Computer Graphics and Product modelling

ME 735

Submitted by: Group 9

213104014	Galla Rama Srinivas (Team Leader)
213101001	Deepak Kumar Thakur
213100053	Thirumalasetty Sravan Sai Kumar
213100061	Yogesh Shrikant Sale
213101002	Bijay Kumar Shah

Under the Guidance of

S S Pande

Department of Mechanical Engineering

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

Powai, Mumbai 400076

02, November 2021

Abstract:

In this Project, we are defining a 2D layout with obstacles and defined pathways and we are working on finding an optimized path to cover the distance between the user defined starting and final destination points in a path where distance travelled is the minimum of all the possible paths between the two said points. The shortest (optimized) path is generated using Dijkstra's Algorithm with Python as the coding software.

Introduction:

Project Title:

Graphical representation of shortest travel path in the given layout between two locations.

Objective:

Generate a 2D based layout and find optimized path on the layout to move from one location to other in the shortest path and interpret the same graphically.

Deliverable:

1. Graphical work to replicate the layout (in 2D)/2D simulator-based layout generation.
2. Optimization algorithm to move from one point to other in shortest or best path.
3. Merging Graphical part with Optimization.

Programming Languages used:

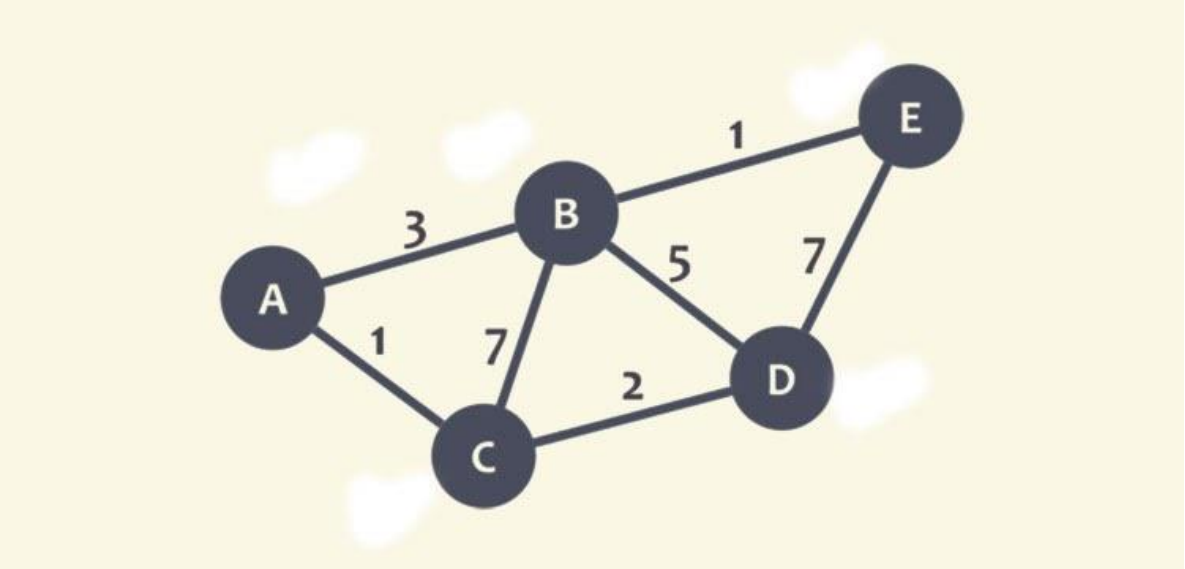
Python (libraries used: NumPy, matplotlib, pygame, queue, math, random (future scope))

Algorithm Used:

Dijkstra’s algorithm

It is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

The algorithm exists in many variants. Dijkstra's original algorithm found the shortest path between two given nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.



For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road (for simplicity, ignore red lights, stop signs, toll roads and other obstructions), Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. A widely used application of shortest path algorithms is network routing protocols, most notably IS-IS (Intermediate System to Intermediate System) and Open Shortest Path First (OSPF). It is also employed as a subroutine in other algorithms such as Johnson's.

The Dijkstra algorithm uses labels that are positive integers or real numbers, which are totally ordered. It can be generalized to use any labels that are partially ordered, provided the subsequent labels (a subsequent label is produced when traversing an edge) are monotonically non-decreasing. This generalization is called the generic Dijkstra shortest-path algorithm.

Methodology:

1. Generate a fixed 2D layout image of size 20X20 pixels.
2. Allocate a value of "0" to the obstacles and "1" to the pathways.
3. Identify nodes in the image of the layout.

   Note: Nodes are the intersections of the pathways present in the layout.
4. Rearrange the nodes in ascending order as an adjacency list and record the distances between the nodes in a N X N matrix (N=Number of nodes).
5. Convert adjacency list to Dictionary list for the implementation of Dijkstra's algorithm.
6. Adjust the dictionary list to provide input to Dijkstra's algorithm as a graph with vertices defined.
7. Dijkstra's Algorithm.
8. Input start and end vertices for starting and ending points of the path respectively.
9. Call Dijkstra's algorithm to compute the optimized path and print out the node vertices along the optimized path.
10. Invert the Y-axis of the layout for plotting the optimized path using Matplotlib, as the layout has inverted Y axis and we would like to locate origin on bottom-left part of the layout.
11. Covert nodes to points on the layout.
12. Plot an animated path from starting to end point on the layout.

Extra work done:

Library Used:

Pygame, queue, math

1. Used pygame for making the process interactive.
2. Allow user to select the start point with mouse
3. Allow user to start and restart process with “S” and “N” key from keyboard respectively

Future scope:

To have to different starting points and one common end point, they must not collide with each other.

Results and Conclusion:

1. Successfully generated a layout and the shortest path between 2 nodes in the layout using Dijkstra’s algorithm.
2. Plotted the layout and optimized path graphically as the output of the Algorithm.

As declared in the objective, we were successfully able to Generate a 2D based layout and find optimized path on the layout to move from one location to other in the shortest path and interpret the same graphically.