

TSP Solution Documentantation

Before starting the project, I thoroughly understood the task given and decided which technologies I would use. I chose to use the Unity Game Engine because I was asked to solve the TSP problem in a 3D environment with the C# programming language.

First of all, I designed a UI in line with the required items in the project. Then, I created the cities (nodes) in runtime and started coding.

My first goal was to create cities in runtime, find the distance of each to the others and give each a unique name. To do this, I first learned how many cities the user wanted, and then created the button.

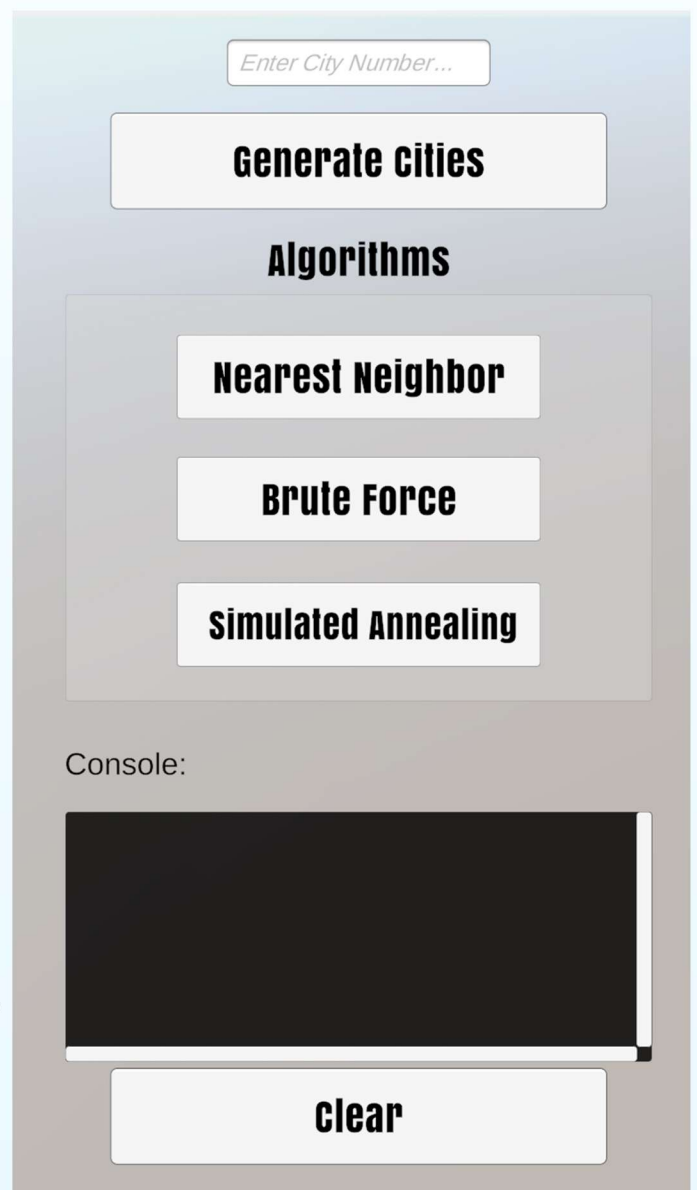
Afterwards, I created the console section for both debugging and informing the user.

Now that I am creating my cities dynamically and keeping them in a collection, I started working on algorithms.

The first algorithm I worked on to solve this problem was the Nearest Neighbor algorithm. Since it worked better than Brute Force in terms of performance, it helped me to check the codes I had written before.

Later, I chose Brute Force for my second algorithm. Although it was not good in terms of performance, I always found the shortest and ideal path. (Performance drops significantly after 8 cities, if you are going to test it)

