Applications of Graph Theory For Optimal Route Planning with User-Defined Waypoints In The Crew 2

Kenneth Ricardo Chandra - 13523022[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
www.20ricardo05@gmail.com, 13523022@std.stei.itb.ac.id

***Abstract*—Navigation plays a critical role in both real-world and virtual environments, especially in open-world games where exploration is a core mechanic. One such game, a racing game, The Crew 2, features an expansive map modeled after the United States of America, enabling players to explore various terrains using vehicles such as cars, boats, and planes. Given the game's focus on racing, effective navigation is essential due to the vast map size, which spans approximately 1900 in-game miles² or 3000 km². The developers implemented waypoint and routing systems to assist players in navigating the large environment. However, these systems face limitations when dealing with complex terrains, particularly for ground vehicles. This paper explores the use of graph theory as a method for optimizing route planning in *The Crew 2*. By simulating graph-based routes, we demonstrate the potential for improving navigation systems to handle diverse terrains and obstacles more effectively. The study highlights the feasibility of integrating graph theory into route planning systems, similar to real-world GPS applications, to create more adaptive and efficient navigation tools in open-world games.**

***Keywords*—Navigation, Waypoint, The Crew 2, Ground Vehicles.**

I. Introduction

Navigation has been a core part of the world since the age of exploration and even way past before that. In everyday life, navigation is needed to get to the places people need to go and that applies too in the world of games. Games, especially open world games, have a lot of areas that can be explored by the players, just like the real world, where there are a lot of places to be explored. One of these games is The Crew 2, where the field of play is based on The United States of America, making it almost an identical copy of the real world. However in this game, everything is dictated by racing festivals, whether it is by land, water, or air, which is why the game is about racing. The player, outside of the player's home and the crew's base, can only use vehicles such as cars, motorbikes, boats and planes. This makes almost every part of the game to be played while the player is riding a vehicle.

As the game is about racing and vehicles, navigation is of course a core part of the game, as the map has a total size of around 1900 in game miles^2 or around 3000 in game km^2 which makes it really big and even for the fastest vehicle in the game, it would take a pretty long time to go through the whole map, not to mention if the vehicle is land based, as the terrain varies greatly depending on the location. There are some in-game objectives as well that require the player to go to a specific location in the map from another starting point which could make it hard for the player to navigate to the place they are required to go to especially if the player doesn't know the map well. For that exact reason, the developers of the game made a waypoint system where there are some waypoints that can be tracked down and the route towards that waypoint.will be shown to the players on the minimap so the player could follow the given route to head to the waypoint. As the map is big, the developers had also made a custom waypoint system so that players can easily mark wherever the player wanted to go and then the game would make a route towards that custom waypoint. The game automatically routes the player towards the waypoint using the nearest route available that is dependent on the endpoint. If the endpoint is somewhere accessible by normal roads, while the player is using a ground vehicle like a car, the route given by the game is to follow the roads leading up to the waypoint, giving the player the nearest route available. However if the endpoint is somewhere not accessible by normal roads like in the middle of a forest, the game automatically will give the player a straight line towards the waypoint, making it the nearest route available. This however, becomes a problem when the player is using a ground vehicle because as stated before, the terrains in the game vary greatly. There could be a mountain dividing the player and the endpoint which meant that the player couldn't just go straight towards the endpoint as the game suggests. Which is why there needs to be another route planning system for the game to follow for these kinds of waypoints which might be achievable using graph theory. By using graph theory, it becomes possible to make the best route possible for land vehicles to traverse the terrains of the game as graph theory can also account for different in-game conditions such as the mountain stated previously. Graph theory has also been used in real world applications like GPS that uses roads as edges and intersections as nodes which would allow the GPS to give the best possible route to go. Which is the reason why the graph theory is chosen to make the optimal route planning for the user-defined waypoints.

