Robert Grier

10/28/2024

Simple Traffic Sim V3

Overview

The game allows the player to view the world from a flying camera. The player can switch between tools for viewing, spawning buildings, spawning roads, and erasing. Each tool has a gizmo that appears when in that mode which follows the mouse. When left clicking, the tools may spawn or remove objects from the world. Objects are spawned onto a 2D grid overlayed on the floor plane. Objects can only be spawned if the grid is not occupied at that location. There is a user interface implemented with “egui” that provides buttons, controls, and stats about the game.

As roads and buildings are constructed and removed, a graph network is updated to reflect the state of the world. This graph network is queried by vehicles to calculate a path from one building to another. Vehicles follow their path by moving across road segments and turning at intersections. The game can be saved and the save state is automatically loaded on start.

Camera

Rotate – (Middle Mouse) or [Q / E] or [Left Control] + (Left Mouse)

Panning – (Right Mouse) or [WASD] or [Left Alt] + (Left Mouse)

Zoom – (Scroll Wheel)

Grid

Toggle Visualization – [G]

The grid is represented as a 2d array of Option<Entity>. Queries to the grid take the form Result<Option<Entity>> where the result will be an error if the request is out of bounds or not valid. The option is none when the grid is empty and contains some entity when occupied at that location. I am using a crate to draw an infinite grid for debugging. There are helper classes for GridCell and GridArea to represent individual cells and areas of cells on the grid.

User Tools

Switch to view tool – [`]

Switch to building tool – [1]

Switch to road tool – [2]

Change road orientation – [Tab]

Switch to eraser tool – [3]

Adjust tool size – [F/R]

Tools are swapped between using a state machine in toolbar.rs. The building tool spawns scaled cubes to represent the buildings. The road tool allows you to drag after clicking to determine the size of the road. The road tool automatically creates intersections, extends roads, and bridges roads

based on adjacent data in the grid. The eraser tool deletes buildings and roads. The view tool just disables all other functionalities.

Graph

Toggle Visualization – [H]

The graph connects buildings, roads, and intersections so that the city can be navigated. The graph observes spawning and deleting events to keep everything up to date. The graph is visualized with gizmos to show all the connections.

Vehicles

Spawn – [P]

Spawn Many – Hold [L]

When spawned, vehicles calculate a path from one random building to another if a path is possible. They query the graph to do a simple DFS for the path. If the path is edited after spawning the vehicle, it will just stop. Events should be added to recalculate the path if the relevant graph components are changed. Vehicles drive freely and follow the desired lane on the current road. The desired lane is based on the upcoming turn direction in their search path. Vehicles must turn manually at intersections to end up in the correct direction and lane. In addition, vehicles use the "bevy_mod_raycast" crate to detect each other with a raycast. This allows them to slow down and stop when they are close to each other, preventing a collision. This behavior causes traffic to build up when the roads become congested.

User Interface

The game integrates the "egui" system using the "bevy_egui" crate. I used this to add a very basic user interface for switching between tools, describing controls, and showing stats about the game state. Although the immediate mode GUI is not great for a game application, it is a good start for getting some basic info on screen and can be used for development tools. Additionally, the bevy engine's native GUI system is not particularly powerful or concise, so I am avoiding it.

Saving

Save the Game – [F5]

The game state can be saved to a file using the Serde Json crate. This generates a Json file containing all the data needed to recreate the city built by the player. When the game starts, the city data is loaded from the file and updates the world. The graph and grid components are idempotent, which allows the loaded items to be re-spawned easily. Currently, only one save file can be used.

How to play the demo

Start the game and view the saved city loaded from the save file. This city can be extended by the player. Place some roads using the road tool. Intersections will be generated on adjacent roads (in most cases). Place some buildings adjacent to the roads using the building tool. Erase buildings and roads as needed. Observe the graph update by toggling the visualization with [H]. Spawn a vehicle with [P] and observe that it generates and follows a path from one building to another. Press [V] to view the path and debug the ai behavior. Vehicles will begin spawning automatically. Observe that

vehicles detect each other and slow down to prevent collisions. The vehicles should also change lanes based on what turn they need to make. Press [F5] to save any changes to the city structure.