**Project´s requirements**

Daniel José Plazas Cortés - A00400085

Sebastián Erazo Ochoa - A00400086

Juan Manuel Casanova Marín - A00400090

**Problem Specification Table**

|  |  |
| --- | --- |
| CLIENT | Aristizabal and Marlon |
| USER | Everyone |
| FUNCTIONAL REQUIREMENTS | FR01: Game dynamics  FR02: Graph implementation.  FR03: Dynamic change of Implementation.  FR04: Show score. |
| CONTEXT OF THE PROBLEM | A game is required, with a focus on the application and analysis of graph algorithms. The game must be modeled using graph structures, with a minimum requirement of 50 vertices and 50 edges. In addition, the solution must allow the application of at least two of the graph algorithms, including traversals over graphs (BFS, DFS), minimum weight paths (Dijkstra, Floyd-Warshall), and minimum covering trees (Prim, Kruskal).  Flexibility is required in the implementation of the network, with the possibility of alternating between at least two versions of the network without affecting the performance of the program. |
| NON-FUNCTIONAL REQUIREMENTS | NFR01: Efficiency of Graph Algorithms.  NFR02: Resource Optimization. |

**Functional Requirements Analysis Tables**

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R1: Game dynamics | | |
| Summary | The player should be able to select the city to go through buttons. When a city is selected, the current location is adjusted to the current city, and the fuel is consumed depending on how long the trip was. | | |
| Inputs | Input name | Data type | Selection or repetition condition |
| City | Button |  |
| Distance | double | Is the sum of the edges' weight |
| Result or postcondition | * Vehicle’s current location: city selected. * Fuel is reduced depending on the distance traveled. | | |
| Outputs | Output name | Data type | Selection or repetition condition |
| msg | String | “Enjoy the trip”  “Not enough fuel” |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R2: Graph implementation | | |
| Summary | The game must support the application of at least two graph algorithms, including paths over graphs (BFS, DFS), minimum weight paths (Dijkstra, Floyd-Warshall), and minimum spanning trees (Prim, Kruskal).  Additional Details:  The implementation of graph algorithms should allow players to interact effectively with the game.  Clear options should be provided for selecting and applying each algorithm according to the user's preference.  The application of these algorithms should be reflected visually in the game, allowing players to observe how solutions unfold within the context of the game, in this case, as the length of the trip and the weight of the trip.  For paths over networks (BFS, DFS), the game should clearly show the order in which cities are visited, providing a clear understanding of the exploration process.  For minimum weight paths (Dijkstra, Floyd-Warshall), the interface must represent the optimal paths selected, including visualization of the nodes and edges involved.  For minimum overlay trees (Prim, Kruskal), the application should clearly highlight the selected edges as part of the overlay tree, providing visual information about the construction of the tree on the city map. | | |
| Inputs | Input name | Data type | Selection or repetition condition |
| AdjacencyListGraph | List | It must have the necessary parameters to build the method as boolean hasWeight, boolean isDirected |
| AdjacencyMatrixGrap | Matrix | It must have the necessary parameters to build the method as |
| BFS | queue | It must have the necessary parameters to build the method as T source |
| DFS | Stack | It must have the necessary parameters to build the method |
| Dijkstra | PriorityQueue | It must have the necessary parameters to build the method as T source |
| Floyd-Warshall | Matrix | It must have the necessary parameters to build the method |
| Prim | PriorityQueue | It must have the necessary parameters to build the method as  T source |
| Kruskal | disjoint-set union | It must have the necessary parameters to build the method |
| Result or postcondition | The application of these algorithms should be reflected visually in the game.  For paths over networks (BFS, DFS), the game should clearly show the order in which cities are visited, providing a clear understanding of the exploration process.  For minimum weight paths (Dijkstra, Floyd-Warshall), the interface must represent the optimal paths selected, including visualization of the nodes and edges involved.  For minimum overlay trees (Prim, Kruskal), the application should clearly highlight the selected edges as part of the overlay tree, providing visual information about the construction of the tree on the city map. | | |
| Outputs | Output name | Data type | Selection or repetition condition |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R3: Dynamic change of Implementation. | | |
| Summary | There must be two possible game implementations: adjacency list and matrix. The player must be able to select the desired implementation, allowing switching between the two implementations without affecting the performance of the program.  The application should be able to manage the transition between network versions transparently to the user, without noticeable interruptions to the game experience. | | |
| Inputs | Input name | Data type | Selection or repetition condition |
| selection | int | 1-2 |
| Result or postcondition | * The implementation of graph algorithms should allow players to interact effectively with the game. * Clear options should be provided for selecting and applying each algorithm according to the user's preference. | | |
| Outputs | Output name | Data type | Selection or repetition condition |
| SwitchMode | ComboBox |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R4: Show score. | | |
| Summary | The system should calculate the player's score based on the efficiency in managing the distance traveled compared to the target distance, providing a quantitative measure of the player's performance in route planning. | | |
| Inputs | Input name | Data type | Selection or repetition condition |
| DistanceObjetive | int |  |
| RealDistance | int |  |
| Calculus | int |  |
| Result or postcondition | After the player completes a route, the system will calculate the final score based on distance efficiency and other relevant factors. This score will be associated with the player's route planning performance and will be used to evaluate his success in the game. | | |
| Outputs | Output name | Data type | Selection or repetition condition |
| Score | msg | Win or Lose condition |