Map::Tube – Lightweight Routing Framework

About me:

- Perl & I, are in relationship for nearly ***20*** years.
- Published about ***78*** CPAN modules, pause id "***MANWAR***".
- Maintains some of the most popular distributions e.g. **PDF::Create**, **XML::XPath**, **SVG** etc.
- Contributed to ***293*** distributions as of today e.g. **Dancer2**, **Dist::Zilla**, **Test::More** etc.
- Over ***1000*** consecutive days of releasing to CPAN.

London Perl Workshop 2017

(www.manwar.org)

Background of Map::Tube

- Lightweight ***Moo-based*** role.
- Actively maintained for the last ***8 YEARS***. There have been ***152*** releases so far, last being Map::Tube ***V3.40***.
- Supports the following plugins.
  - **Map::Tube::Plugin::Graph**
  - **Map::Tube::Plugin::FuzzySearch**
  - **Map::Tube::Plugin::Formatter**
- Supports command line tool '***map-tube***' supplied by **Map::Tube::CLI**
- Provides command line tool '***map-data-converter***', that can help you change the map data format between ***JSON*** and ***XML***.
- Contributors
  - Michal Špaček (***SKIM***)
  - Gisbert W. Selke (***GWS***)
  - Slaven Rezic (***SREZIC***)

Maps Available

Barcelona	Beijing	Berlin	Bucharest	Budapest	Delhi
Dnipropetrovsk	Glasgow	Kazan	Kharkiv	Kiev	Koeln Bonn
Kolkatta	Kuala Lumpur	***London***	Lyon	Malaga	Minsk
Moscow	New York	Nanjing	Nizhny Novgorod	Novosibirsk	Prague
Saint Petersburg	Samara	Singapore	Sofia	Tbilisi	Tokyo
Vienna	Warsaw	Yekaterinburg			

Main Features

- Find the shortest route between two stations.
- Plot nice map using the plugin **Map::Tube::Plugin::Graph**
- Allow fuzzy search of station name using the plugin **Map::Tube::Plugin::FuzzySearch**
- Get the search result in many formats using the plugin **Map::Tube::Plugin::Formatter**

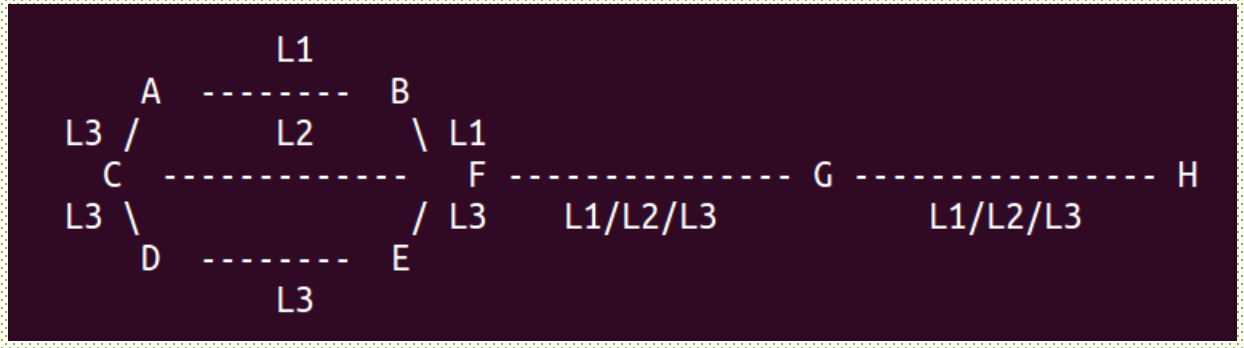
Lets build a new map

- Step 1: Collect the source data of the new map.
- Step 2: Decide the format of map data. e.g. ***JSON*** or ***XML***.
- Step 3: Build map data in the selected format.
- Step 4: Create package to consumes the role **Map::Tube**.

We will prepare the environment first

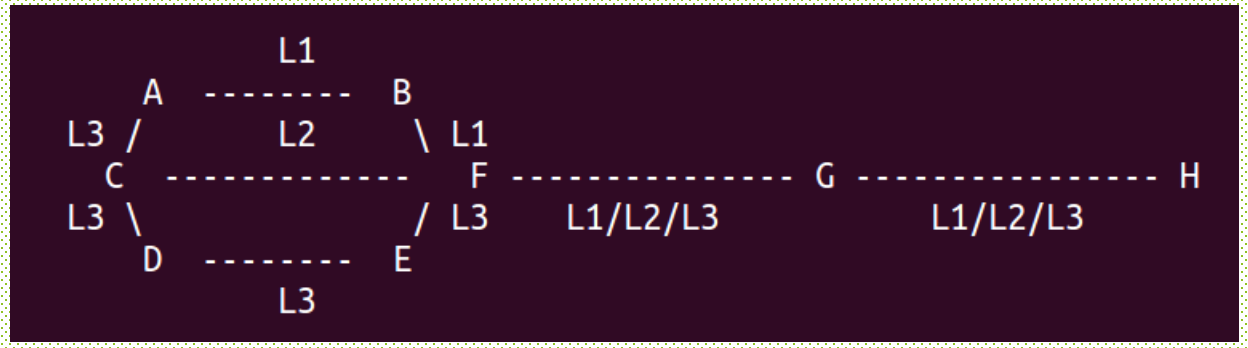
```
$ sudo cpanm -v Map::Tube
$ sudo cpanm -v Map::Tube::Plugin::Graph
```

