Representing Component Authorship Using Randomly Generated Glyphs

– work in progress –

Alexandre Bergel
Pleiad Lab, DCC, University of Chile

Abstract

VisualIDs are randomly generated glyphs. A glyph visually describes an object and is designed to let a human easily recognize similarity between the represented objects.

In this short abstract, we uses glyphs to represent code authorship. We have employed VisualID glyphs to represents author collaboration across over 160 classes.

1. VisualID Glyph

VisualID [1] is a technique to randomly generate visual icons, called glyphs. A glyph represents an object and uses cognitive abilities to identify similarity between the represented objects. The visual aspect of a glyph is randomly generated and indicates the similarity an object has with other objects: two similar objects are represented with two visually similar glyphs.

Technically, glyphs are generated from a *comparison function* and a *threshold*. The comparison function indicates how similar two objects are by producing a numerical value between 0.0 and 1.0. The threshold indicates whether the two objects are similar or not, *i.e.,* whether their comparison is equal or greater to the threshold. In the remaining of this short abstract we will illustrate the glyph production using classes as the seed objects.

To produce a glyph for a class C, the VisualID algorithm checks whether a previous glyph has been generated for another class D similar to C. If the class D has already a glyph G_D, then G_D is mutated to produce G_C, which will be used to represent C. If no glyph has been previously produced, then a random glyph is generated for C and is kept for future comparison.

The VisualID is a relatively simple algorithm. The glyph generation is based on a grammar made of 8 production rules: figure, line, path, shape, null, radial, spiral, and symmetry. These production rules are recursive. Recursion ends when a complexity or depth is reached. A glyph is mutated by slightly modifying some parameters associated to each production.

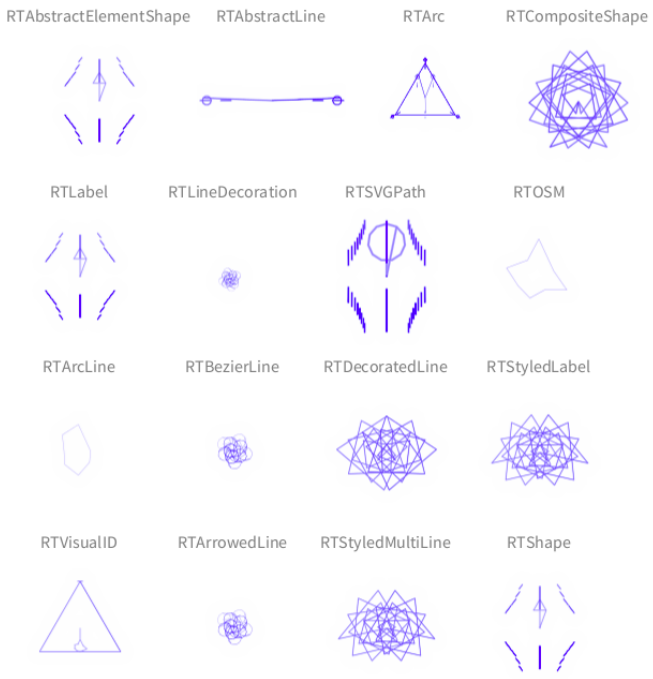

Figure 1: VisualID glyphs example

In our previous work [2], we have employed glyphs to address two software engineer tasks: (i) identify classes with the same dependencies and (ii) identify classes having a similar set of methods. In this short paper, we use glyphs to represent class authorship (Section 2) and presents some possible future explorations (Section 3).

SATTOSE '16, Month d–d, 20yy, City, ST, Country.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnnReprinted from SATTOSE '16, [Unknown Proceedings], Month d–d, 20yy, City, ST, Country, pp. 1–2.

2. Expressing Authorship

Figure 1 represents 16 classes taken from a large application. Similarity between two classes indicates that these two classes have similar authors. The comparison function we employed in this visualization takes two classes and compare the list of authors for each class. We use the Jaccard operator to compare these lists of authors.

The figure shows that the classes `RTStyledLabel` and `RTStyledMultiLine` have the same authors since the glyphs are identical. The class `RTDecoratedLine` share some authors with these two classes.

