Graphnet Pre-GSoC Task

TASK- go over the library I mention above (*Ant Design Charts*) and study and write the review on how they have implemented (a) canvas layouts, (b) node and edges position algorithms, and (c) ways of projecting the node and edges to the canvas.

SOLUTION

**Ant Design Charts** is a charting library built on top of G2Plot, a visualization library based on G2. The library provides various chart types, including line charts, bar charts, scatter plots, and graphs. The "**graphs**" package within the library specifically deals with rendering graphs using canvas.

(a) Canvas Layouts:

Ant Design Charts uses the Canvas API to render its graphs, which allows for direct manipulation of pixel data. The library uses a force-directed layout algorithm to position nodes on the canvas. This algorithm simulates a physical system in which nodes repel each other and are attracted to their neighbors, resulting in a visually pleasing layout. Ant Design Charts also provides support for **custom layout algorithms** through the use of the "layout" attribute.

(b) Node and Edge Position Algorithms:

Ant Design Charts implements several algorithms for positioning nodes and edges in a graph. For **node positioning**, the library offers several options such as the force-directed layout mentioned above, as well as a radial layout, a circular layout, and a grid layout.

For **edge positioning**, the library provides different types of curves, including straight lines, quadratic curves, and cubic Bezier curves. Users can also specify their own custom positioning algorithms.

(c) Projection of Nodes and Edges to the Canvas:

Ant Design Charts uses the canvas's 2D context to draw nodes and edges onto the canvas.

To project **nodes** onto the canvas, the library calculates their positions using the layout algorithm specified by the user, and then draws them onto the canvas using the specified node style.

For **edges**, the library calculates the positions of the start and end nodes and draws the edge between them using the specified edge style.

In Addition, Ant Design Charts offers several different **types** of graphs within its "graphs" package, including the decomposition tree graph, radial graph, and seven other types of graphs.

Here is a brief overview of some of the graphs available in Ant Design Charts:

**Decomposition Tree Graph**: This graph is used to show hierarchical relationships between data. It has a tree-like structure where nodes are connected by edges, and each node represents a subset of the data.

**Radial Graph**: This graph has a circular layout where nodes are placed along concentric circles, and edges connect the nodes. It is commonly used to show relationships between data points that are circular or cyclical in nature.

**Sankey Diagram**: This graph is used to show the flow of data or resources between different stages or components of a system. It uses horizontal or vertical bars of varying widths to represent the flow between nodes.

**Sunburst Graph**: This graph is used to show hierarchical data in a circular layout, similar to a pie chart. It has concentric circles representing different levels of the hierarchy, and the size of the sectors indicates the relative size of the data.

**Treemap Graph**: This graph is used to show hierarchical data using nested rectangles, where the size of each rectangle represents the relative size of the data. It is commonly used to visualize the distribution of data across different categories or groups.

**Waterfall Graph**: This graph is used to show how a starting value changes over time or across different categories. It uses bars to represent the change in value, with upward bars representing increases and downward bars representing decreases.

**Word Cloud Graph**: This graph is used to visualize text data, where the size of each word represents its frequency or importance. The words are arranged in a random or clustered layout, with the most important words placed in the center.

In conclusion, Ant Design Charts provides a comprehensive set of tools for rendering graphs using canvas. The library's force-directed layout algorithm and support for custom layout algorithms make it flexible enough to handle various graph types, while the different node and edge positioning algorithms and styles allow for customization of the graph's appearance. The library's use of the Canvas API provides fast rendering performance and the ability to directly manipulate pixel data.