3. Simulation

For simulation, Anylogic Personal Learning Edition (v8.7.2) was used. The libraries used throughout the modelling include:

- Road Traffic Library: For road traffic components such as roads, vehicles. The core library for this project.
- Process Modelling Library: For logic and entities such as agents.
- Pedestrian Library: For simulating pedestrian crossing.
- Presentation Library: For 3D simulation and other presentation entities.
- Space Markup Library: For marking zones for pedestrians to cross inside pedestrian crossings.
- Analysis Library: For analysis and charts to measure data during the simulation.

3.1 Design Approach

The layout of the model was created based on Google maps of the node in the same level of scaling. Extra lanes were added to act as Tram lines since there is no explicit Tram/street car library in Anylogic.

Key Components:

The key components in the simulation include the agents and entities that control the core flow of the traffic.

- Vehicles
- Pedestrians
- Tram
- Bus
- Traffic Signal
- Intersection (Including allowed turns to different roads)

3.2 Program Structure

Two main segments of the Anylogic simulation models were extensively used in this project.

- Agent: These are the core components that make the model run
    - Main: This is the main agent where the design of the node takes place.
    - Cars/Busses/Trams: These are sub agents which act their part inside the main agent.

- Experiment: This segment is to create experiments and test different scenarios.
    - Design experiment: These were the actual experiments performed. Discussed further in the Experiments section.
    - Traffic Variance: These are experiments where we increased the traffic load ranging from 1(current load) to 5(500% of the current load) and analysed the outcomes.

3.3 Program Concept & Model

The simulation program was designed by modelling the roads and the intersections of the node along with implementing the expected behavior of the agents and other elements.

Roads and Intersection: The roads are created using the Road markup element from the Road Traffic Library. Intersections were automatically formed when two or more roads were connected. They could also be manually inserted from the same library.

Tram Lines: Since Anylogic does not have tram lines, extra lanes in the roads were created to act as Tram lines and custom agents of length 20 (for comparison, car's length is 5) were created and instructed to travel only on those specified lanes.

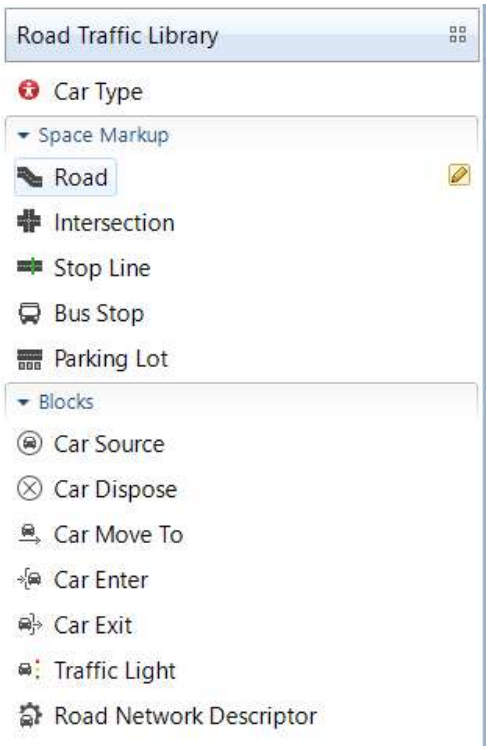

Figure 24: Road Traffic Library of Anylogic.

Traffic Light: Taken from the same Road Traffic Library, the signal was instructed to manage traffic flow of all the roads coming into the intersection (except the half arm of Gerhart-Hauptmann Straße joining into Olvenstedter Straße south side).

Pedestrians: These agents were simulated by using the Pedestrian Library for creation of pedestrians and Space Markup library for marking the area for those pedestrians to travel.

The model was designed to run both on 2D as well as 3D view.