Similarly, the couple `RTAbstractElementShape` / `RTLabel` / `RTShape` and `RTBezierLine` / `RTArrowedLine` have the same set of authors.

As a larger illustration, consider Figure 2. It shows the 165 classes composing the Pharo collection library. 189 programmers have contributed to these classes.

Currently, we consider all the authors having the same degree of authorship of the class, implying that no weight is being used for the moment.

3. Future Work

Glyphs have a great potential to enhance the programming environment to convey contextual information enabling comparison. However, randomly generated glyphs are unfortunately rarely considered to address some software engineering tasks. VisualID glyphs have been employed in a number of diverse situations, and have been successfully employed.

As future work, we plan to work on the following points:

- improve the notion of authorship to reflect the degree of participation of each author.
- improve the visual rendering of glyph. Currently, a glyph is unicolor and painting sub-glyph in a different color could improve the cognitive ability of the glyphs.

References

[1] J. Lewis, R. Rosenholtz, N. Fong, U. Neumann, VisualIDs: automatic distinctive icons for desktop interfaces, ACM Transactions on Graphics 23 (3) (2004) 416–423.

[2] I. Fernandez, A. Bergel, J. P. S. Alcocer, A. Infante, T. Gîrba, Glyph-based software component identification, in: Proceedings of the 24th IEEE International Conference on Program Comprehension (ICPC '16), 2016.

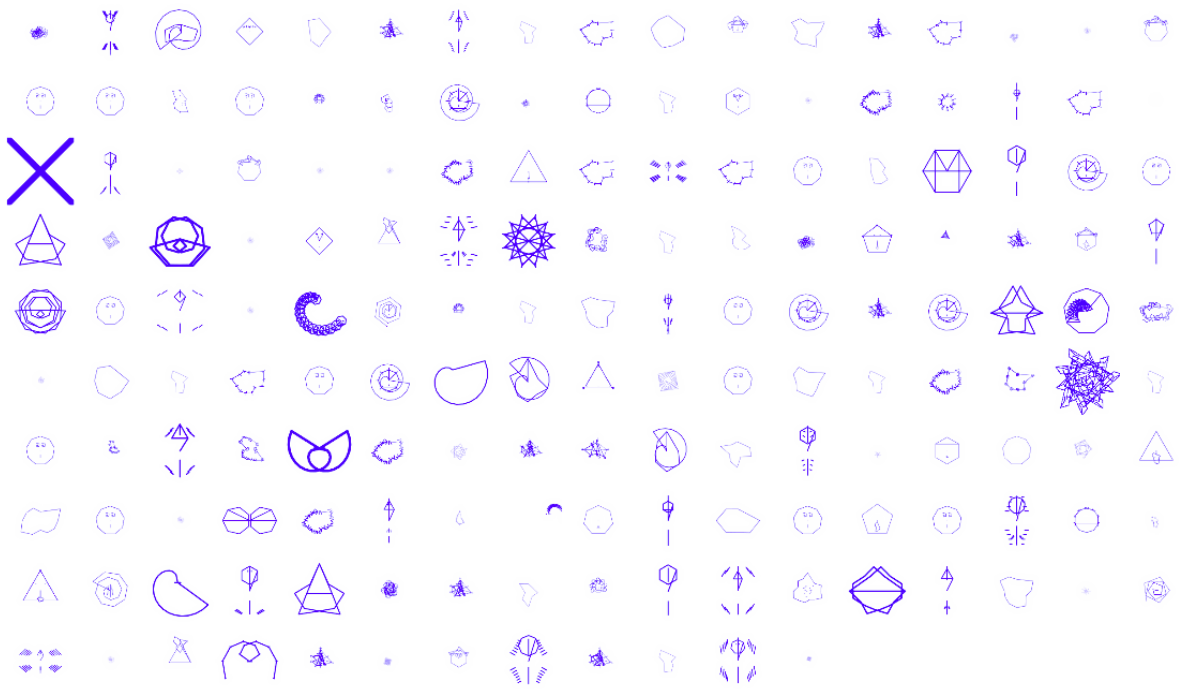

Figure 2: VisualID glyphs example