Final Project of CS260: Motion Planning

Procedural City Generation with Integrated Motion Planning and Traffic System

Team: Anirudh Nittur Venkatesh (anitt003), Ashik Mysore Padmanabha (amyso001), Rakshith Mahishi(rmahi001)

1. Introduction

This project deals with real-time planning and simulation of a city with pedestrians, cars, roads and multiple buildings across the city. For urban planning, it is crucial to develop and simulate realistic simulations for testing and validating the integration of pedestrians and vehicles in a city landscape. We build a city simulation that covers all the necessary details of a city and also implements effective use of motion planning algorithms to guide the entities in the simulation. To achieve this we have used Unity along with C# programming for implementing the cityscape procedurally and also for path-finding. Blender 3D was used for generating modular 3D assets needed for the simulation of buildings, cars and pedestrians.

2. Implementation

In this section, we give a detailed explanation of the implementation techniques and the process of city generation.

2.1 Procedural City Generation

In the first part of the implementation, we start by creating the city and all of its entities. Unity and Blender were used for generating the city layout and once the basic road layout was established we started by adding all the other entities like buildings and parks.

Unity and Blender: We used C# scripts to implement the logic and interactions. Then we make use of Blender to create a modular 3D design such as buildings for residential, commercial and industrial which are then imported to Unity where the cityscape is generated.

City Layout Generation: First the road network is generated using a simple crawler algorithm. A crawler moves through the grid, laying down road pieces and making random turns to create a varied layout. After the initial road network is created, it is optimized to replace overlapping roads with appropriate junctions (T-junctions or crossroads) to create a more realistic road layout.

Zoning and Building Placement: Buildings are placed based on zone type and density, determined using Voronoi diagrams and Perlin noise.
Voronoi Diagrams: The city grid is divided into zones (residential, commercial, industrial) using Voronoi diagrams, with each zone having a designated type of building.
Perlin Noise: Used to vary the density and type of buildings within each zone. For example, areas with higher noise values might have larger buildings.

2.2 Motion Planning Integration

We apply some motion planning algorithms for the simulation of the vehicles and pedestrians that move across the city.
Pathfinding Algorithms: We implement an A* algorithm for both vehicles and pedestrians to find the shortest path between two points in the city. This algorithm ensures that objects reach the point and avoid obstacles during their course.
Collision Avoidance: Unity's physics engine is used to detect collisions between entities. We wrote a script that was developed to enable vehicles and pedestrians to react dynamically to their surroundings, adjusting their paths to avoid collisions while maintaining realistic movement patterns.
Waypoint Graphs: The city is built upon a huge graph network for both roads and pedestrians. With this graph and the A* algorithm entities can navigate through the city and also avoid any form of collision until they reach their destination.

2.3 Realism and Optimization

We used a game engine like Unity and an industry-standard 3D software like Blender for generating high-quality 3D assets to ensure the simulation was realistic.
Graphics and Visual Effects: Using high-quality textures, lighting effects, and shaders in Unity enhances the visual realism of the simulation. The modular assets created in Blender contribute to the detailed and varied appearance of the cityscape.

3. Results

After implementing all the above things in the project for urban planning and simulation, we observe that both the city generation and the motion planning algorithms work together to bring out a beautiful city with buildings, vehicles and pedestrians. A road network is created for the vehicles to move around in a said manner without colliding with each other. A condition is also taken into consideration that when pedestrians cross the road all the vehicles on the four sides wait for the cross and all the vehicles start moving one by one. The integration of collision avoidance protocols ensured smooth and safe movement, avoiding congestion and conflicts. Overall, the results from our simulation show the potential of combining procedural city generation using Blender and Unity with motion planning algorithms to create realistic and functional urban environments for autonomous navigation of both pedestrians and vehicles.

