**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 nodes, where players can move between them and interact with the entities represented by the nodes.

1. **Gathering Necessary Information**

Graph

A graph is a data structure that can be used to represent a network of interconnected objects. It comprises 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 represent a social network, with vertices representing people and edges representing friendships. Or it could 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 breadth first, 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.

<https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/>

<https://www.simplilearn.com/tutorials/data-structure-tutorial/graphs-in-data-structure>

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.

<https://openjfx.io/>

<https://es.wikipedia.org/wiki/JavaFX>

<https://www.tokioschool.com/noticias/que-es-javafx-usos/>

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.

<https://jgrapht.org/>

<https://www.baeldung.com/jgrapht>

<https://dl.acm.org/doi/10.1145/3381449>

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.

<http://graphstream-project.org/>

<https://en.wikipedia.org/wiki/GraphStream>

1. **Search For Creative Solutions**

1st Solution: Dimensions

Players explore a world composed of an interconnected network of dimensions. Each node represents a different dimension with unique challenges, and players must navigate through the edges to progress. Players can manipulate the connections between dimensions to solve puzzles and overcome obstacles.

2nd Solution: Galaxies

A space exploration game where players control a spaceship traveling through a stellar graph. Nodes represent star systems, and edges are travel routes. Players must plan their routes and make strategic decisions to explore new worlds, trade, and face extraterrestrial challenges.

3rd Solution: Data Network

Players enter a virtual world where the graph represents a data network. Players must hack and navigate the graph by solving puzzles, overcoming security obstacles, and battling computer viruses to protect the system.

4th Solution: Empires Territory

A strategy game where players build and manage an empire based on a graph. Nodes are cities or colonies, and edges represent trade routes or alliances. Players must expand their empires, establish trade routes, and form strategic alliances while competing for control of the graph.

5th Solution: Road Explorer

A game focused on navigating a vehicle along a road network represented as a graph. The main objective of the game is to explore the graph's structure by traversing all the edges and visiting each city at least once.

6th Solution: The Maze

Players explore an interconnected labyrinth where nodes are rooms and edges are passages. The goal is to find the exit of the maze, but rooms constantly change position, making navigation a challenge. Players must solve puzzles and avoid traps in their quest for the exit.

1. **Transition From Idea Formulation to Preliminary Designs**

First Selection: Road Explorer

**Description:** A game focused on navigating a vehicle along a road network represented as a graph. The main objective of the game is to explore the graph's structure by traversing all the edges and visiting each city at least once. This game will require the implementation of a multigraph data structure to represent the road network, where different types of edges can represent different aspects of the transportation system, such as traffic conditions or road closures. Additionally, the game will incorporate a variety of traversal techniques, such as depth-first search or Dijkstra's algorithm, to provide players with different strategies for navigating the road network and completing the game objective efficiently.

* **Graph Map:** The game presents a graph map with nodes representing cities and edges symbolizing roads. Players start in a designated starting city and must traverse all the edges to visit each city.
* **Navigation:** Players control a vehicle that moves along the graph's edges. They can select connected edges to advance to the next city.
* **Exploration:** The goal is to efficiently explore the graph by covering all the edges and visiting all the cities without retracing any edge. The game tracks time and exploration efficiency.
* **Optional Challenges:** To increase complexity, you can add challenges such as time limits, movement restrictions, and obstacles on certain edges.
* Scoring and Leaderboard: Players can earn scores based on their efficiency in exploration. The game may include an online leaderboard where players can compare their results with others.

Second Selection: Data Network

**Description:** A game set in a virtual world where the graph represents a complex data network. Players take on the role of skilled hackers on a mission to navigate the data network, solve puzzles, overcome security obstacles, and battle computer viruses to protect the system. To create an immersive and engaging gaming experience, the user interface will be designed to allow players to interactively adjust views, providing them with optimized representations of the graph and its relational structures. By using a multiple-views technique, including a main view and context views, players will be able to easily identify their focused region in the current context and make informed decisions on their next moves.

* **Network Exploration:** The game presents a visual data network represented as a graph. Nodes represent critical data servers, and edges are the connections between them. Players must explore this network to uncover secrets and complete missions.
* **Puzzle-Solving:** Each node within the data network presents unique puzzles and security challenges. Players must solve these puzzles to gain access to the servers and advance through the network. Puzzles can vary in complexity and may involve logic, cryptography, and pattern recognition.
* **Security Obstacles:** To add complexity, the data network is guarded by security measures. Players must bypass firewalls, avoid security cameras, and outwit AI security programs. Failing to do so may trigger alarms and impede progress.
* **Virus Battles:** Throughout the game, players will encounter malicious computer viruses that threaten the network's integrity. Engage in strategic battles to neutralize these viruses, protect critical data, and maintain control over the network.
* **Story-Driven Missions:** The game features a compelling storyline with various missions, each contributing to a larger narrative. Players must complete objectives, retrieve sensitive data, and uncover hidden agendas.
* **Hacking Tools:** Players have access to a variety of hacking tools and gadgets that aid in puzzle-solving and battling viruses. These tools can be upgraded as players progress through the game.
* **Progression and Upgrades:** Successful mission completions and puzzle-solving earn in-game currency, which can be used to purchase upgrades for hacking tools and improve skills.

1. **Evaluation and Selection of the Best Solution**

Evaluation Criterion

* **Time (credit 4):** Due to the project assignment date, the time is limited to a few weeks. So, the project development must consider the time spent on research, documentation, and coding.
* **Knowledge (credit 3):** Due to research being limited to time, the project must be possible for the team to develop according to the current knowledge and aptitudes.
* **Creativity (credit 2):** The idea is challenging and has creative functionalities.
* **Problem Solution (credit 1):** The idea must work according to the solution expected for the problem.

Points: 0 - 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Idea | Time | Knowledge | Creativity | Problem Solution | Total |
| Road Explorer | 7 | 8 | 4 | 10 |  |
| Data Network | 4 | 6 | 7 | 10 |  |

**Selected Solution: Road Explorer**

1. **Preparation of Reports and Specifications**

Problem Specification:

Problem: Traversing a Graph implementing a Game

Input: City (node) to go.

Output: A message indicating if it is possible to do the trip to the desired city from the current position.

Considerations

The following cases must be considered to indicate if the trip could be done:

1. The path to each city has a weight. When a city is selected, depending on the fuel and the cost to go to the selected city, the trip can be done or not.
2. The trip can’t be done if there is any possible path to go to the selected city from the current position.
3. In case the vehicle doesn’t get enough fuel, the map indicates which cities have gas stations.

A screenshot of a computer

Description automatically generated

**Pseudocode**

If (!existPath(city)) {

Print “the city can’t be accessed from the current position”;

}Else{

Weigh <- calculateTotalWeight(edges);

If (!eneughtFuel(weight)) {

Print “You need to fill up the vehicle”;

}Else {

CurrentPosition <- city;

Fuel <- fuel - weight;

Print “enjoy the trip!”;

}

}

}

1. **Design Implementation**

Java implementation

Tasks:

1. Find the shortest path.
2. Calculate if there’s enough fuel.
3. Find the closest gas station.