RapidGraph

Web-based Visual Graph Analyzer Framework

Rob McGuire-Dale and Mike Anderson
(Details on bottom banner)

Project Description

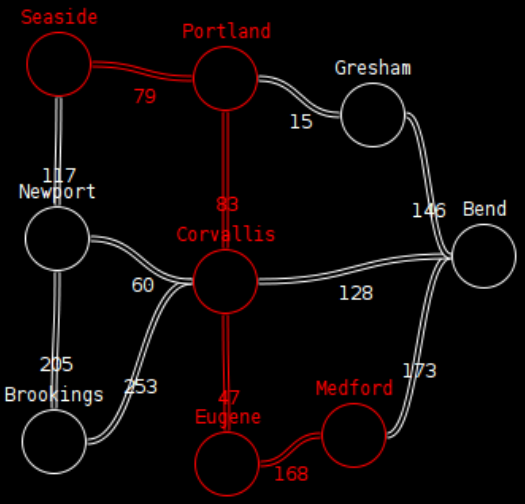
The mathematical graph analyzers that exist today require installation and/or use of unintuitive, complicated mathematics software.

RapidGraph is a web-based graph analyzer framework that will run in any modern web browser.

It is easy to rapidly create graphs with an intuitive user interface, and efficiently analyze them with an plugin-based, easily-extendable API.

Applications

Bundled example plugins demonstrate real-world applications and power of the RapidGraph framework, including flight scheduling, the "traveling salesman" problem, and shortest path.



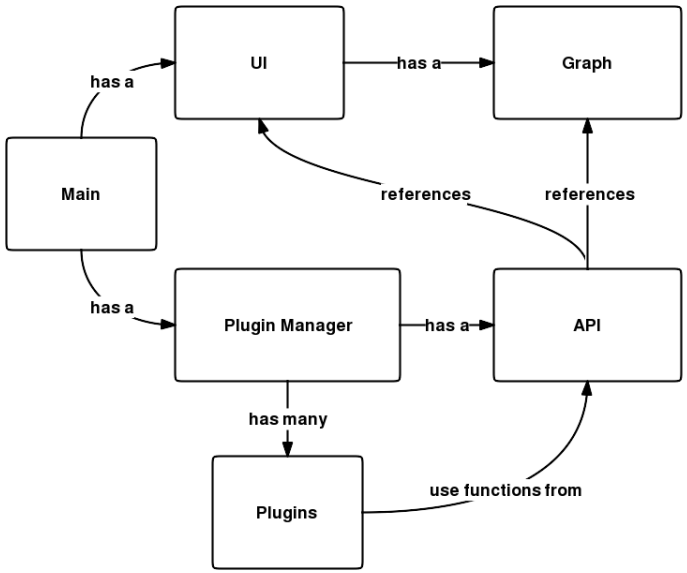
Features

Accessible – Its web-based design allows it to run in any modern browser without the need for users to install any additional software.

Createable - An intuitive and easy-to-use interface design encourages fast and accurate graph creation.

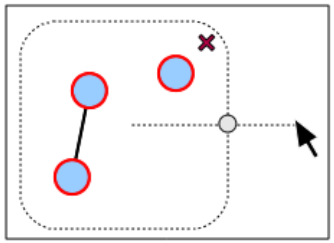
Portable - RapidGraph is built with standard, open languages ensuring maximum portability both server-side and client-side.

Modular – A robust plugin system allows for fast and consistent functional extendibility.

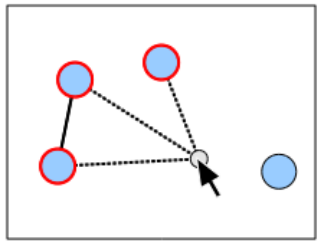


Creating Edges:

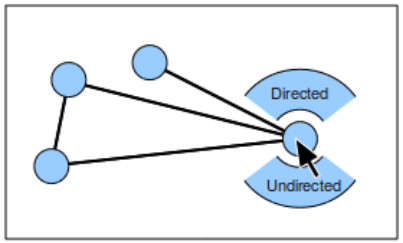
A "handle" orbits around selected



Prospective edges created



Context menu appears when created



Technical Details

RapidGraph is mostly written in Javascript with the jQuery and Raphaël libraries. The UI uses standard CSS, and HTML, and the <canvas> tag.

The server Django, which handles server-side tasks and the plugin system. Python is used with plugins requiring high algorithmic performance.

Git was used for code versioning, and Trac was used for project management. You can find out more at http://trac.osuosl.org/rapidgraph

Conclusions

Successes:

Javascript is an excellent client-side language

The jQuery and Raphaël libraries help with tremendously with graphics and user interfaces

Django and Python can be used successfully to procedurally generate client-side code

Challenges:

Steep learning curve with acquiring multiple new languages and libraries

Synthesizing multiple open-source technologies to work cooperatively together

Work was often under-appreciated due to non-topographical accomplishments

Credits: Rob McGuire-Dale, Project manager (mcguirer@engr.oregonstate.edu) | Michael Anderson, Developer (andermic@engr.oregonstate.edu) | Christine Wallace, Sponsor (wallach@eecs.oregonstate.edu)