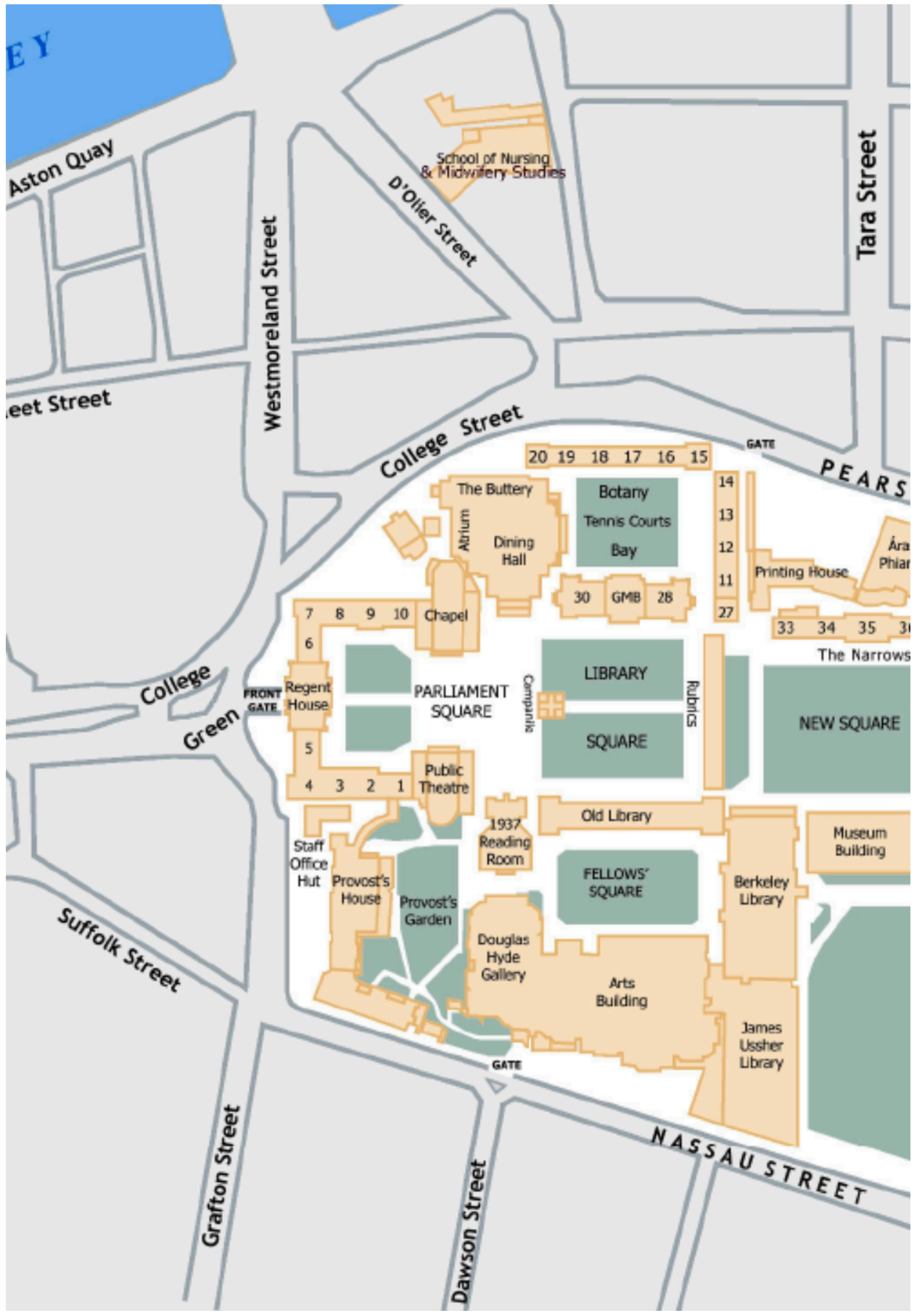
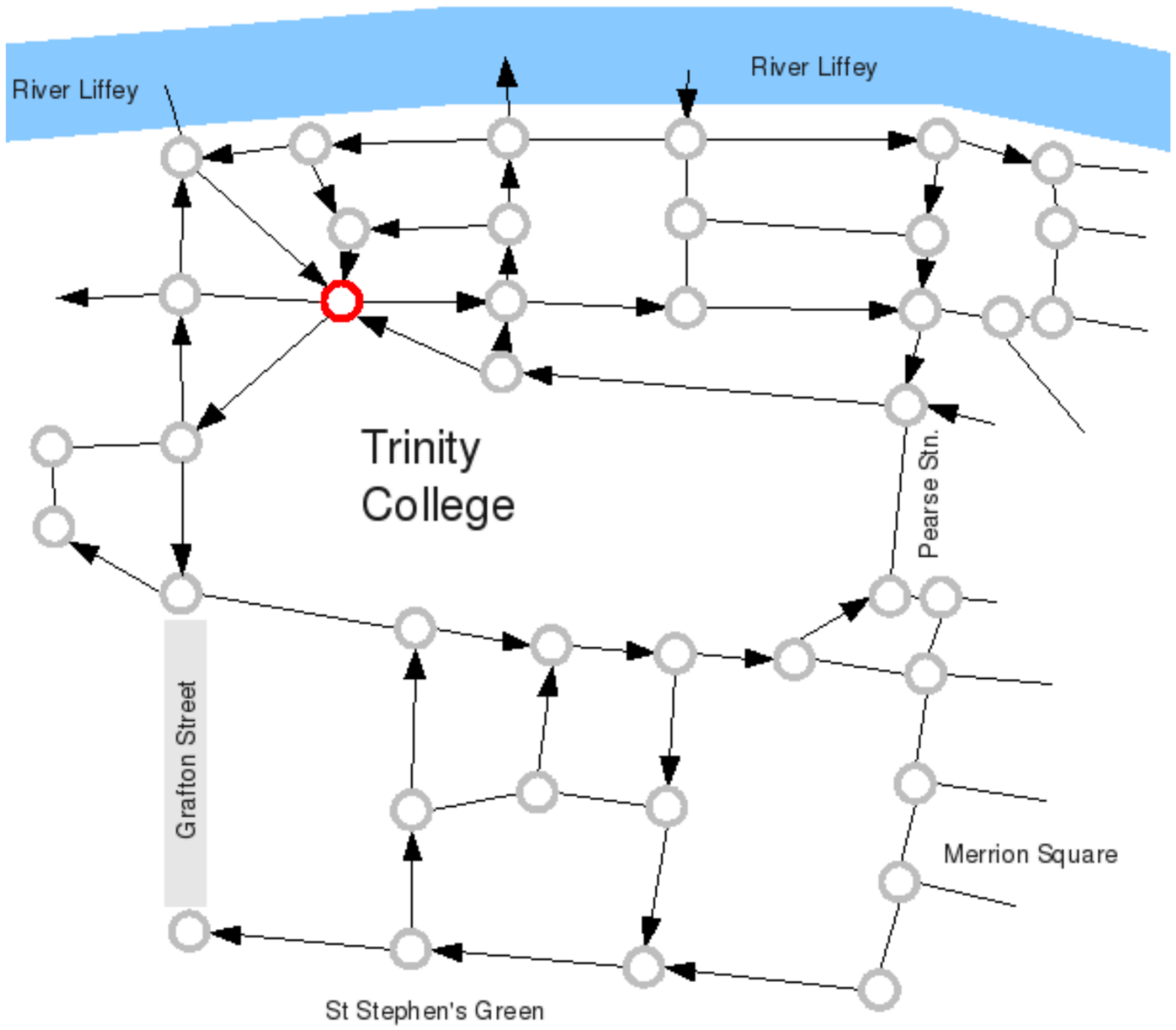
Graphs - Example with Evolutionary Algorithms