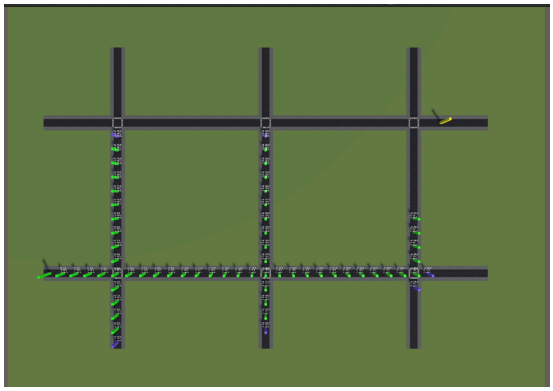


Figure (a)

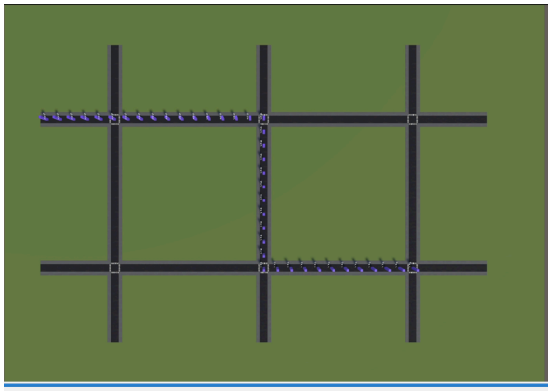


Figure (b)

Figure (a) represents the A path solving & Figure (b) represents the Bactracked/final path of the A* algorithm*

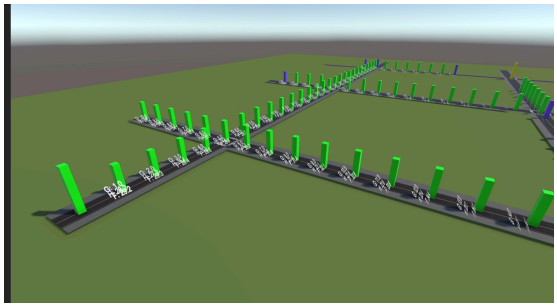
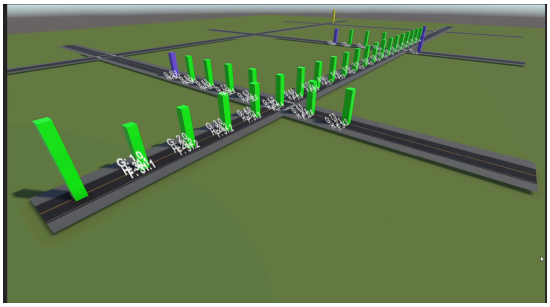


Figure (c)



Figure(d)

Figure (c) and (d) represents the top and side view of the path markers

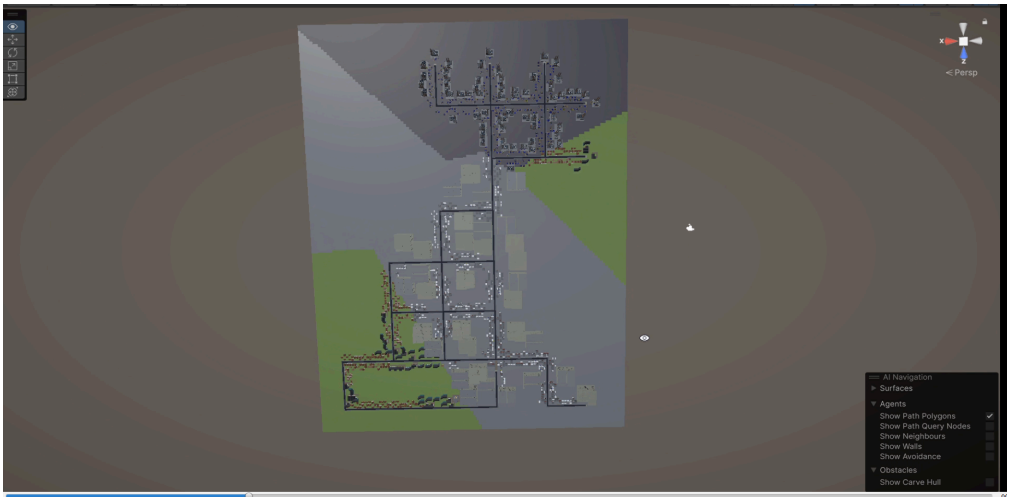


Figure (e)

The figure (e) is the procedural city generated in order to showcase the Voronoi diagram and Perlin noise implemented.