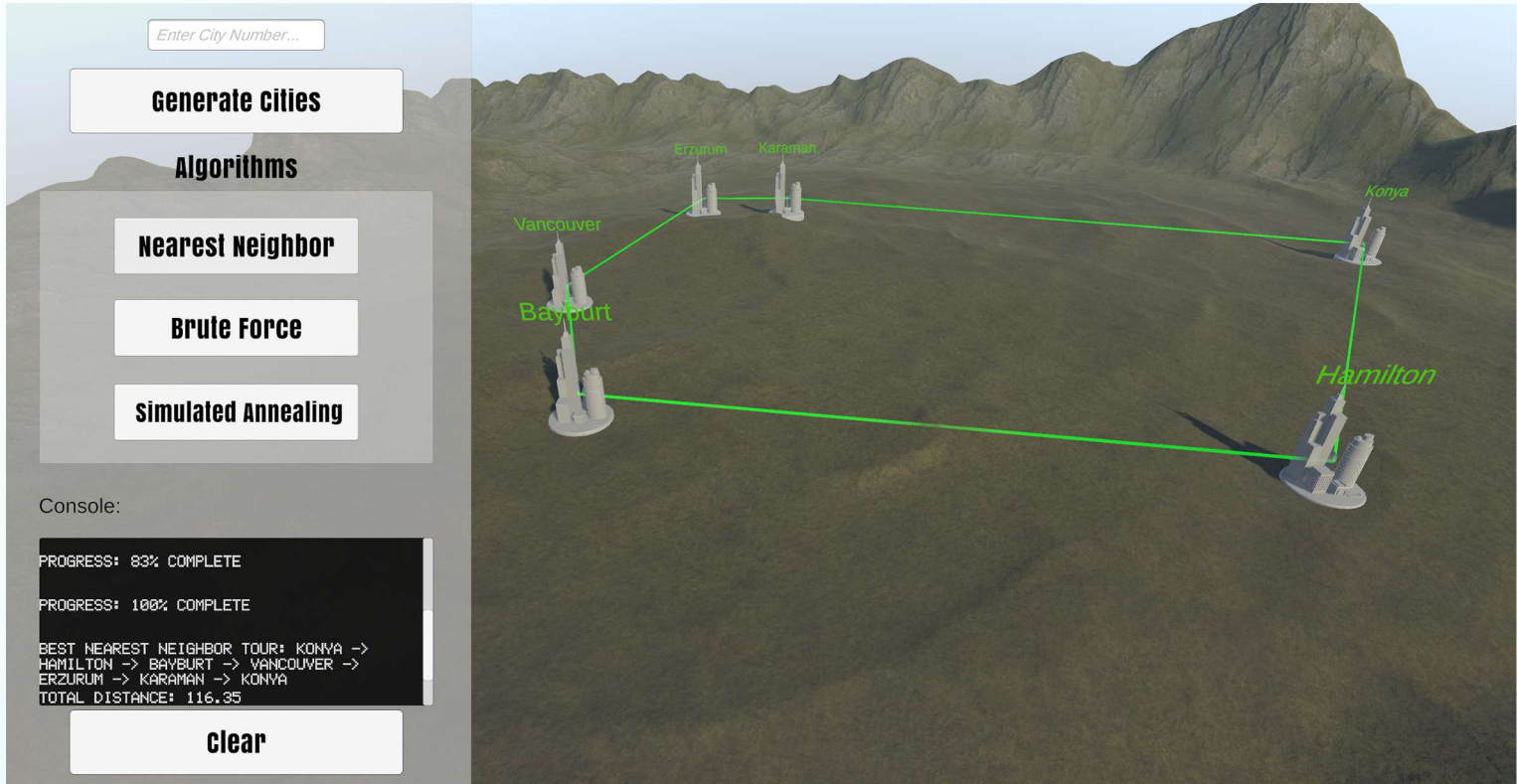
Creating a button to clear and if the users wants, They can clear the currently created city objects and console outputs and create new ones.



The image shows a user interface for a Traveling Salesman Problem (TSP) solver. At the top, there is a text input field with the placeholder text "Enter City Number...". Below this is a large button labeled "Generate cities". Underneath the button is a section titled "Algorithms" in bold. This section contains three buttons: "Nearest Neighbor", "Brute Force", and "Simulated Annealing". Below the algorithm buttons is a label "Console:" followed by a large black rectangular area representing the console output. At the bottom of the interface is a button labeled "clear".

After making sure that the algorithms were working properly, I created lines (Linerenderer) in the runtime to provide feedback to the intercity user.

Finally, for visual purposes, I downloaded terrain textures from the Unity asset store and created small, simple city diagrams in Blender.



In order to develop the project further, the path taken may vary depending on the terrain (For example, a road cannot pass over a river). Different algorithms can be added. The appearance of 3D models may be different depending on the size of the cities (Istanbul is a larger 3D object, while Siirt may be smaller).