Step 1: Collect the map data.



- For this short talk, let us take simple map like above, named “Trial”.
- In the above map, we have stations named as A,B,C,D,E,F,G and H.
- The lines are named as L1,L2 and L3.

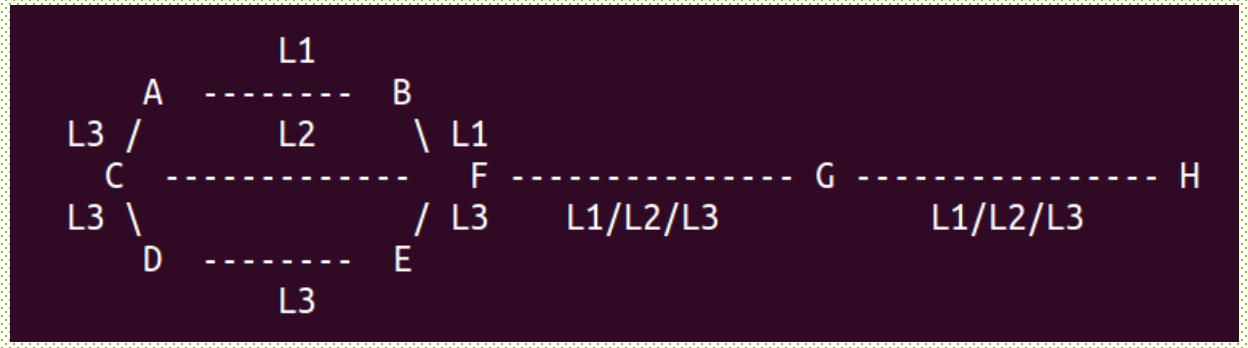
Step 2: Decide the format of map data.



- Let us assume we decided on JSON format.
- Let us build the skeleton of map data as below:

```
{
    "name": "Trial",
    "lines":
    {
         "line":
         [
         ]
    },
    "stations":
    {
        "station":
        [
        ]
    }
}
```

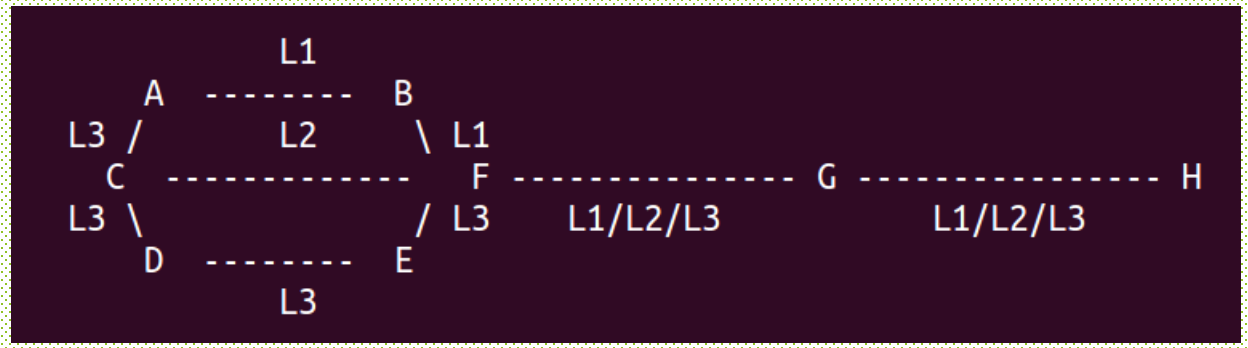
Step 3: Build map data in selected format



- Let us add the line information first.

```
{
    "name": "Trial",
    "lines":
    {
        "line":
        [
            { "id" : "L1", "name" : "L1" },
            { "id" : "L2", "name" : "L2" },
            { "id" : "L3", "name" : "L3" }
        ]
    },
    "stations":
    {
        "station":
        [
        ]
    }
}
```

...continued (Step 3)