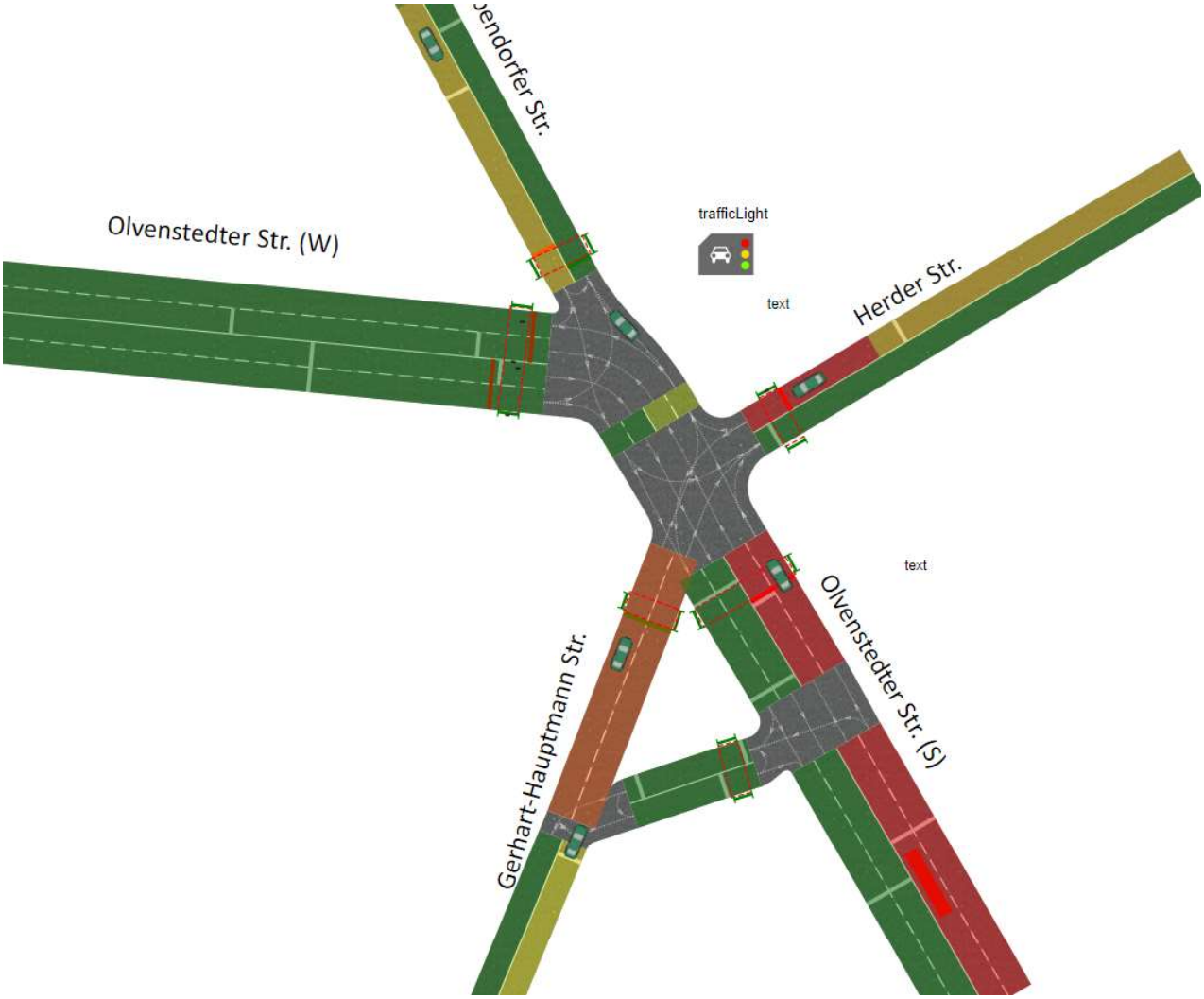

Figure 25: Simulation model 2D version

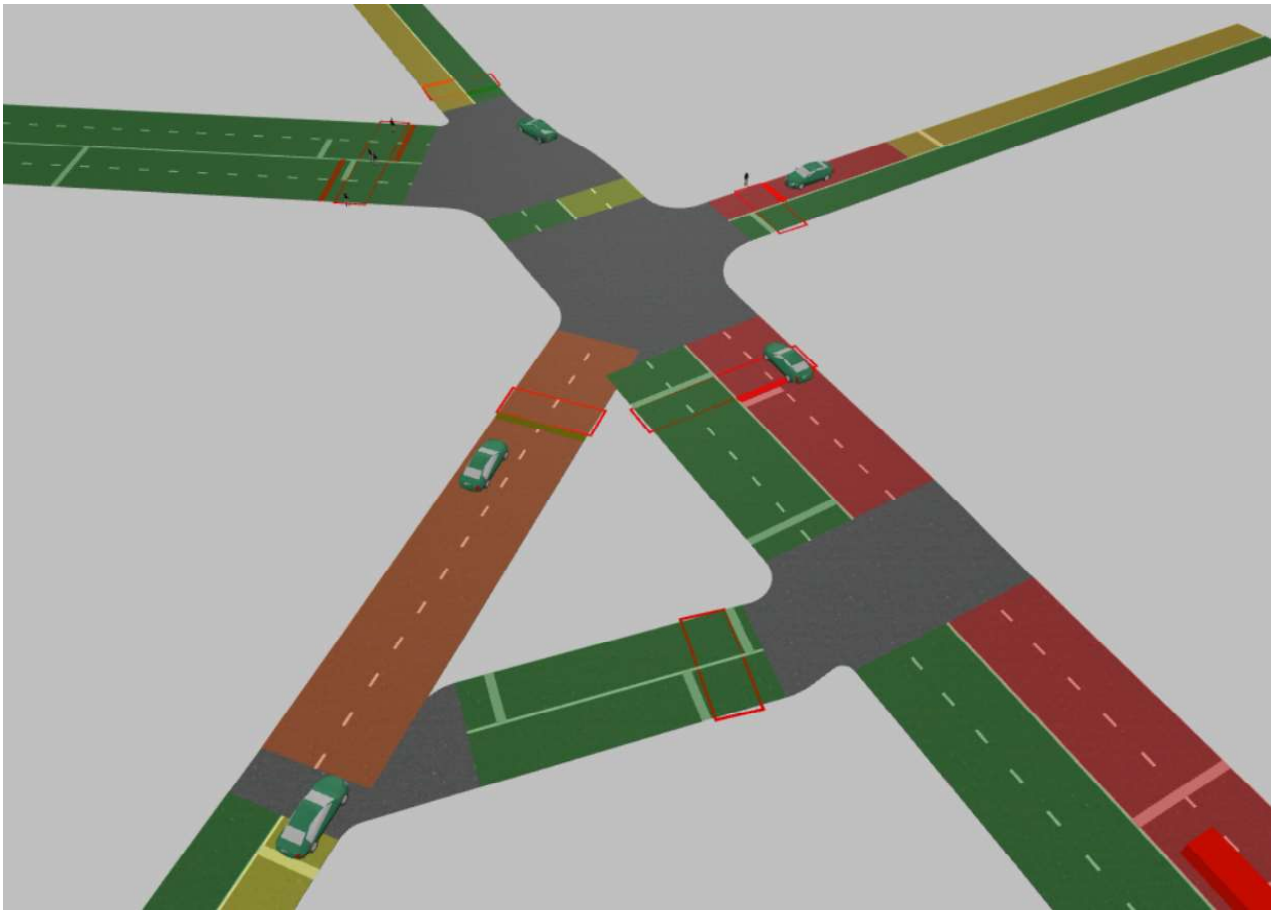

Figure 26: Simulation model 3D version

3.4 Simulation Logic:

The simulation ran on a few predefined logics and rules that the model and the agents followed in order to imitate real world situations.

Stop Line markup:

This is a line markup that is placed on the road in order to control the flow of vehicles. We have used the stop lines to:

- Control traffic when linked with the signal, acting as the boundary of the roads.
- Control incoming cars whenever a pedestrian is still crossing while the signal changes.
- Direct cars and trams in Olvenstedter Straße to stay in their lane.
- Create virtual Tram/Bus stops when combined with a delay block.

