Traffic Modeling with Unity 3D

Isaiah Martinez
CSUN
Mathematics Department
isaiah.martinez.891@my.csun.edu

04/21/2024

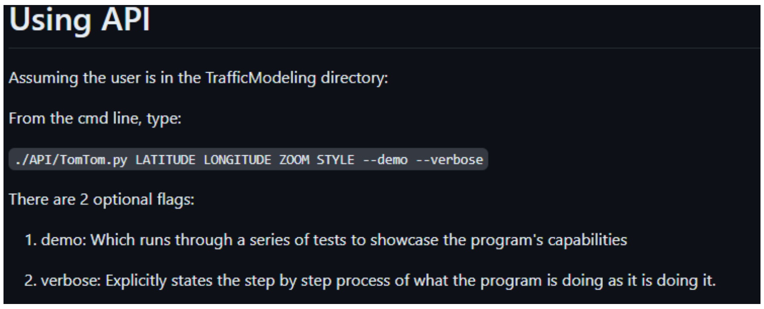
Contents

1 Change History

Version: 0.42
Modifier: Isaiah Martinez
Date: 4/21/24
Description of Change: Finished TomTom API using Python. Made Python Script accessible via Command line.

2 Progress

Where are we with design/implementation?
Full implementation of the Python Script that connects to the TomTom API and utilization of the Command Line interface to obtain image of a particular Longitude and Latitude with zoom and image style.


Using API

Assuming the user is in the TrafficModeling directory:

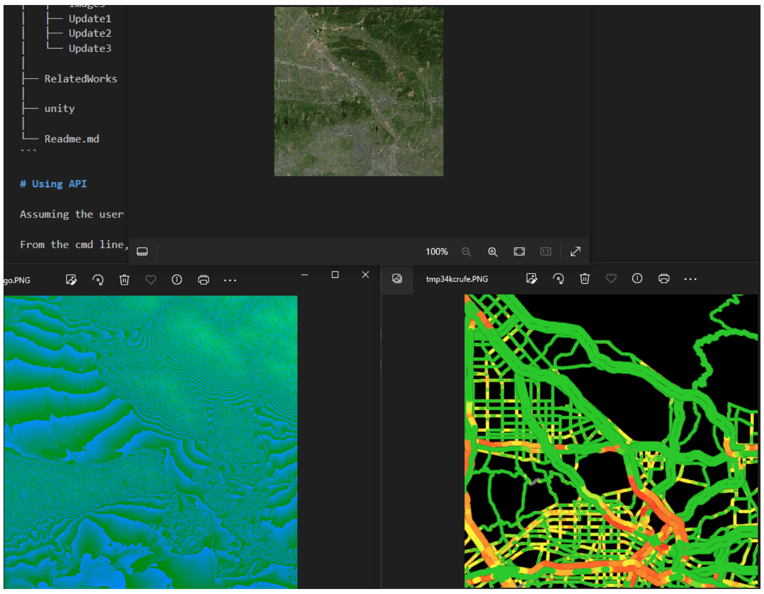
From the cmd line, type:

```
./API/TomTom.py LATITUDE LONGITUDE ZOOM STYLE --demo --verbose
```

There are 2 optional flags:

1. demo: Which runs through a series of tests to showcase the program's capabilities
2. verbose: Explicitly states the step by step process of what the program is doing as it is doing it.

Figure 1: How to use the Python Script.


```
│     ├── Update1
│     ├── Update2
│     └── Update3
│
├── RelatedWorks
│
├── unity
│
└── Readme.md
```

Using API

Assuming the user

From the cmd line,
```

Figure 2: Demo from script.
```

Starting to think about the UI/UX of the program a little more and changed the camera position to follow the car moving from a distance. We also baked a new pathfinding scene from a new network of roads and traffic that include intersections and more streets.

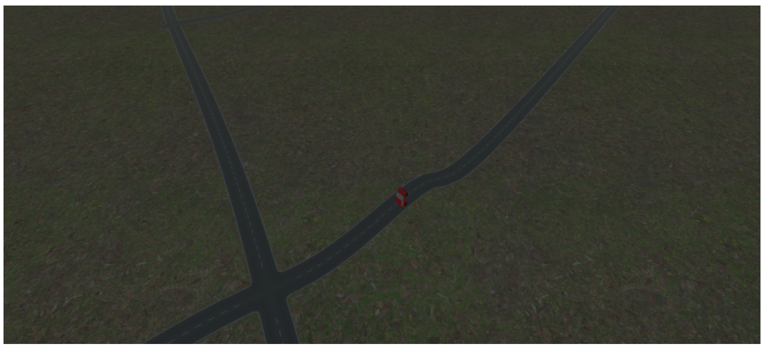


Figure 3: Camera view of scene from game screen.

With that, we started testing Unity's built in pathfinding function and seeing what limitations it has. A limitation we found was that between two roads, the pathfinding would have a preference for the road that it was closest to. If it was already on the road, it would continue further down. The issue here is that if there is an obstacle at the end of the road, then the object would continue down the path then be stuck when it ran into it. We would need to have a system that it could "see down the road" and choose to go the other way or it would continue down a road then turn around when it runs into an obstacle.

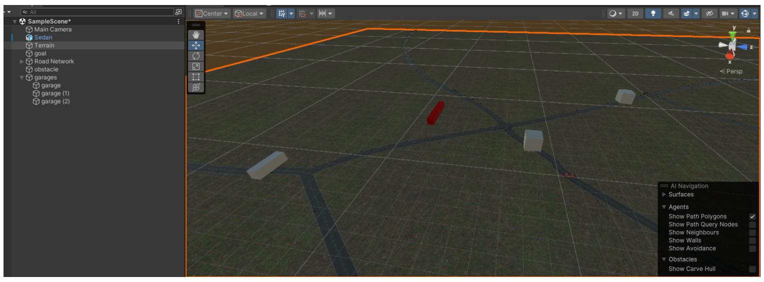
As for the road network from the image processing may be more difficult than expect. As for now, we are currently just using a hand made road network. If it proves to be more challenging than the time allows, then we will switch gears and create hand-made road layouts and then primarily focus upon the pathfinding AI.

3 Challenges

What challenges have occurred for this week?

Learning how to implement a list of options as well as type checking in Python was a very weird process. Python works off of the Ostrich Algorithm which is really funny to see, but horrifying when trying to work with particular types.

Another challenge we ran into that will be worked on in our next "sprint" is placing the traffic and placing moving obstacles on the road that is the traffic. I thought of placing spawnpoints on the street that a car Gameobject would spawn with the Sedan prefab and immediately assume moving on the road based on the default pathfinding function, but the current issue I'm running into is attaching that to the new gameobjects and how to make them recognize eachother as obstacles. The car could move, but it would just phase through the other cars, and the main car that will be caring about the end goals has a different function from the spawned in cars that I'll have to figure out if there's a more efficient way to script that or that I'll just continue with another script and worry about efficiency later.



AI Navigation
Surfaces
Agents
Show Path Polygons ✓
Show Path Query Nodes
Show Neighbours
Show Walls
Show Avoidance
Obstacles
Show Carve Hull

Figure 4: The grey boxes that are on the road, currently titled "garages", are the spawnpoints for cars.

The long red rectangle in the distance is an obstacle object I was testing with blocking the road and seeing where the car goes, which was how I found out that the car will keep going down a road regardless of obstacles.