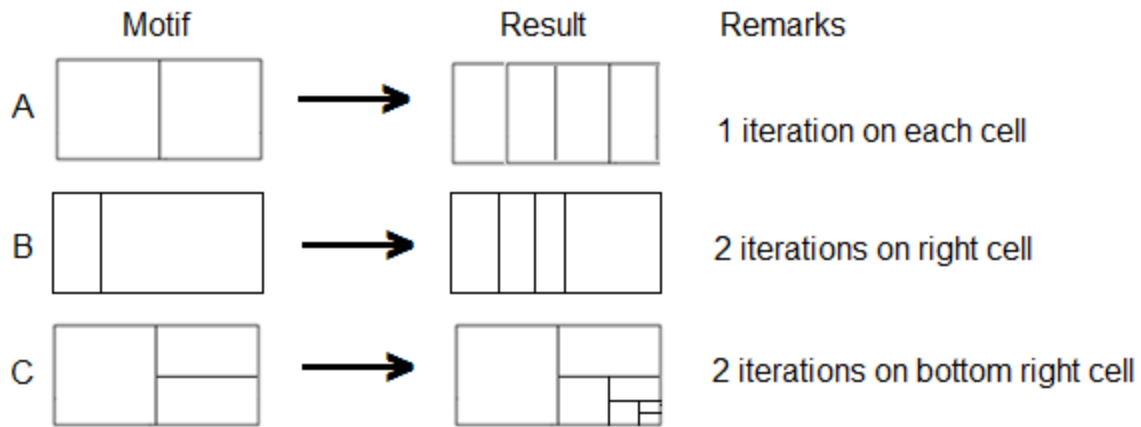


Figure 5. Some motifs and their outcome.

Motifs A and B can be designed with the previous CSS rules by simply commenting or removing the `div.center` and `div.container-center` classes and changing width values. For instance for motif A, we can use a width of 50% for the `div.left` and `div.right` classes and for motif B we can use 25% for the `div.left` class and 75% for the `div.right` class. Actual screenshots are given below.

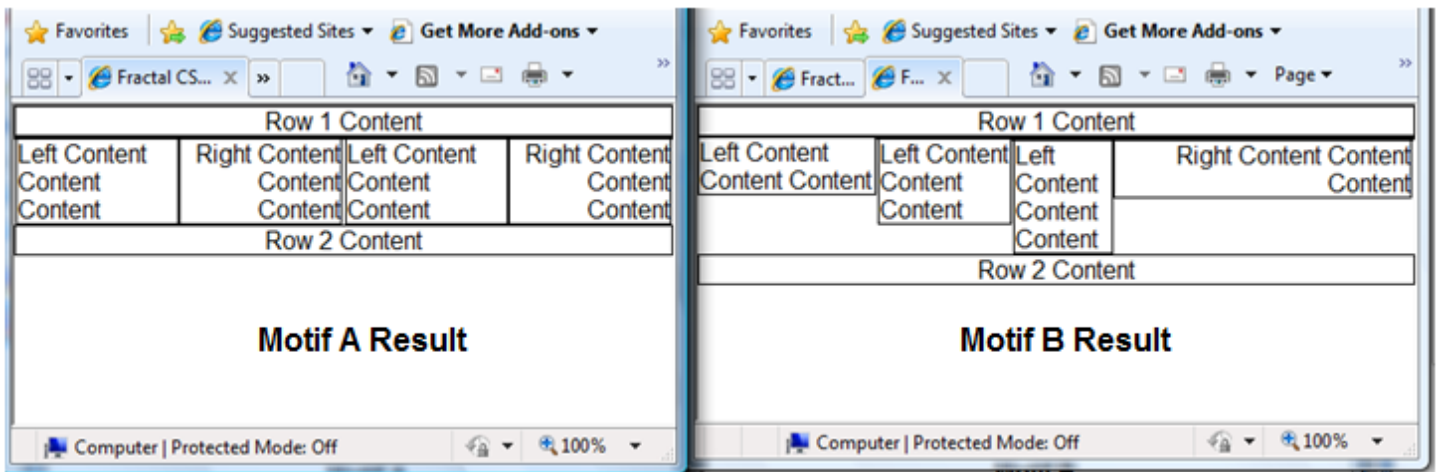


Figure 6. Screenshot for motifs A and B..