II. Theoretical Basis

A. Video Games

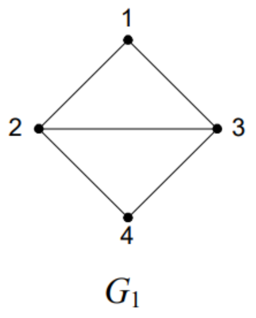
Games are something that people use to get through time by playing them and that applies as well to video games. Video games are a type of game that can be played by using various devices and has a display on a screen for the player to see. Video games, as the name implies, is a form of game that uses videos that can be interacted with by the player by using inputs like a controller, a keyboard and even touchscreens. There are a lot of video games so much so that it can be said that almost everything has a video game, in one form or another. Video games have a lot of genres, starting from the usual genres like action, adventure, fantasy, and to the special genres of video games like shooter games, MMORPG's, and racing games. The Crew 2 is one of the most popular racing video games to date because it has great graphics and has a pretty good control. The Crew 2 is also a unique racing game as the game revolves around the player being a newbie that joins a few racing crew that goes on to race in a lot of events. These racing crews have their own specialties like street racing, offroading, freestyle, and pro racing. These specialties have their own vehicles to be used based on what the race is, although the game mainly focuses on racing with cars, boats and airplanes, there are also jet boats, motorbikes and rally cars that can be used as a variety. The Crew 2 takes place in a modified United States of America, which means that in a way, the game takes place in a real life location. Although not all of the states are in the game, most of the popular locations are included such as Las Vegas, Los Angeles, Miami and San Francisco.

B. Terrains and Waypoints

Terrains are a form of land in the world and that applies to in video games. There are a lot of kinds of terrains in the world like the desert, the grasslands, the rainforest, tundra and much much more. In The Crew 2, there are a lot of varieties of terrains as well because the game takes place in The United States of America which has a lot of cities, a lot of grasslands, deserts, and much much more. Due to these vast varieties of terrains, it would be hard to navigate through the world of The Crew 2, if not for the main roadway that is in the game which can be used to get anywhere. Players are able to traverse through the off road however if the player does not use the suitable vehicle, the player will have a harder time traversing through those roads. There are a lot of roadways leading to various places, leading the players to explore more. This however becomes a problem as more roads means more options and more options means harder navigation. To solve this problem, the developers of The Crew 2 have made a waypoint system which can be used to fast travel or to mark the location of where the player wants to go. There are two kinds of waypoints in the game, pre-made waypoints and user-defined waypoints. Pre-made waypoints are made by the game to show the important places in the game, such as the home of the player, and mainly racing events where the player have to travel to so that they can start the race. The players also have the option to either make an automatic route to the waypoint or to immediately fast travel to the waypoint. The other waypoint however, is made by the player. Players have the ability to mark anywhere on the map and it will route the player to the waypoint through the nearest path. There are also two kinds of route, a direct route which means the path to the waypoint is a direct straight line that also shows the distance between the player and the waypoint and the other one is a route that uses the roadways as paths. The game will try to find the nearest path to the waypoint using the roadways and will navigate the player through the roads. This however, only works if the waypoint is placed on a place accessible directly by the road. If the waypoint is on a random place, it will automatically use the direct route and will never change to the roadway paths. This also applies if the player is using other vehicles other than ground vehicles such as boats and planes.

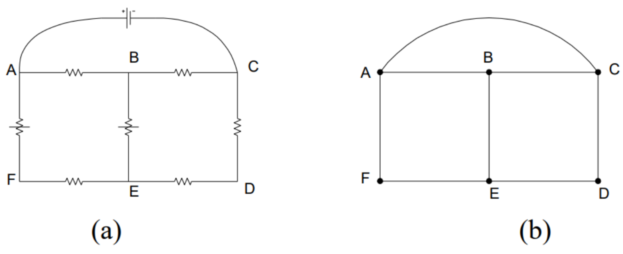
C. Graph Theory

Graph theory is a theory about graphs which is normally used to represent objects and the connections between the objects. There are a lot of applications using graphs and some of them are very useful such as the GPS in the real world and in the video game, the route given to the players by The Crew 2. A graph needs to have a minimum of a node or vertices and no edges are needed to construct a graph. There are a lot of kinds of graphs, such as simple graphs, pseudo graphs which contain loops, multi graphs which contain multiple edges and much much more. This is an example of a simple graph.



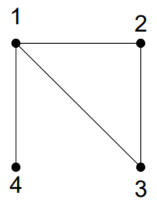
Picture 1. Simple Graph example with 4 vertices and 5 edges.