SelectOutput5 blocks:

This block takes upto 5 possible probabilities and distributes the output based on them. The probabilities of vehicles going into different destination roads from any source road are given as input. The inbuilt functionality instructs the vehicles to move to those roads based on the given probabilities.

Rectangular node:

This is a space markup entity which can be used to contain agents. In this case, these are used as pedestrian crossing zones and the pedestrians are instructed to stay inside them. The access to these zones are blocked if the pedestrian signal is Red.

3.5 Agent Behavior:

Based on what was expected of each agent, the applied behavior differs.

Cars: Once cars are created, they are forwarded to their destination roads by using the carMoveTo block. Each destination road has a carMoveTo block that will be selected through the SelectOutput5 element based on turn probabilities that have been entered.

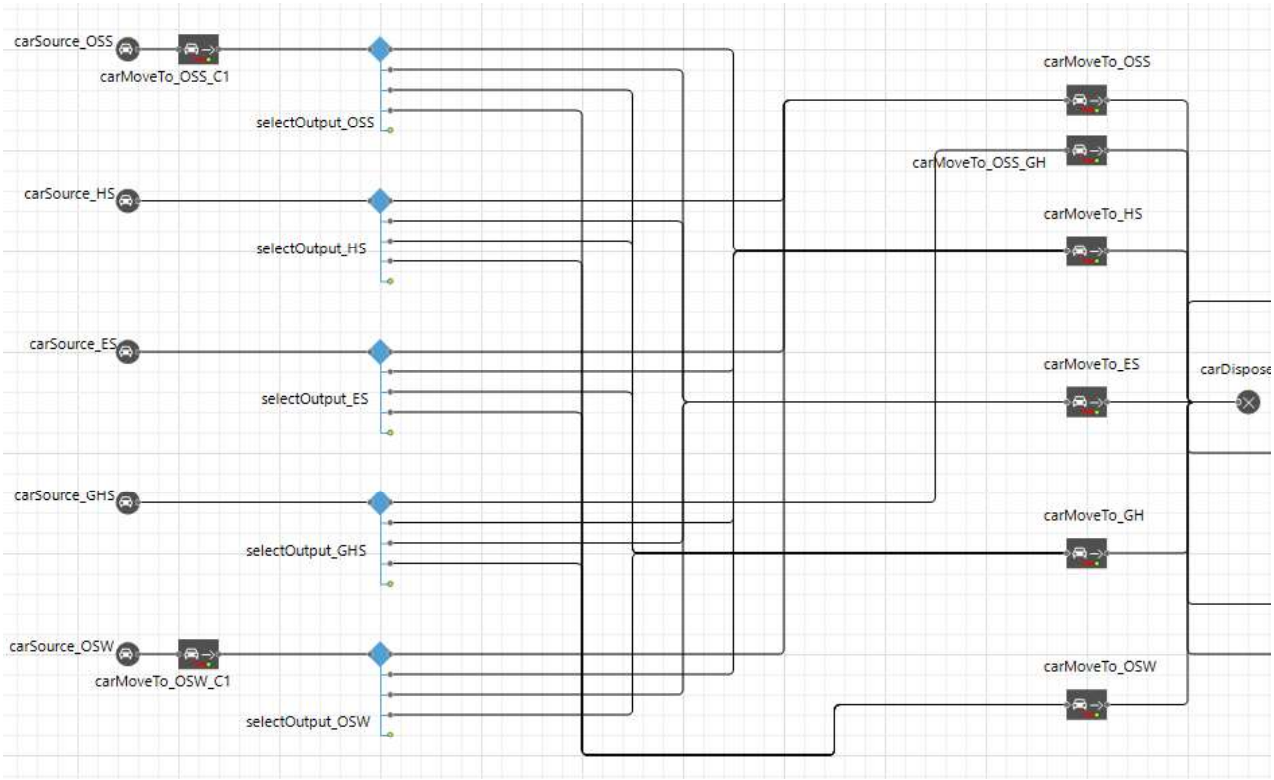

Figure 27: Car agent life cycle and behavior

Bus/Trams: These agents behave similar to Cars but they have one extra step in their life cycle. To imitate stops made by the real world public transports, an instruction was added to stop near their destination stopLine (acting stations), wait for a set amount of time (delay blocks) and then move towards dispose.