We have resized the browser windows to demonstrate that column lengths resize just fine. The small gap in some cells is due to the fact that we are adding a border to the wrappers and that some cells do not contain the same number of wrappers within wrappers. The gap is not so obvious in Figure 4.

To fill these gaps, one might be tempted to add empty wrappers or use the overflow:hidden property, but this might bring back other design issues to work around. We prefer to hide or remove completely the borders from the final layout.

Motifs A and B lead to multi-cell layouts similar to those found across the Web, but that unnecessarily have been designed in a cumbersome fashion or with clunky HTML tables. Motif C leads to a bit more complex layout. Adding other transformations like rotation leads to even more complex patterns.

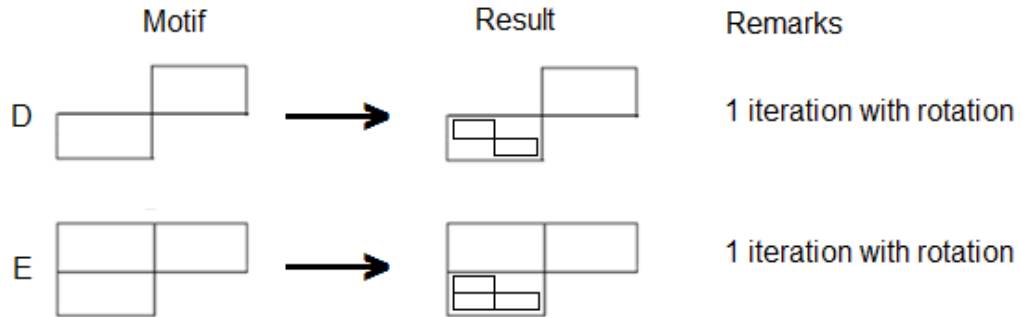


Figure 7. Additional motifs and their outcome, with rotation operations..

The “missing” cells in some of the motifs imply that no textual content is assigned to these or, for instance, that these are reserved for graphics. This is just a convention.

Last, but not least, by simply renaming all div.left and div.right classes for the three motif examples herein discussed and adding these to the CSS source, a highly “liquid” multifractal collage layout is obtained with a small set of 14 CSS rules. This is shown in Figure 8.