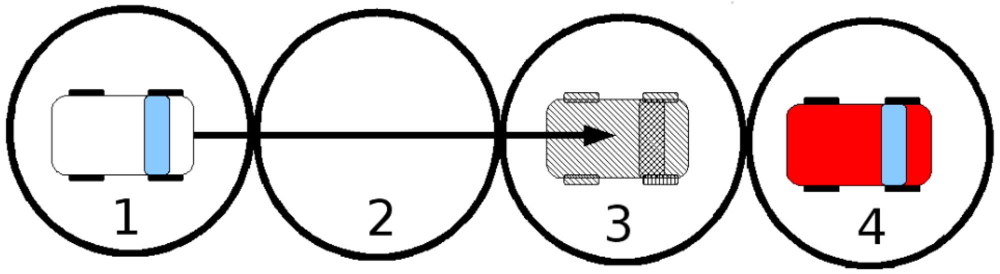
Anton Gerdelan <gerdela@scss.tcd.ie>

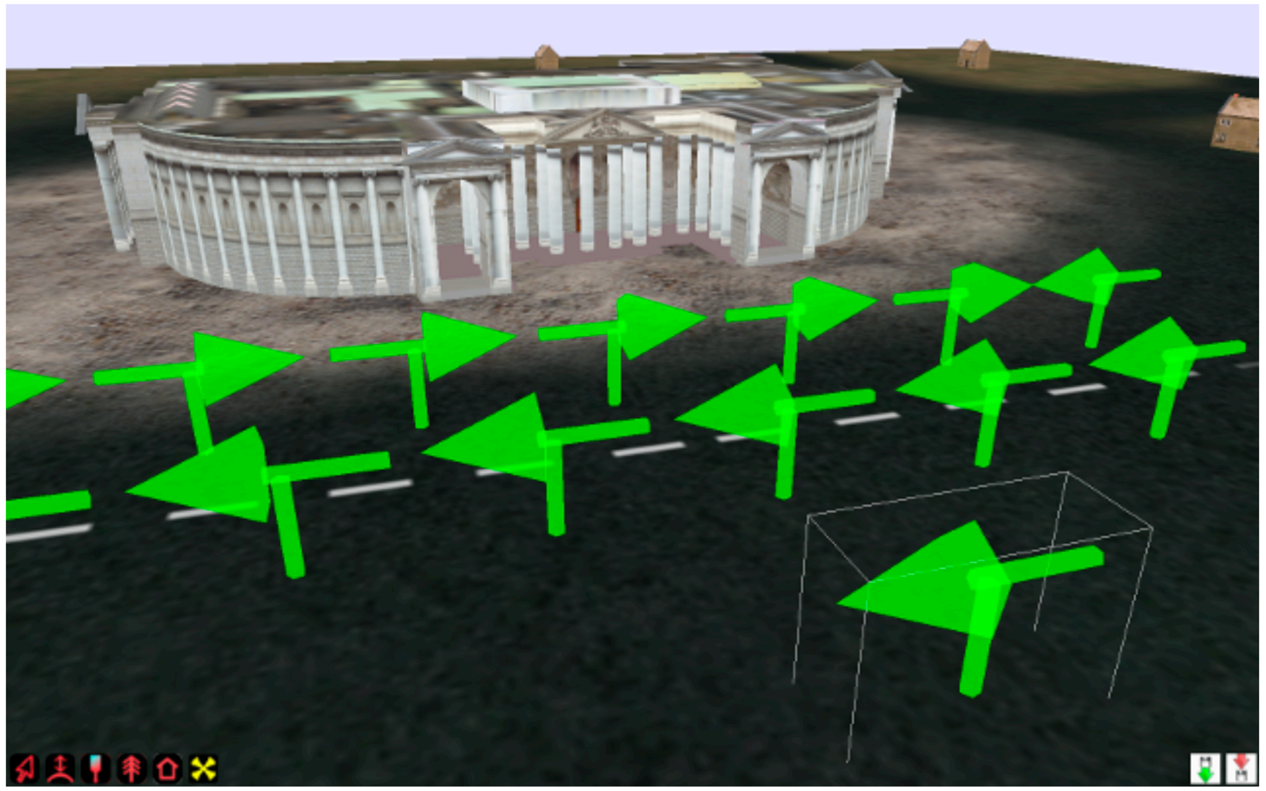
Graphs in *Metropolis*

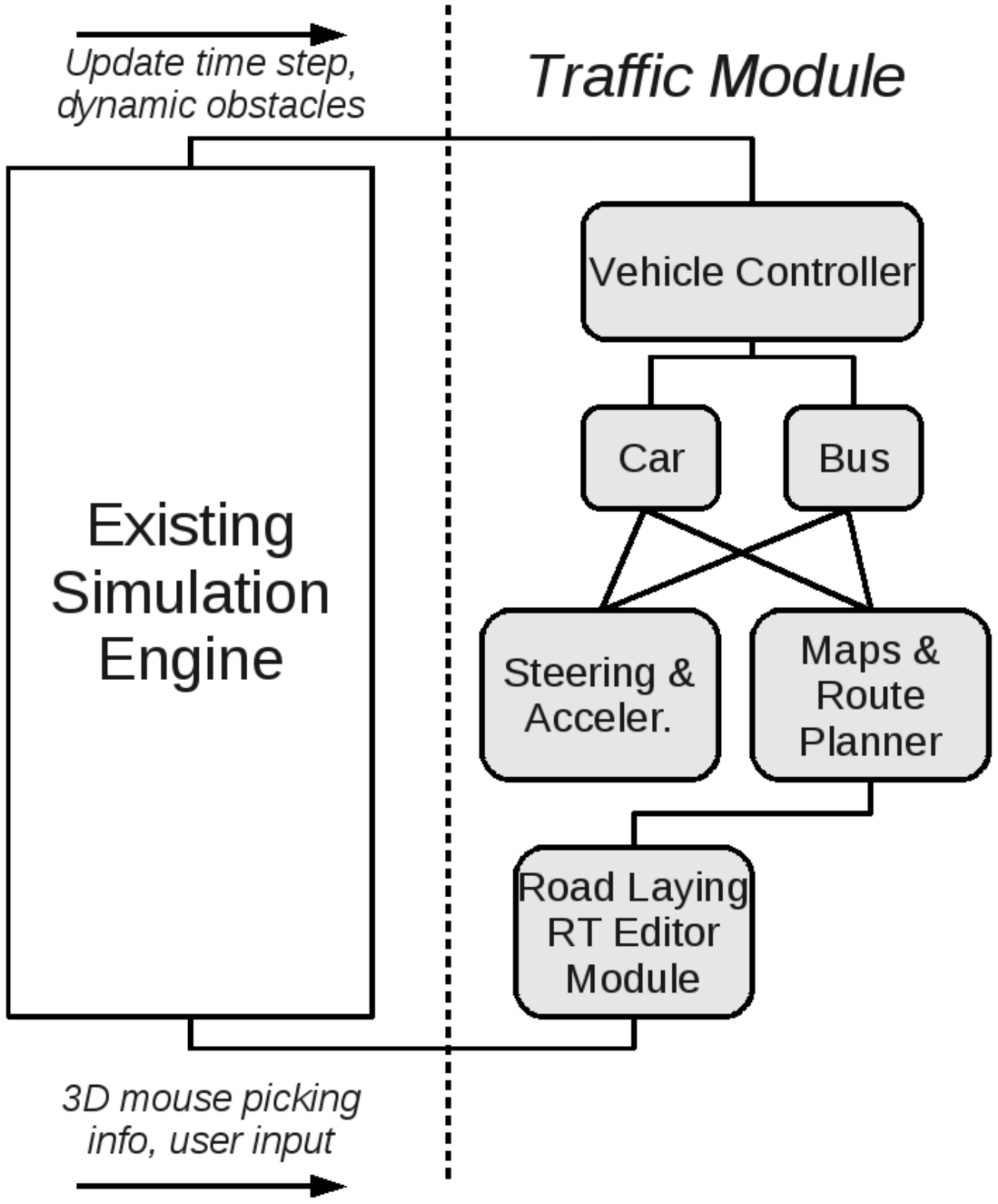




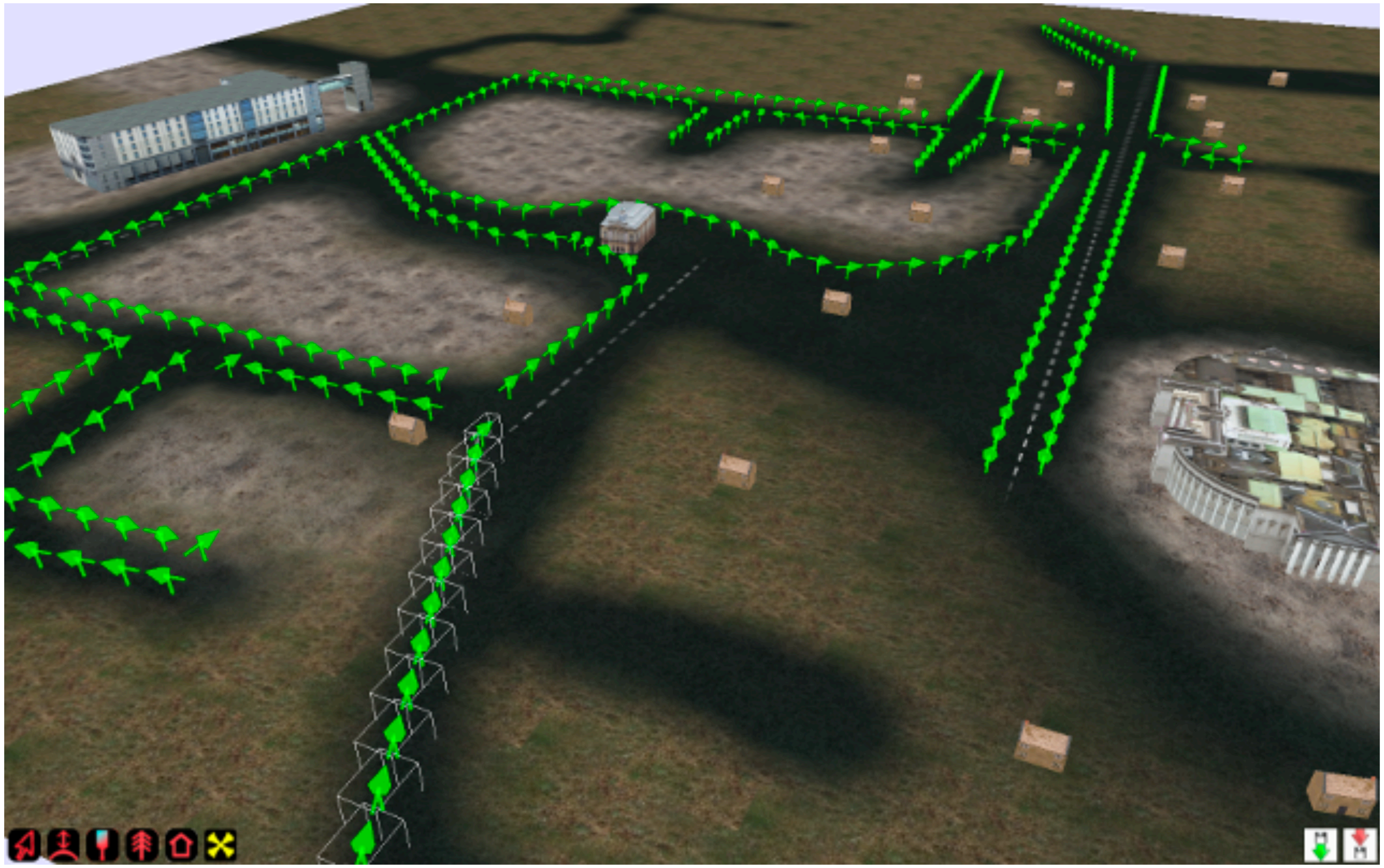
Edges are lanes






Update time step, dynamic obstacles
Traffic Module
Existing Simulation Engine