As mentioned previously, graphs are used to represent connections between objects such as a food chain or an electric circuit.



Picture 2. Example of graphs as an electric circuit design

In a graph, there also exists paths and circuits. A path is a line of connected vertices that goes from vertice A to vertice B while a circuit is a path that goes back to its own vertice, for example, a route is taken from point A goes to point B and then goes back to point A. To make the shortest route of a graph, there are some theorems that can be used to create the path and circuits. A path that goes through each vertices is called a Hamilton Path which can also be said is the shortest way in a graph.



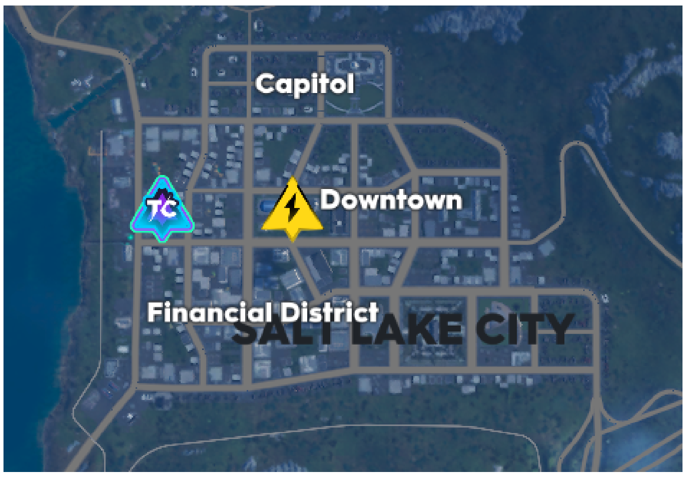
(a)

Picture 3. Example of a graph that has a Hamilton Path.

Every connected graph contains at least a spanning tree. A spanning tree is an epigraph of the original graphs that contains all of the vertices of the original graph These spanning trees are used to make routes, which is the core of this paper's discussion. Minimum spanning trees, the version of a spanning tree but has the shortest length. can be made by using Prim's Algorithm and Kruskal's Algorithm. Both of these algorithms can be used to create the minimum spanning tree or by context, the shortest route towards an endpoint. Prim's Algorithm works by taking the shortest edges of a graph, then continuing on to the next shortest edges, connecting both edges with the only exception if the connection makes a circuit then repeat the process until every vertex has been traveled to. Kruskal's Algorithm is a bit different in the way that Kruskal's requires you to list all of the available routes, starting from the shortest ones and connects it to the next shortest ones (or leave floating if there are no connections between the routes), with the same exception of not connecting with the routes if it makes a circuit. Dijkstra's Algorithm is used to find a certain node in a connected, closed graph like a road network.

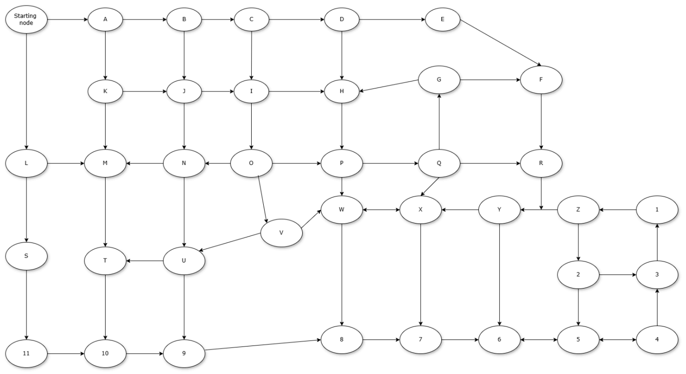
III. Data

The data used in this paper is the map of The Crew 2, especially in the town of Salt Lake City which from the original map, will be converted into a graph, which will also contain a waypoint to compare the game's routing ability with the algorithm used to make the minimum spanning tree.



Picture 4. Map of Salt Lake City from The Crew 2

Each road works as the edges and each turn acts as a vertex. This is the diagram made from the in game map which is not exactly a 1:1 replica but just a representation.



Picture 5. Map of Salt Lake City from The Crew 2 in the form of a graph

Using this map, it is possible to create the graphing to make the best route towards a certain point. In a table, here are all the connecting vertices in a list.

