Implementation of the Shortcut Tree Optimization

Terts Diepraam

December 2022

For the implementation accompanying the presentation, I decided to implement the generation of the shortcut tree from [3] and visualize the result. The input is an SVG document containing a single path and the output is the same SVG, but with a representation of the shortcut tree, initial winding numbers and abstract segments. This SVG can be loaded by a web page to toggle the different layers of information.

The procedure is implemented in Rust, using the `svg` [1] and `kurbo` [2] libraries. First, the path in the SVG is split into abstract segments, which are then organized into the shortcut tree. This is not done in parallel, since the goal of this implementation is visualization and not performance. The maximum depth of the tree is 4, since visualization becomes difficult at a higher depth.

The web page is naturally powered by JavaScript. Most elements in the output SVG can be hidden in the web page for clarity. Additionally, when the mouse hovers over the picture, a ray and corresponding winding number is shown with intersection with the segments. If the shortcut tree is enabled, only the segments in the hovered cell are considered, as shown in figure **??**. Shortcut segments and initial winding numbers are also taken into account when enabled. The `kld-intersections` [4] library was used to compute the intersections between the segments and the ray.

The source code and a live demo are respectively available at the following links:

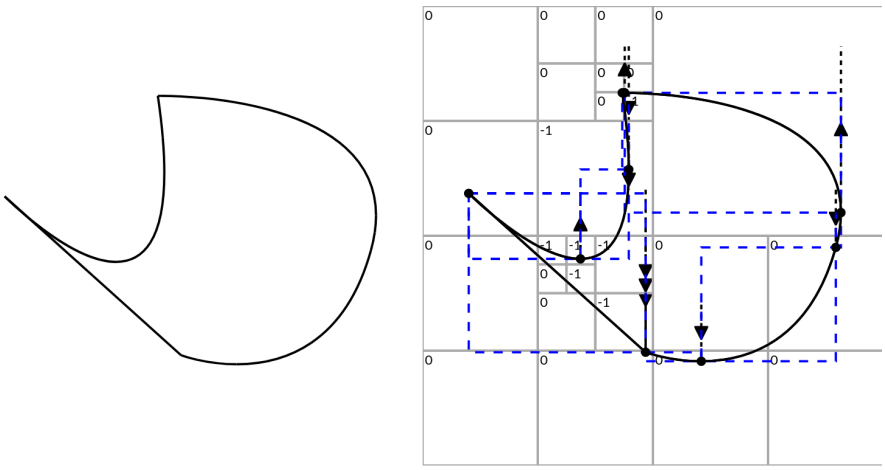
- Source code: `https://github.com/tertsdiepraam/shortcut-tree/`
- Live demo: `https://tertsdiepraam.github.io/shortcut-tree/`

References

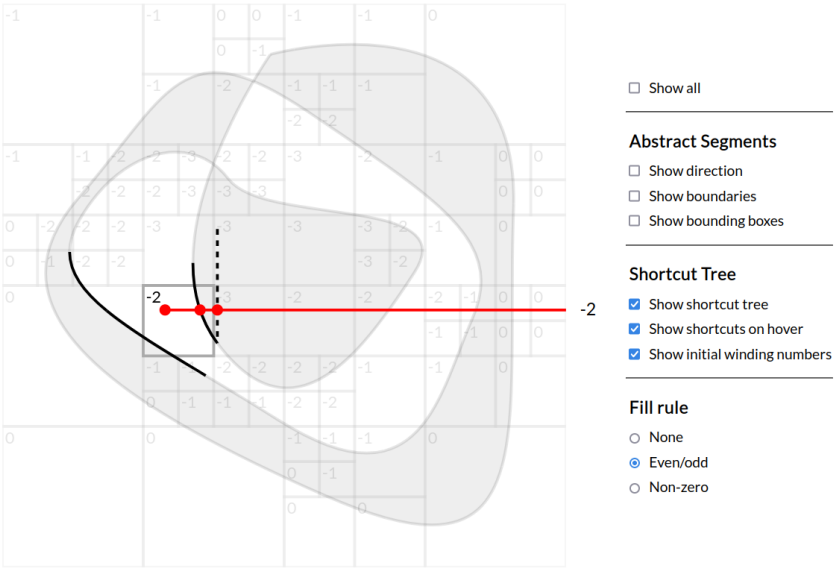
[1] Ivan Ukhov et al. *svg*. URL: `https://github.com/bodoni/svg`.

[2] Raph Levien et al. *kurbo*. URL: `https://github.com/linebender/kurbo`.

[3] Francisco Ganacim et al. "Massively-parallel vector graphics". In: *ACM Transactions on Graphics (TOG)* 33 (2014), pp. 1–14.



(a) Input SVG (b) Output SVG



(c) An output SVG loaded into the webpage, showing a ray from the mouse pointer intersecting with the segments in a cell and an abstract segment, while an initial winding number is applied as well.

[4] Kevin Lindsey, Brett Zamir, and Robert Benko. *kld-intersections*. URL: `https://github.com/thelonious/kld-intersections`.