Vehicle Controller
Car
Bus
Steering & Acceler.
Maps & Route Planner
Road Laying RT Editor Module
3D mouse picking info, user input

3d editor to build network



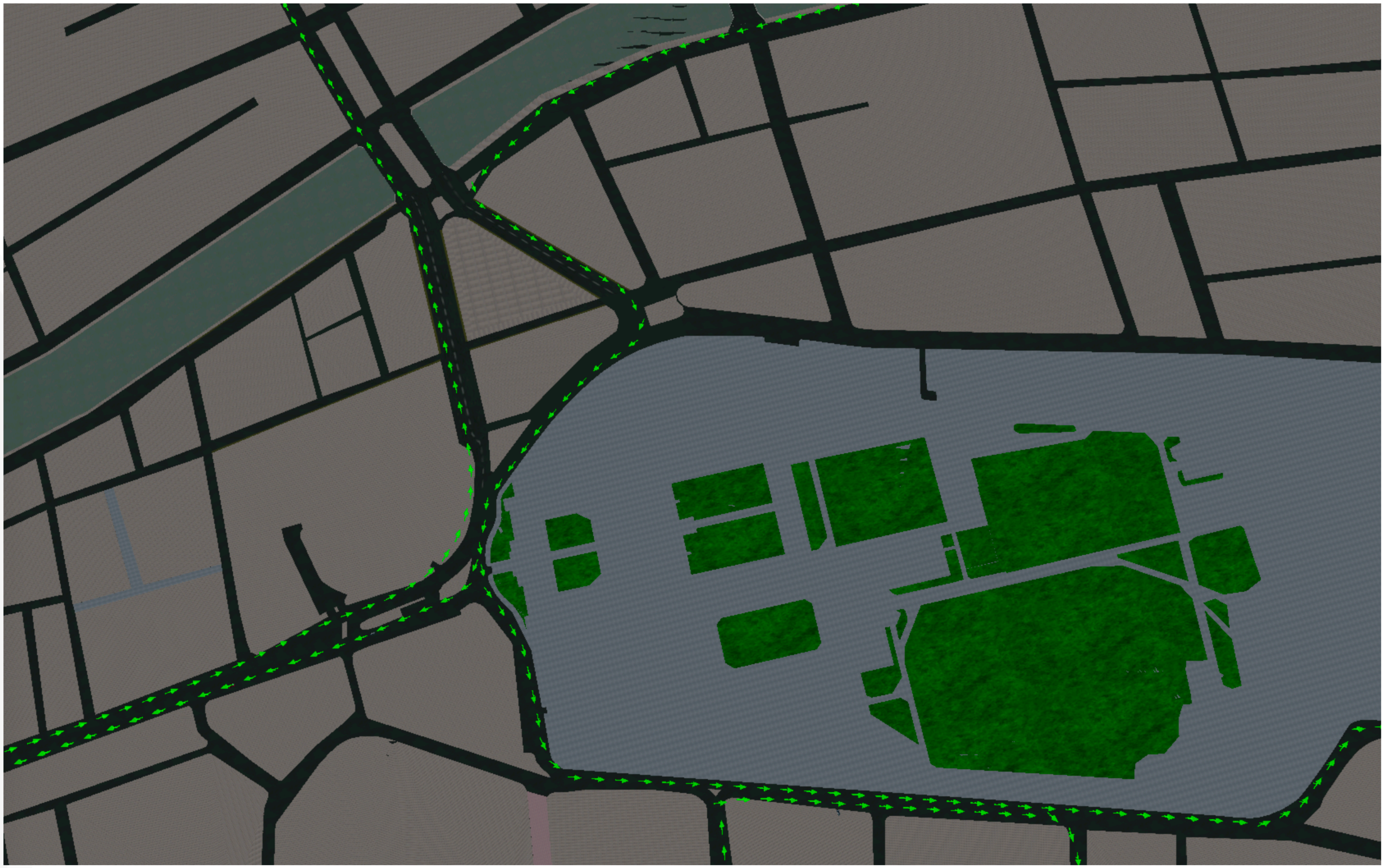


Figure 7.8: This figure shows us a birds-eye view of the original street area from Figure 7.1. A subset of the road lanes taken from the map in Figures 7.2 and 7.3 have been recreated, with the one-way directionality of lanes preserved. We can see that some of the nodes will need to be slightly manually adjusted to follow the modelled roads more closely as many of the points have been automatically interpolated and have not quite matched the model in this case.