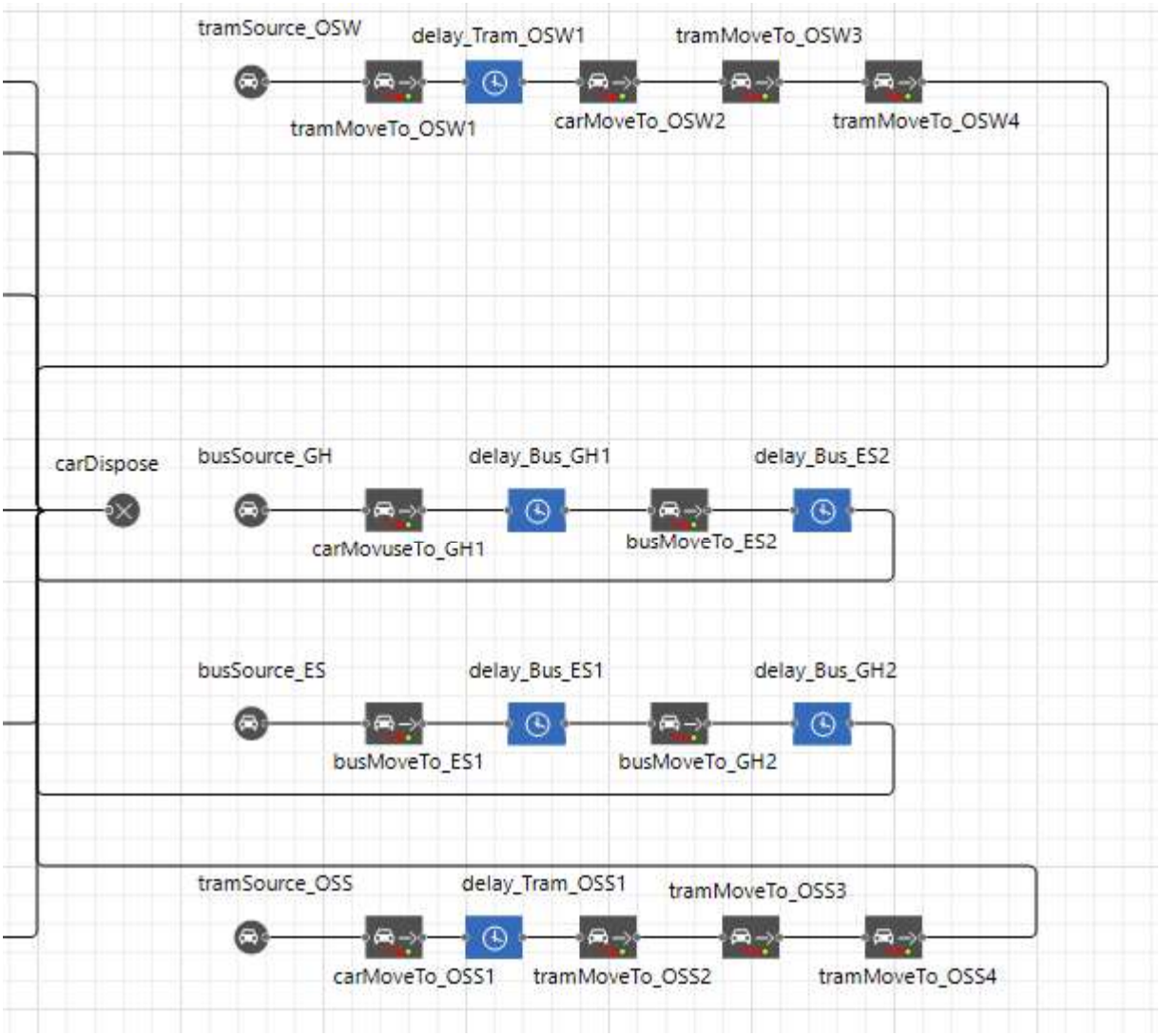

Figure 28: Public Transport - Bus/Tram agent life cycle and behavior

Pedestrians: These agents were modelled to get created at one end of a Pedestrian crossing, wait for the signal to allow the cross and then 'sink' (or get destroyed) when they reach the other side of the road/crossing.

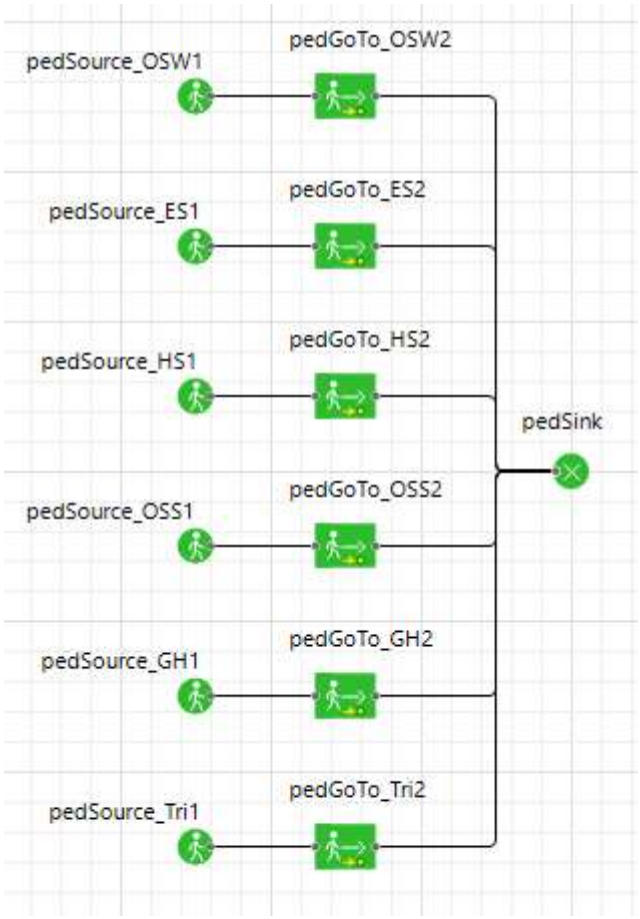

Figure 29: Pedestrian agent life cycle and behavior

Calculation of variables:

There are two main variables being calculated: Average queue length and average time in node.

Average Queue Length is calculated after every signal phase change. Whenever the signal of a particular road turns Green, all the vehicles standing in that road (previously waiting in Red signal) would be recorded. This would then be converted into an average after all the signal phase changes during the complete simulation.

Average time in node is calculated for each car. A custom parameter startTime of type time() is attached to this agent. Whenever a car gets created, the parameter records the startTime. While the car goes through its journey and moves into 'dispose',i.e., its destination finish line, the current time is recorded and startTime is subtracted from it. This gives the time the car spent inside the node or our model. Then the mean is calculated towards the end by reading the time spent by each car.

3.6 Testing:

The testing of the program was done in two different aspects: To check whether the system ran without errors and to check if the system behaved like the real world when data recorded from the real world were given as input.

Simulation Test:

This part consisted testing of the system against errors such as:

- Individual blocks such as Vehicles, Pedestrians, Buses, Trams and their life cycles.
- Regression test with varying parameters
- Collision between vehicles and pedestrians

Comparison with real world:

This part consisted of confirming if the system acted as expected when fed with real world data.

- Vehicle counts in each of the roads
- Vehicle routes based on probabilities
- Signal phases (Approximation)