Connections		
(Start,A)	(A,B)	(B,C)
(C,D)	(E,F)	(F,G)
(G,H)	(H,I)	(I,J)
(J,K)	(L,M)	(M,N)
(N,O)	(O,P)	(P,Q)
(Q,R)	(T,U)	(U,V)
(V,W)	(W,X)	(X,Y)
(Y,Z)	(Z,1)	(3,2)
(4,5)	(5,6)	(6,7)
(7,8)	(8,9)	(9,10)
(10,11)	(Start,L)	(A,K)
(B,J)	(C,I)	(D,H)
(F,R)	(G,Q)	(H,P)

(I,O)	(J,N)	(K,M)
(L,S)	(M,T)	(N,U)
(O,V)	(P,W)	(Q,X)
(R,Y)	(R,Z)	(Z,2)
(1,3)	(3,4)	(2,5)
(Y,6)	(X,7)	(W,8)
(U,9)	(T,10)	(S,11)

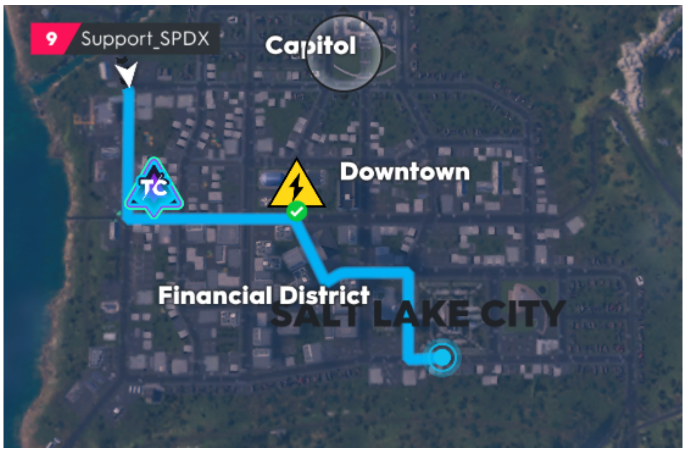
Table 1. List of Connecting Vertices

A program in python can be made to calculate the minimum spanning tree to get to the point desired. The program uses BFS as giving the routes specific lengths is difficult, so to get the shortest route based on the vertices available. Here is the program.

```
class Graph:
    def __init__(self):
        self.graph = {}

    def add_edge(self, u, v):
        if u not in self.graph:
            self.graph[u] = []
        if v not in self.graph:
            self.graph[v] = []
        self.graph[u].append(v)
        self.graph[v].append(u)

    def bfs_shortest_path(self, start, goal):
        visited = set()
        queue = [[start]]

        if start == goal:
            return [start]

        while queue:
            path = queue.pop(0)
            node = path[-1]

            if node not in visited:
                neighbors = self.graph[node]
                for neighbor in neighbors:
                    new_path = list(path)
                    new_path.append(neighbor)
                    queue.append(new_path)

                    if neighbor == goal:
                        return new_path

                visited.add(node)

        return None


def main():
    # Initialize graph
    g = Graph()

    # Add edges based on given data
    edges = [
        ('Start', 'A'), ('A', 'B'), ('B', 'C'), ('C', 'D'), ('E', 'F'), ('F',
'G'),
        ('G', 'H'), ('H', 'I'), ('I', 'J'), ('J', 'K'), ('L', 'M'), ('M', 'N'),
        ('N', 'O'), ('O', 'P'), ('P', 'Q'), ('Q', 'R'), ('T', 'U'), ('U', 'V'),
        ('V', 'W'), ('W', 'X'), ('X', 'Y'), ('Y', 'Z'), ('Z', '1'), ('3', '2'),
        ('4', '5'), ('5', '6'), ('6', '7'), ('7', '8'), ('8', '9'), ('9', '10'),
        ('10', '11'), ('Start', 'L'), ('A', 'K'), ('B', 'J'), ('C', 'I'), ('D',
'H'),
        ('F', 'R'), ('G', 'Q'), ('H', 'P'), ('I', 'O'), ('J', 'N'), ('K', 'M'),
        ('L', 'S'), ('M', 'T'), ('N', 'U'), ('O', 'V'), ('P', 'W'), ('Q',
'X'),
        ('R', 'Y'), ('R', 'Z'), ('Z', '2'), ('1', '3'), ('3', '4'), ('2', '5'),
        ('Y', '6'), ('X', '7'), ('W', '8'), ('U', '9'), ('T', '10'), ('S', '11')
    ]

    for u, v in edges:
        g.add_edge(u, v)

    # Input destination
    destination = input("Enter the destination point: ")

    # Find shortest path
    path = g.bfs_shortest_path('Start', destination)

    if path:
        print("Shortest route: ", ' -> '.join(path))
    else:
        print("No route found to the destination.")


if __name__ == "__main__":
    main()
```