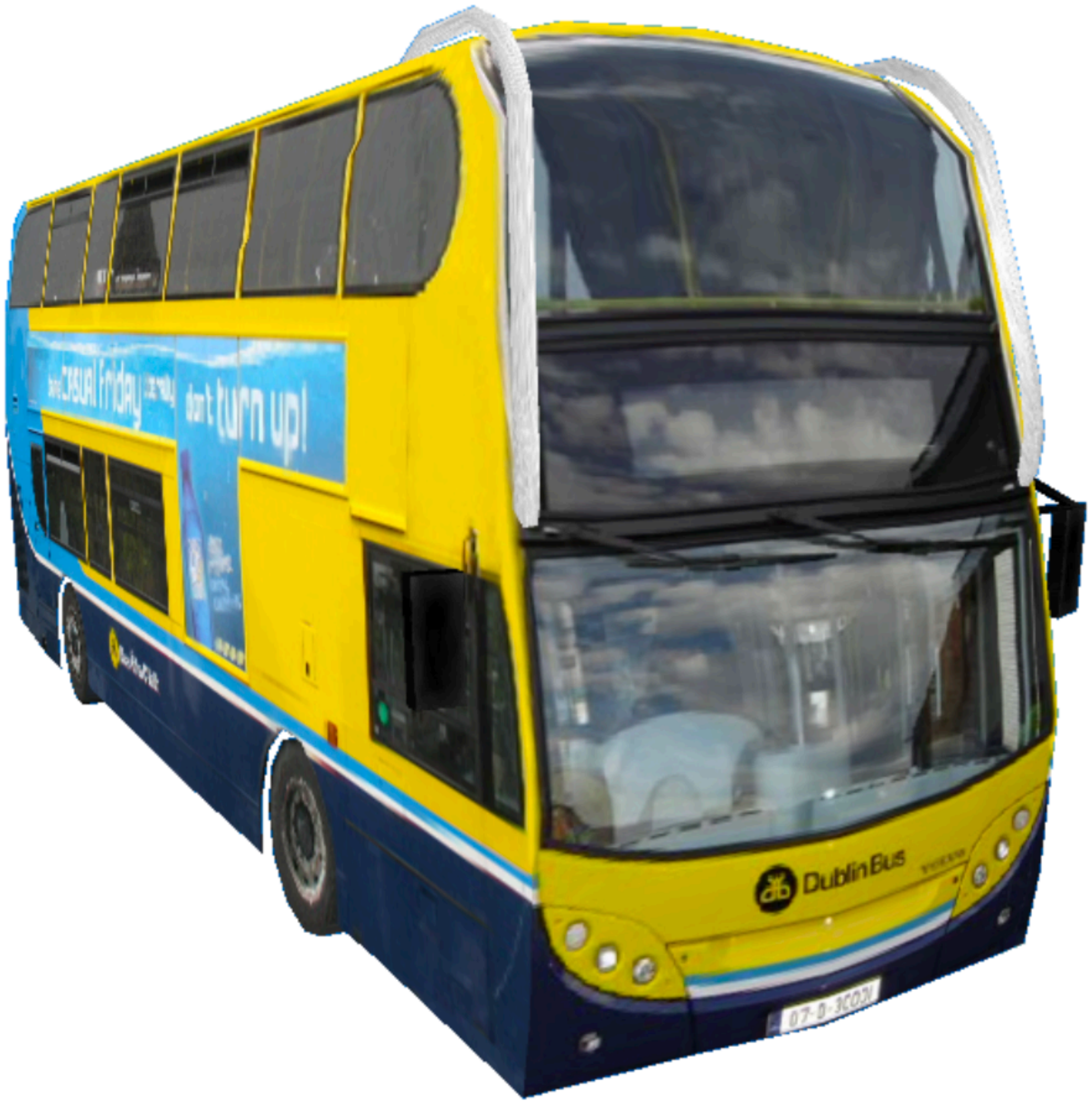


Figure 8.2: A test vehicle, created for the traffic simulation - an Enviro 400 Dublin Bus.

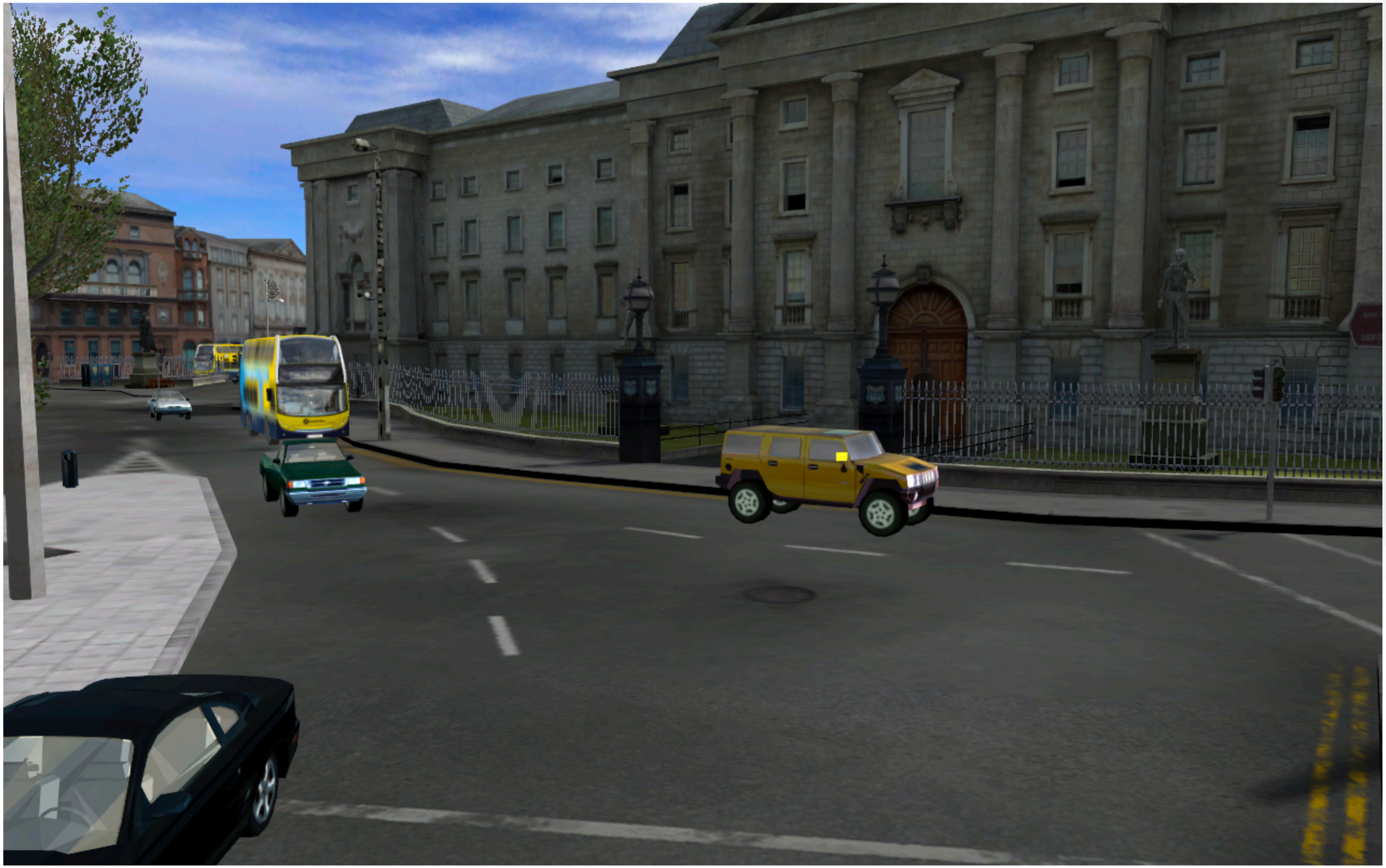


Figure 7.9: Simulated traffic automatically navigate the previous empty streets of the Dublin model.

