Network Graph Visualisation Technology Report

Overview

Network graphs are a great way to understand relationships and identify groupings between objects. Uni Portfolio will take advantage of this by using network graphs to visualise the social networks users will have on the platform. Nodes will contain users' names and profile pictures, this will allow users to see all the people they have formed connections with and mutual connections between other users. Clicking on a node will redirect the user to the selected person's portfolio page.

Data Driven Documents (D3) [1] is a JavaScript library that can be used to create dynamic and interactive visualisations of data using SVG, HTML, and CSS. It contains a wide range of tools that can be utilised to bind data to visual elements in formats such as bar charts, bubble charts, force-directed graphs, and many others.

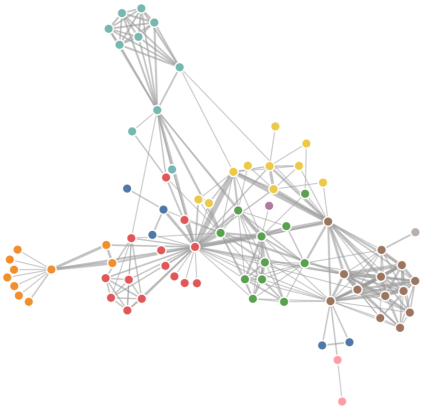

Image A: force directed graph created using D3.js and React-D3-Graph library

React-D3-Graph [2] is a component library that supplements D3.js by allowing users to create configurable and interactive graphs using React. The library provides a number of useful tools, including functions that handle events such as **`onClickNode`**, **`onMouseOverNode`**, and **`_onDragMove`**. These functions will make implementing features on the Uni Portfolio graph page simple. Using the online interactive demo [3], developers can create mockup configurations of force graphs, which can then be added into their own code using the data they wish to represent.

Implementation

Implementing a network graph will be made a lot easier using the react-d3-graph18 library if the user has experience with React, as it is essentially calling components.

```
function ForceGraph({
  nodes, // an iterable of node objects (typically [{id}, …])
  links // an iterable of link objects (typically [{source, target}, …])
}, {
  nodeId = d => d.id, // given d in nodes, returns a unique identifier (string)
  nodeGroup, // given d in nodes, returns an (ordinal) value for color
  nodeGroups, // an array of ordinal values representing the node groups
  nodeTitle, // given d in nodes, a title string
  nodeFill = "currentColor", // node stroke fill (if not using a group color encoding)
```

The code on the left [4] is a small section of a function (called **`ForceGraph`**) that takes data as parameters and manipulates it to create a force-directed graph. In the full example, you can see how the node attributes are set as well as how events such as dragging and clicking on nodes are handled by creating and calling other functions.

React-D3-Graph18 provides ready-made components that can be included in this function. This allows users to implement specific configurations without the need to create new functions manually.

A graph object (called **`chart`** in this instance) can then be created that calls the **`ForceGraph`** function and passes an array of data (**`miserables`**), along with node options, to return an SVG element, which can then be inserted into the DOM (Document Object Model) to render the graph seen in ***image A***.

```
chart = ForceGraph(miserables, {
  nodeId: d => d.id,
  nodeGroup: d => d.group,
  nodeTitle: d => `${d.id}\n${d.group}`,
  linkStrokeWidth: l => Math.sqrt(l.value),
  width,
  height: 600,
  invalidation // a promise to stop the simulation when the cell is re-run
})
```

```
miserables = ▼Object {
  nodes: ▶Array(77) [Object, Object, Object, Object, Object, Object, Object,
  links: ▶Array(254) [Object, Object, Object, Object, Object, Object, Object,
}
```

Resources:

[1] D3 Documentation - [https://github.com/d3/d3/wiki]
[2] React D3 Graph Documentation - [https://danielcaldas.github.io/react-d3-graph/docs/index.html]
[3] React D3 Graph Interactive Example - [react-d3-graph]
[4] Force-Directed Graph Code Example [https://observablehq.com/@d3/force-directed-graph]