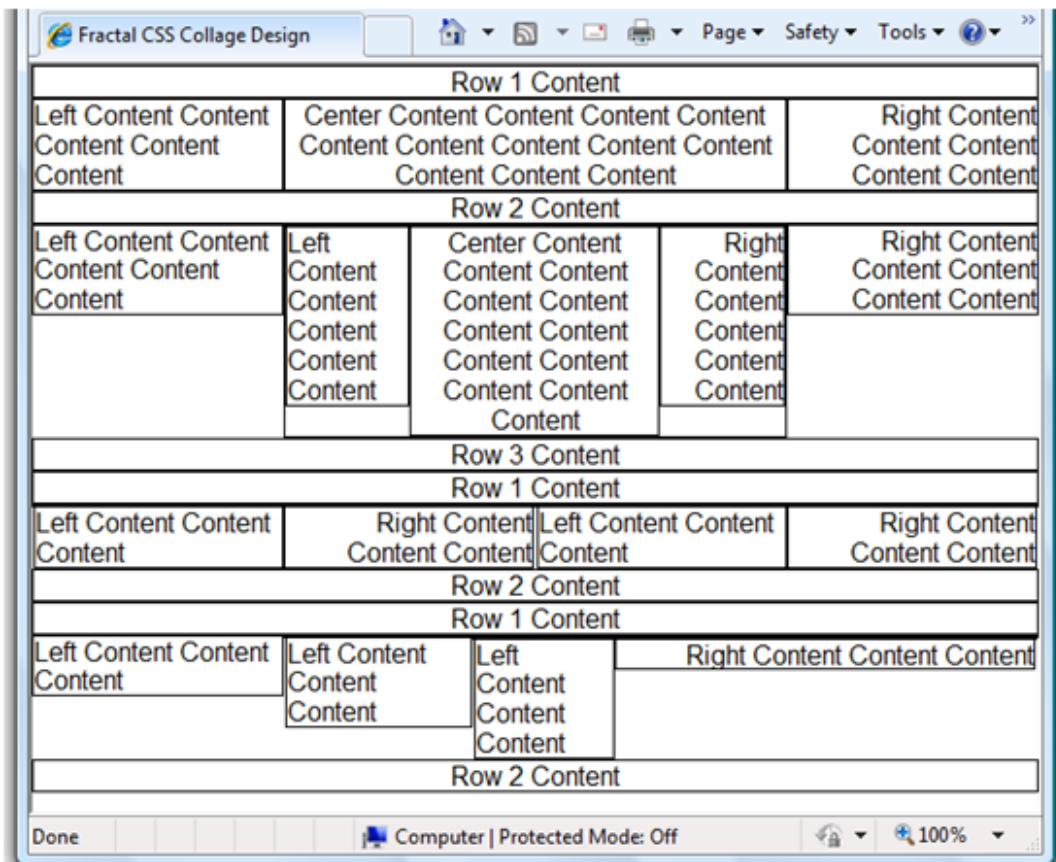


Figure 8. Multifractal Layout Collage.

Conclusion

While the concept of using CSS rules for emulating HTML table layouts is not new, the main difference between our approach with other solutions is that ours is purely algorithmic and based on fractal theory; i.e., it is result of defining a small set of CSS rules for individual motifs, iterating these, and using the resultant fractal layouts as building blocks for larger layouts, which frequently turn out to be multifractals. With the exception of the renamed divs, the CSS code is used over and over; so, this is a minimalistic approach.

All the layouts presented have been tested with the Internet Explorer and Firefox browsers. These have been created with The Fractal CSS Design Studio and, to verify the results, manually. We plan to incorporate some of these layouts into the Mi Islita.com site in the near future, if we can find the time.

Feel free to test our approach with your very own patterns and send us some feedback. You might also want to mix patterns for a richer, table-like layout experience. At this point, we can safely say that HTML tables are no longer required for rendering highly complex, tabular Web pages.

Now for the IR-inclined, here are some questions, great for a graduate thesis:

- How would fractal CSS layouts affect keyword and content relevancy across search engines?
- Comparing two visually similar layouts, one based on HTML tables and the other on fractal design, how is the distribution of keywords (words spacing, localized frequency, contextuality, etc) affected?
- How CSS reusable patterns impact the lexicographical trees present in a document?

These and similar questions are addressed in Reference 5. The impact of iterated layouts on document parsing is discussed in a recent article (9).

References

1. The Fractal Nature of Semantics
<http://www.miislita.com/fractals/fractal.html>
2. Fractal Motifs and Iterated Function Systems
<http://www.miislita.com/fractals/motifs-iterated-function-systems.html>
3. Overlapping Patterns: EF-Ratios, Separators, Patterns and Pitfalls
<http://www.miislita.com/fractals/overlapping-patterns.html>
4. Grammar, Semantics, Knowledge and Fractals
<http://www.miislita.com/fractals/grammar-semantics-fractals.html>
5. The Keyword Density of Non-Sense
<http://www.miislita.com/fractals/keyword-density-optimization.html>
6. Fractal Patterns, L-Systems and Semantics
<http://www.miislita.com/fractals/fractals-l-systems-semantics.html>
7. Fractal Clusters - Fractal Networks
<http://www.miislita.com/fractals/fractal-clusters-fractal-networks.html>
8. Fractals in Information Retrieval
<http://www.miislita.com/fractals/fractals-information-retrieval.html>
9. Fractal CSS Design as a Row Primary Technique
<http://www.miislita.com/fractals/fractal-css-design-row-primary-technique.pdf>