The program checks if the path is correct, and if the path is the shortest route based on the vertices available.

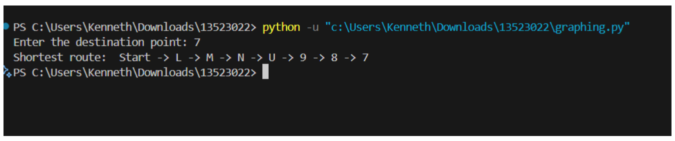
IV. Implementation



Picture 6. Routing from The Crew 2 to Vertex 7 in the game.

As a test, the desired point is at vertex 7 and by using the program, it can be located using the program

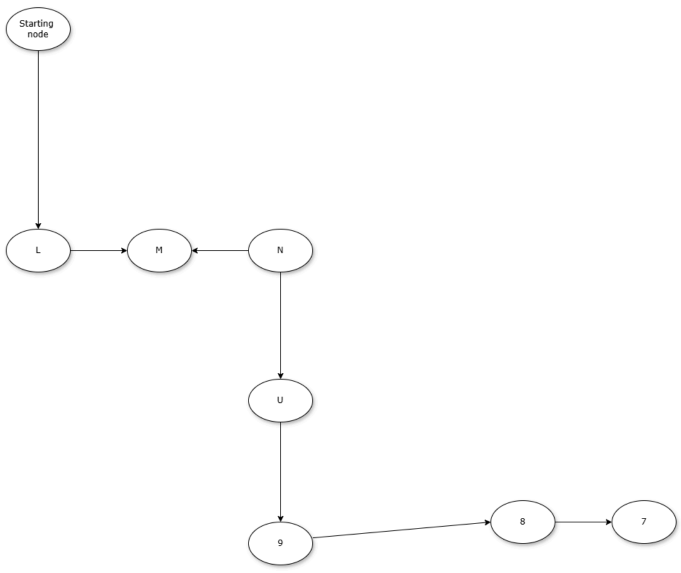
Here are the results



```
PS C:\Users\Kenneth\Downloads\13523022> python -u "c:\Users\Kenneth\Downloads\13523022\graphing.py"
Enter the destination point: 7
Shortest route:  Start -> L -> M -> N -> U -> 9 -> 8 -> 7
PS C:\Users\Kenneth\Downloads\13523022> 
```

Picture 7. Program’s shortest route to Vertex 7

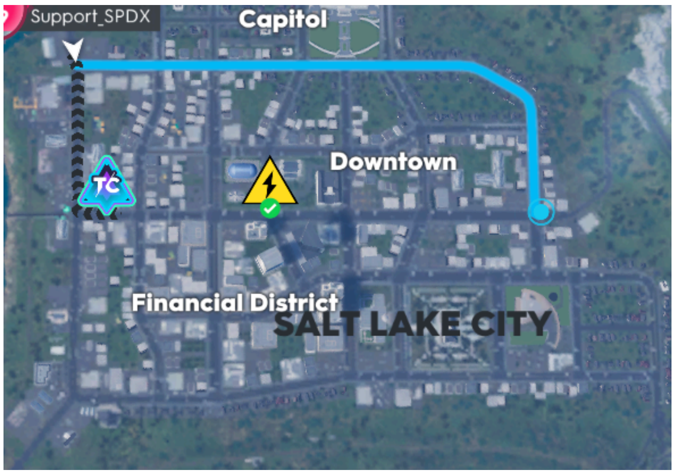
The system goes from Start -> L -> M -> N -> U -> 9 -> 8 -> 7 and this is how it looks in the graph.



