

# RAMA: An Interactive Artist Network Visualization Tool

**Diogo Costa**  
INESC Porto  
dcosta@inescporto.pt

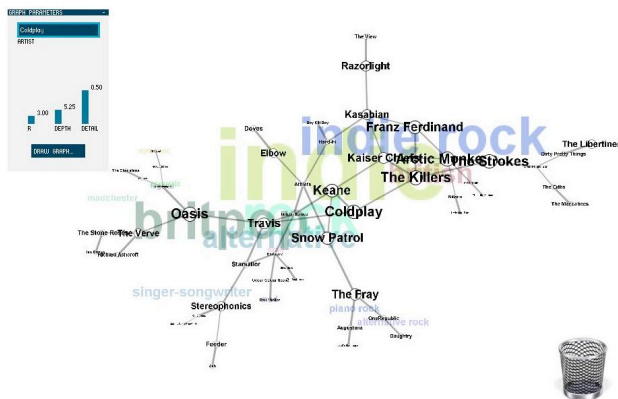
**Luis Sarmiento**  
LIACC/FEUP  
las@fe.up.pt

**Fabien Gouyon**  
INESC Porto  
fgouyon@inescporto.pt

## ABSTRACT

We present RAMA (Relational Artist Maps), a web-based tool for visualizing and interacting with networks of music artists. RAMA operates on data that covers hundreds of thousands of artists and three million user-generated tags, collected from Last.fm web radio. Artist names are presented in a connected graph, which also contains visual information concerning (i) artist similarities, (ii) artist popularity and (iii) associated tags. Differing from existing artist network visualization tools, RAMA emphasizes commonalities as well as main differences between artist categorizations at the level of user-defined tags. A number of interactive features also provide an enhanced user browsing experience, and aims to help users expand their knowledge about the world of music. RAMA is available at <http://rama.inescporto.pt>

## 1. RAMA



RAMA is a web application that allows visualizing data crawled from Last.fm's API onto a local database. Technically, this consists of a backend REST web service mapping and publishing the data, and a JAVA applet frontend responsible for data visualization and manipulation. RAMA front end was developed using Processing.

### 1.1 Graph representation

Artists' networks are represented as connected graphs: edge length is an indicator of the similarity between two artists while node sizes represent artist popularity. For reflecting the folksonomy associated with a specific graph, the user-generated tags related with the artists are presented on top of the graph. Tag positions provide visual hints to possible clusters of artists, while tag size is a function of the number of artists with that tag.

### 1.2 Graph construction

Starting from a specific user query (e.g. artist A), the application will construct the graph by recursively finding the most similar artists to every other artist. A number of user parameters are available to define the topology of the graph: (i) the *initial ramification* factor, (ii) the *depth* (how far should we go from the seed), and (iii) the *population* factor that controls how the rami-

fication decays with depth. Optionally the user can create the graph *manually* expanding each artist's neighbors, or deleting some nodes. Increasing the initial ramification and decreasing the depth will promote artists directly connected to the initial artist and the reverse will widen the search providing a more panoramic view, possibly covering different music genres.

### 1.3 Graph drawing

The strategy we use for graph-drawing was originally inspired by the *spring physic model*: artists have "charges" proportional to their popularity, and springs connecting artists have a natural rest distance inversely proportional to the similarity between those artists. The drawing procedure is run on the front-end iteratively, providing the user with real time animation.

## 2. INTERACTIONS

### 2.1 Requesting new graphs

The user can request a new graph at anytime. Once the corresponding data is available on the client side, the old graph (if one had already been requested before) is progressively transformed into the new one: nodes that are common to both graphs will not change.

### 2.2 Manually expanding nodes

The user may expand information associated with any artist in the graph by clicking on the corresponding artists: new nodes will be added to the graph. This may be used to add artists to an already generated graph, or to construct a graph starting with just one artist.

### 2.3 Zoom/Pan/Framing and Graph manipulation

When the graph is first presented, the application will rescale and center it, as it is drawn. This is important to give the user a correct overview of the graph. The user may then pan and zoom freely, as well as draw an area of interest to be zoomed. The user can at any time drag, pin and delete nodes in the graph, or delete parts of graphs dragging them to the recycle bin.

### 2.4 Tag highlighting

Tag highlighting is an important part in visual exploration and filtering. By hovering over tags the user will be able to see which artists share a tag. Hovering over edges will reveal the tags that are specific to each of the connected artists, i.e. those tags that are different between them.

### 2.5 "Did you mean?"

When the user types in a name not found in our database (due to e.g. a spelling mistake), the back-end tries to provide the user with a list of options containing names that are lexically close. These are computed by combining the *Soundex* algorithm with the *Levenshtein* distance metric and discarding the most frequent *stopwords* in the database. This feature is extremely useful because, as we concluded from previous versions of RAMA, users tend to frequently type incorrect or incomplete versions of artist names, without being aware of it.