- Finally we will now add the station details.

```
{
    "name": "Trial",
    "lines":
    {
        "line":
        [
            { "id" : "L1", "name" : "L1" },
            { "id" : "L2", "name" : "L2" },
            { "id" : "L3", "name" : "L3" }
        ]
    },
    "stations":
    {
        "station":
        [
            { "id": "A", "name": "A", "line": "L1,L3",     "link": "B,C"      },
            { "id": "B", "name": "B", "line": "L1",       "link": "A,F"      },
            { "id": "C", "name": "C", "line": "L2,L3",    "link": "A,D"      },
            { "id": "D", "name": "D", "line": "L3",       "link": "C,E"      },
            { "id": "E", "name": "E", "line": "L3",       "link": "D,F"      },
            { "id": "F", "name": "F", "line": "L1,L2,L3", "link": "B,C,E,G"  },
            { "id": "G", "name": "G", "line": "L1,L2,L3", "link": "F,H"      },
            { "id": "H", "name": "H", "line": "L1,L2,L3", "link": "G"        }
        ]
    }
}
```

Step 4: Create package to consumes Map::Tube selected format

- This is the easiest step of all. The package Map::Tube::Trial has ***6 lines*** of code in total.

```
package Map::Tube::Trial;

use Moo;
use namespace::autoclean;

has json => (is => 'ro', default => sub { 'trial.json' });
with 'Map::Tube';

1;
```

Create test script

```
#!/usr/bin/perl

use strict; use warnings;
use Map::Tube::Trial;

my $map = Map::Tube::Trial->new;
print $map->get_shortest_route('A', 'D'), "\n";
```

```
my $name = $map->name;
open(my $MAP_IMAGE, ">$name.png");
binmode($MAP_IMAGE);
print $MAP_IMAGE decode_base64($map->as_image);
close($MAP_IMAGE);
```

```
my $line = 'L3';
open(my $LINE_IMAGE, ">$line.png");
binmode($LINE_IMAGE);
print $LINE_IMAGE decode_base64($map->as_image($line));
close($LINE_IMAGE);
```

Bonus Features

- Lines can be color coded as most maps do use color code.
- Stations can be indexed per line.
- Stations can be linked by “other think”.

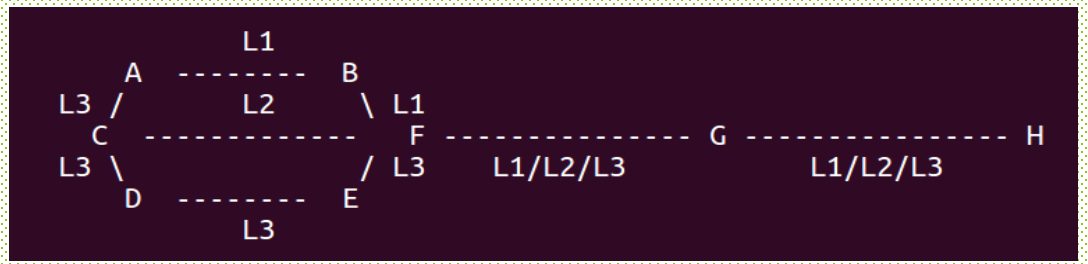
Bonus Feature #1: Color code line

- This will be handy when generating map image (graph).
- Here is the updated sample data with line color code.

```
{
    "name": "Trial",
    "lines":
    {
        "line":
        [
            { "id": "L1", "name": "L1", "color": "red"   },
            { "id": "L2", "name": "L2", "color"; "blue"  },
            { "id": "L3", "name": "L3", "color": "green" }
        ]
    },
    "stations":
    {
        "station":
        [
            { "id": "A", "name": "A", "line": "L1,L3",     "link": "B,C"       },
            { "id": "B", "name": "B", "line": "L1",       "link": "A,F"       },
            { "id": "C", "name": "C", "line": "L2,L3",    "link": "A,D"       },
            { "id": "D", "name": "D", "line": "L3",       "link": "C,E"       },
            { "id": "E", "name": "E", "line": "L3",       "link": "D,F"       },
            { "id": "F", "name": "F", "line": "L1,L2,L3", "link": "B,C,E,G"   },
            { "id": "G", "name": "G", "line": "L1,L2,L3", "link": "F,H"       },
            { "id": "H", "name": "H", "line": "L1,L2,L3", "link": "G"         }
        ]
    }
}
```