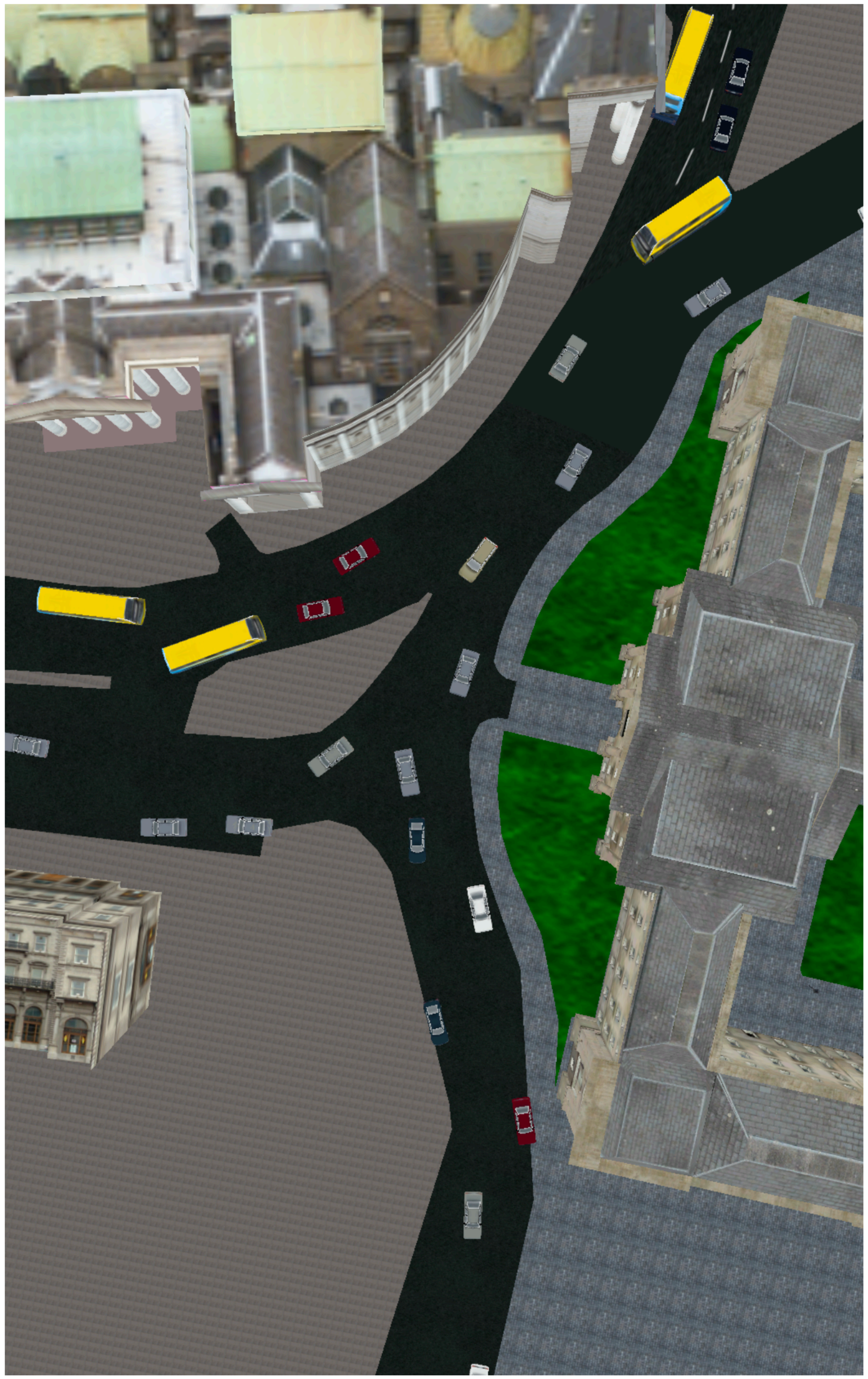


Figure 8.11: Prototype system in operation: real-time congested traffic simulation for College Street, Dublin City.

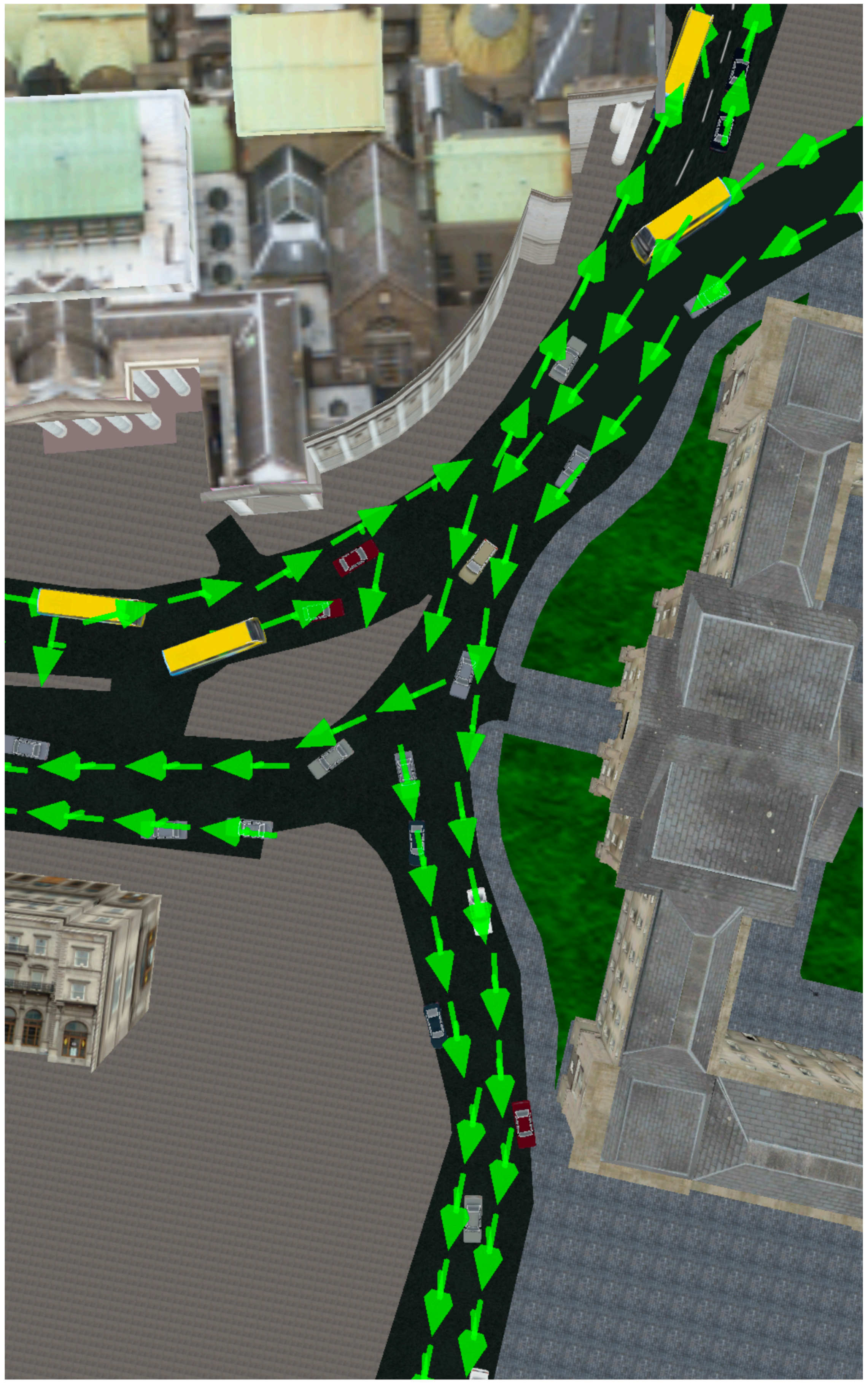


Figure 8.12: Prototype system in operation: lane demarcations for College Street, Dublin City.

“Use a physics engine to make animation look more realistic.”

–The Boss

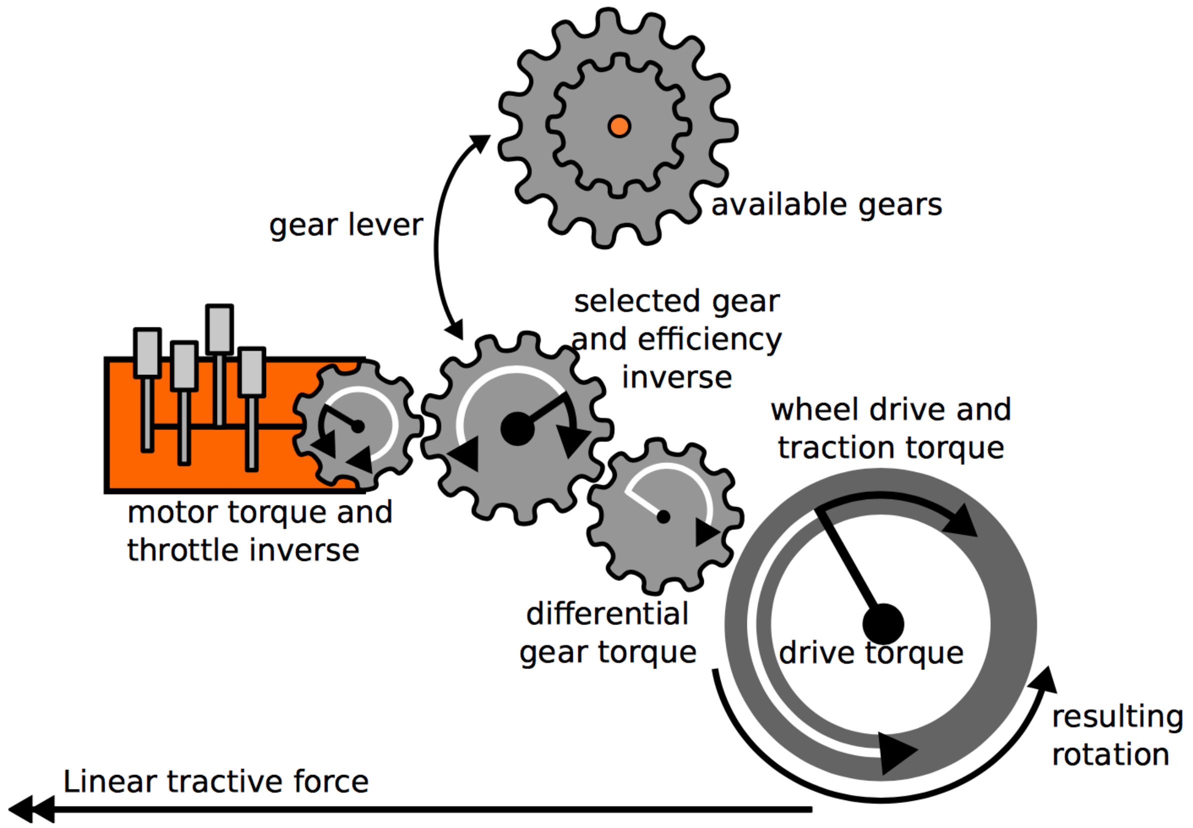

available gears
gear lever
selected gear and efficiency inverse
wheel drive and traction torque
motor torque and throttle inverse
differential gear torque
drive torque
resulting rotation
Linear tractive force

Figure 11.7: The complete drive-train assembly used with all torque-affecting forces and their opposing forces shown.