Picture 8. Routing from Start to Vertex 7 based on the program

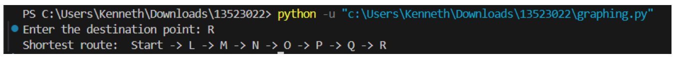
Compared to how the game makes the route, it is a bit different since there are no actual lengths being calculated in the program. However, it is still one of the shortest routes available that can be used to get to vertex 7, especially if it is based on vertices only.

Another test is conducted to get to vertex R and the in game result shows



Picture 9. Routing from The Crew 2 to Vertex R in the game.

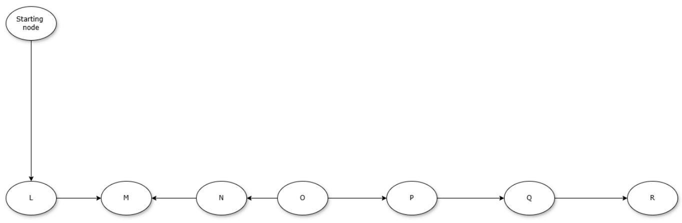
Based on the program, the result of the shortest route is Start -> L -> M -> N -> O -> P -> Q -> R



```
PS C:\Users\Kenneth\Downloads\13523022> python -u "c:\Users\Kenneth\Downloads\13523022\graphing.py"
Enter the destination point: R
Shortest route:  Start -> L -> M -> N -> O -> P -> Q -> R
```

Picture 10. Program’s shortest route to Vertex R

And this is the program’s route in the graph



Picture 11. Routing from Start to Vertex R based on the program

From two of these tests, it can be seen that the game and the program have different ways of routing. Since the program itself does not have the weights or the lengths of the edges, there are some differences that can be visibly seen by the in-game screenshots and the graph results. This means that technically, the game uses the graph theory for optimal route planning as the game has the more optimal route and has already built in lengths. This is however impossible to be researched as the exact lengths of the in-game roads are not available to the players, making it hard to test the data. However, the test result shows that there are still some other routes available to be used to go towards the desired waypoint as there are some other variables to be considered with in the game. Using the routes provided by the program and the game does not really yield a big time difference which is due to largely a small sample. This shows too that routes given are not necessarily the shortest as the shortest actual route is by going in a straight line towards the waypoint, but since that is impossible to do in a ground vehicle, the player needs to swap to an airborne vehicle to do that.

V. Conclusion

To conclude, The Crew 2 have been using the graph theory to make optimal route planning which can be seen by the simulation and differences on the graph. There are not many differences, even if the program doesn’t have the weights or the lengths. However, this still needs to be tested more as the sample is too small and there is not much data that can be used. This small sample also shows, however, that the optimal route planning is available and can be used by using graph theory.

VI. Acknowledgment

Before this paper is finished, I would like to express my deepest gratitude to those who have supported me throughout this journey. First, I would like to thank God for guiding me and being my constant source of strength and inspiration since the beginning of my life. Next, to my family whom I am forever grateful for your unwavering support and belief in me, which has been the foundation of all my achievements. I also extend my thanks to my friends in IF’23 ITB, who have been my trusted companions and to my teachers and professors, I deeply appreciate your dedication and exceptional guidance, which has been invaluable in helping me complete this paper. Finally, to you, the reader, thank you for taking the time to engage with this paper. I hope this paper helps.

References

[1] Munir, R. (2023). Nilai eigen dan vektor eigen (bagian 1). Informatika STEI ITB. Retrieved January 8, 2025, from Munir, R. (2023). Nilai eigen dan vektor eigen (bagian 1). Informatika STEI ITB. Retrieved January 2, 2025, from https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2023.pdf

[2] Munir, R. (2023). Nilai eigen dan vektor eigen (bagian 1). Informatika STEI ITB. Retrieved January 8, 2025, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf

[3] Munir, R. (2023). Nilai eigen dan vektor eigen (bagian 1). Informatika STEI ITB. Retrieved January 8, 2025, from https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/23-Pohon-Bag1-2024.pdf

[4] *The Crew 2*. 2018. Developed by Ivory Tower. Published by Ubisoft. PC.

Statement