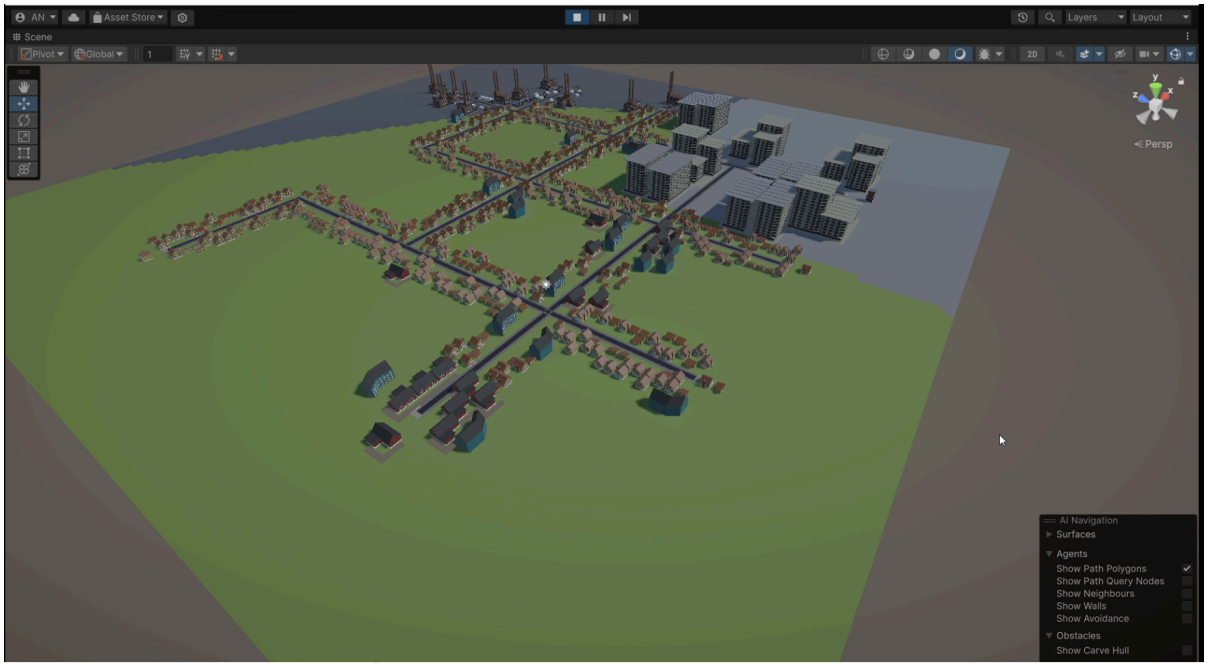


Figure (f)

Figure (f) showcases the three zones that are residential, commercial and industrial along with the final road layout

Figure (g)

Figure (g) showcases the three zones that are residential, commercial and industrial along with the final road layout in a different view.

Figure (h)

Figure (h) represents the traffic system at an intersection showing both cars and pedestrians

Figure (i)

Figure (i) represents the traffic system at an intersection showing both cars and pedestrians from the top view.

4. Challenges and Limitations

Even with all the implementations, there are certain limitations of the project that we list here.

- **Computational Complexity:** Simulation of autonomous navigation can be computationally intensive, requiring significant processing power and memory.
- **Pathfinding Efficiency:** A* algorithm is effective for pathfinding, it may not always be the most efficient choice for large-scale simulations with numerous entities. Optimizing pathfinding algorithms for better performance and scalability is a challenge.
- **Validation:** Validating the accuracy and reliability of the simulation's outcomes against real-world scenarios is essential but can be challenging due to the inherent differences between simulated environments and actual urban settings.

5. Conclusion and Future Work

In conclusion, we successfully implemented a city with entities like buildings, cars and pedestrians with the help of software tools like Unity and Blender and also made use of motion planning algorithms to achieve autonomous navigation of cars and pedestrians with collision avoidance. The use of the A* algorithm and collision avoidance protocols ensured efficient and realistic movement within the simulation. Even though there are limitations to the simulation the overall results and our approach yielded good simulation.

Future Work:

Traffic Light Integration: Incorporating the concepts of traffic lights to ensure proper simulation of the environment would be a great step towards improving the simulation.
Multi-Agent Systems: A more complex multi-agent system can be set up to show a more realistic simulation of an urban city with more entities and communication between vehicles.

6. References

[1] Karamouzas, I., Skinner, B., & Guy, S. J. (2014). Universal power law governing pedestrian interactions. Physical review letters, 113(23), 238701.

[2] https://dl.acm.org/doi/10.1145/383259.383292

[3] https://gamma.cs.unc.edu/HYBRID_TRAFFIC/

[4] https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf