**Project´s Design**

1. **Problem identification**

Graphs are a powerful data structure for modeling relationships and connections between entities in various applications in a real-world context. The goal of this engineering project is to design and implement an interactive user interface game using graphs as the underlying data structure. The game will involve navigating a virtual map with interconnected cities, where players can move between nodes and interact with the entities represented by the nodes. When the player selects to go to any city, the software will use the graph algorithm to efficiently traverse the relationships and calculate the optimal path for the player to reach their desired destination considering the fuel, traffic, and other variables.

1. **Gathering Necessary Information**

Graph

A graph is a data structure that can be used to represent a network of interconnected objects. It is made up of two sets: vertices (also called nodes or points) and edges (also called links or lines). Vertices represent the objects in the network, and edges represent the relationships between them. For example, a graph could be used to represent a social network, with vertices representing people and edges representing friendships. Or, it could be used to represent a transportation network, with vertices representing cities and edges representing roads.

* Breadth-first search (BFS): BFS is a graph algorithm that is used to find the shortest path between two nodes in a graph. It works by exploring the graph in a breadth-first manner, starting at the source node, and then exploring all its neighbors before moving on to the next level.
* Depth-first search (DFS): DFS is another graph algorithm that can be used to find the shortest path between two nodes in a graph. It works by exploring the graph in a depth-first manner, starting at the source node, and then exploring all its neighbors before returning to the source node and exploring the next neighbor.
* Dijkstra's algorithm: Dijkstra's algorithm is a graph algorithm that is used to find the shortest path between a single source node and all other nodes in a graph. It works by maintaining a queue of nodes that have not yet been visited, and then repeatedly visiting the node with the shortest distance from the source node.

Libraries

JavaFX:

JavaFX is a Java library that is used to create rich client applications. It provides a wide range of features for creating user interfaces, including graphics, animation, and multimedia. JavaFX is also a powerful tool for modeling graphs. It provides several classes and methods for creating and manipulating graphs, and it can be used to create a wide variety of graph-based applications.

* Creating graph objects: JavaFX provides several classes for creating graph objects, such as nodes, edges, and graphs. These classes can be used to create a variety of graph structures, such as directed graphs, undirected graphs, and weighted graphs.
* Visualizing graphs: JavaFX provides several methods for visualizing graphs. These methods can be used to create a variety of graph visualizations, such as node-link diagrams, arc diagrams, and tree maps.
* Manipulating graphs: JavaFX provides several methods for manipulating graphs. These methods can be used to add and remove nodes and edges, change the properties of nodes and edges, and perform graph algorithms.

JGraphT:

JGraphT and GraphStream are two popular Java libraries for modeling and analyzing graphs. JGraphT is a full-featured graph library that provides a wide range of features for graph manipulation, analysis, and visualization. It supports a variety of graph types, including simple graphs, multigraphs, and pseudographs. JGraphT also provides a variety of algorithms for graph analysis, such as shortest path, minimum spanning tree, and network flow.

GraphStream:

GraphStream is a Java library for modeling and analyzing dynamic graphs. It provides features for generating, importing, exporting, measuring, laying out, and visualizing dynamic graphs. GraphStream also provides several algorithms for analyzing dynamic graphs, such as community detection and event detection.

1. **Search For Creative Solutions**
2. **Transition From Idea Formulation to Preliminary Designs**
3. **Evaluation and Selection of the Best Solution**
4. **Preparation of Reports and Specifications**
5. **Design Implementation**