Bonus Feature #2: Index station per line

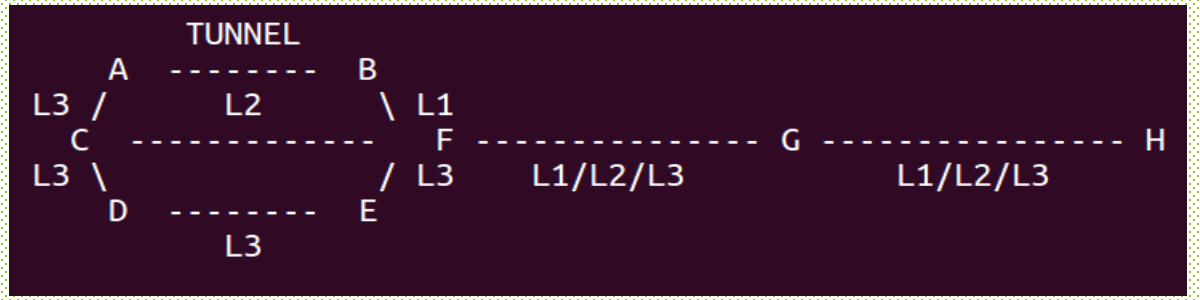
- This will be handy when fetching station lists for a particular line.
- Without index, result station list would be ordered alphabetically instead of how it appears in map.
- Here is the update sample data with station index.




```
{
    "name": "Trial",
    "lines":
    {
        "line":
        [
            { "id": "L1", "name": "L1", "color": "red"   },
            { "id": "L2", "name": "L2", "color"; "blue"  },
            { "id": "L3", "name": "L3", "color": "green" }
        ]
    },
    "stations":
    {
        "station":
        [
            { "id": "A", "name": "A", "line": "L1:1,L3:1",       "link": "B,C"       },
            { "id": "B", "name": "B", "line": "L1:2",           "link": "A,F"       },
            { "id": "C", "name": "C", "line": "L2:1,L3:2",      "link": "A,D"       },
            { "id": "D", "name": "D", "line": "L3:3",           "link": "C,E"       },
            { "id": "E", "name": "E", "line": "L3:4",           "link": "D,F"       },
            { "id": "F", "name": "F", "line": "L1:3,L2:2,L3:5", "link": "B,C,E,G"   },
            { "id": "G", "name": "G", "line": "L1:4,L2:3,L3:6", "link": "F,H"       },
            { "id": "H", "name": "H", "line": "L1:5,L2:4,L3:7", "link": "G"         }
        ]
    }
}
```

Bonus Feature #3: Link station by “other link”

- In some map, two stations are linked by “tunnel” or by some other link. For example, in London tube map, the “Bank” station is also linked to “Monument” station by “tunnel”.
- Here is how it can be represented in the map data.




```
{
    "name": "Trial",
    "lines":
    {
        "line":
        [
            { "id": "L1", "name": "L1", "color": "red"   },
            { "id": "L2", "name": "L2", "color": "blue"  },
            { "id": "L3", "name": "L3", "color": "green" }
        ]
    },
    "stations":
    {
        "station":
        [
            { "id": "A", "name": "A", "line": "L3:1",           "link": "C", "other_link": "tunnel:B" },
            { "id": "B", "name": "B", "line": "L1:1",           "link": "F", "other_link": "tunnel:A" },
            { "id": "C", "name": "C", "line": "L2:1,L3:2",      "link": "A,D"          },
            { "id": "D", "name": "D", "line": "L3:3",           "link": "C,E"          },
            { "id": "E", "name": "E", "line": "L3:4",           "link": "D,F"          },
            { "id": "F", "name": "F", "line": "L1:2,L2:2,L3:5", "link": "B,C,E,G"      },
            { "id": "G", "name": "G", "line": "L1:3,L2:3,L3:6", "link": "F,H"          },
            { "id": "H", "name": "H", "line": "L1:4,L2:4,L3:7", "link": "G"            }
        ]
    }
}
```

Need more information?

- I would recommend **Map::Tube::Cookbook** documentation for detailed description of internals of Map::Tube.
- For all other details, please refer to the documentation of **Map::Tube**.
- In case you still have any questions/suggestions, then please free to contact me by email (**mohammad.anwar@yahoo.com**).

Mini Challenge

- I would like to give you all a mini challenge to create simple map, having at least 2 lines with stations for now, and published to CPAN.
- Whoever do this first by end of today's workshop will receive a gift from me.
- To help you in your challenge, I have picked few maps that are still missing:
  - Paris
    - Download sample data (http://www.manwar.org/talks/paris-metro.json)
  - Madrid
    - Download sample data (http://www.manwar.org/talks/madrid-metro.json)
  - Mexico
    - Download sample data (http://www.manwar.org/talks/mexico-city-metro.json)
- Download source: (http://www.manwar.org/talks/Map-Tube-Trial-0.01.tar.gz)

Any Questions ?

Before I end my talk …

I would like to thank all my friends and families, especially …




And finally Thank you to the sponsors, without which the London Perl Workshop would not be possible:

Eligo, Perl Careers,
CV-Library, WCN,
Adzuna, Bytemark, OpusVL, Booking.com,
SureVoIP, Magnum Solutions,
Perl 6, Geekuni,
University of Westminster, Cogendo,
Science Photo Library,
The Enlightened Perl Organisation,
Evozon, O'Reilly.