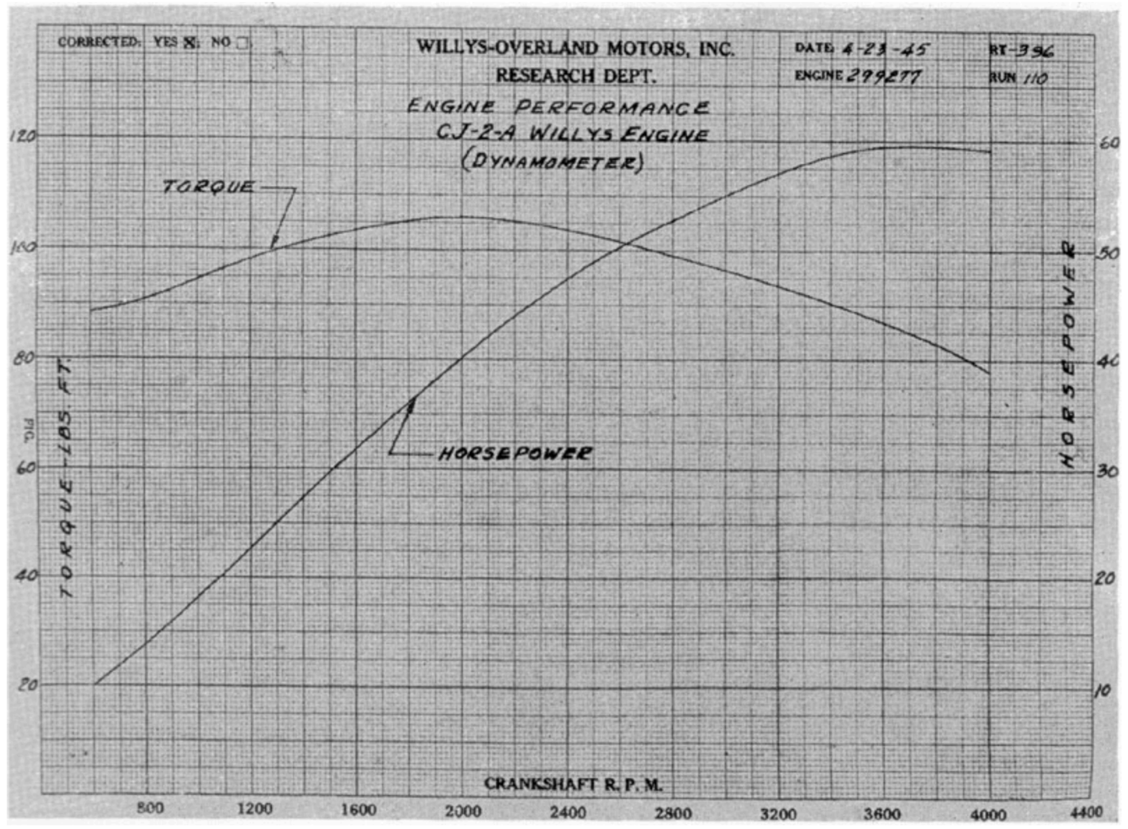


Figure 11.8: The RPM-torque curve for the Willys MB Jeep. The second plot is an RPM-power curve for the same engine. Source: Willys 1945 CJ2A Maintenance Manual.

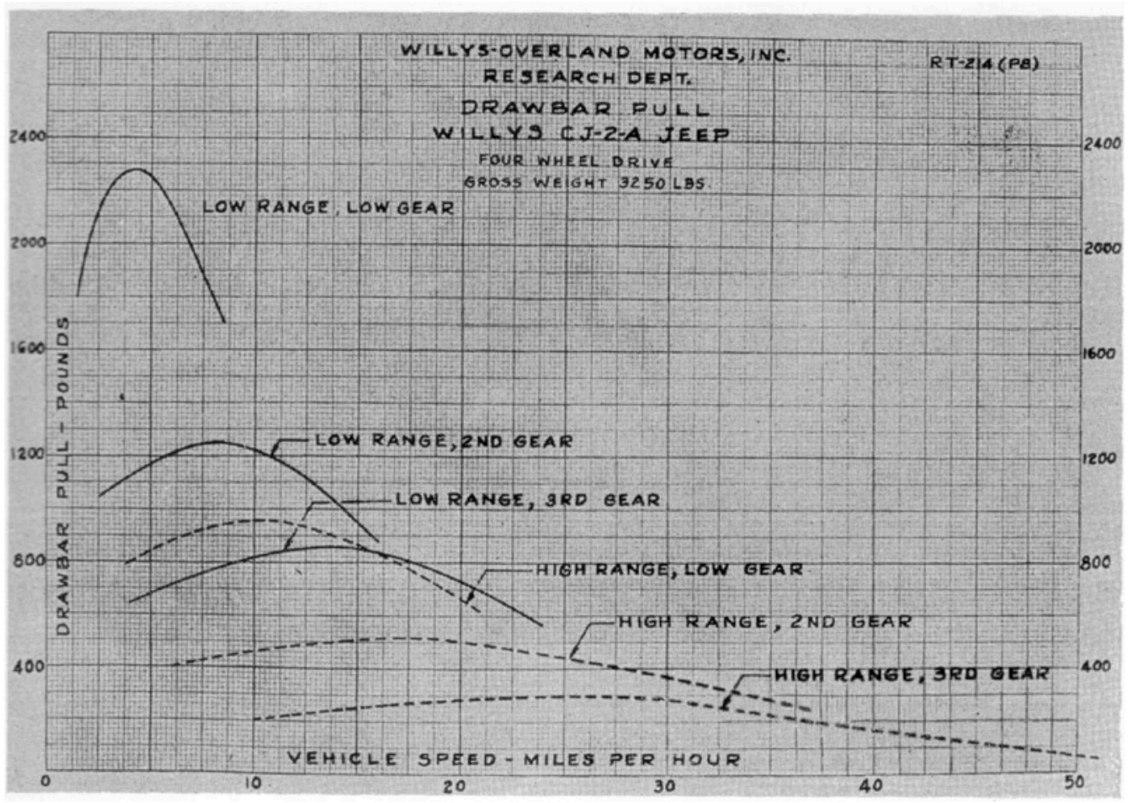


Figure 11.9: The RPM at final drive (proportional to vehicle speed) for each gear of the Willys MB Jeep.

$$\tau_{drive} = \tau_a + (f - f_a)\frac{\tau_b - \tau_a}{f_b - f_a} \cdot x_g \cdot x_d \cdot e$$

$$F_d = \frac{1}{2}\rho \cdot v^2 \cdot C_d \cdot A \qquad C_{drag} = \frac{1}{2} \cdot \rho \cdot C_d \cdot A$$

$$F_{rr} = C_{rr} \cdot v \qquad \tau_f = \tau_{drive} - \tau_{rr} - \tau_d$$

$$\tau_{wheel} = \frac{\tau_a + (f - f_a)\frac{\tau_b - \tau_a}{f_b - f_a} \cdot x_g \cdot x_d \cdot e - C_{rr} \cdot v - \frac{1}{2}\rho \cdot v^2 \cdot C_d \cdot A}{n}$$

"*real buses should lean around corners*"

suspension

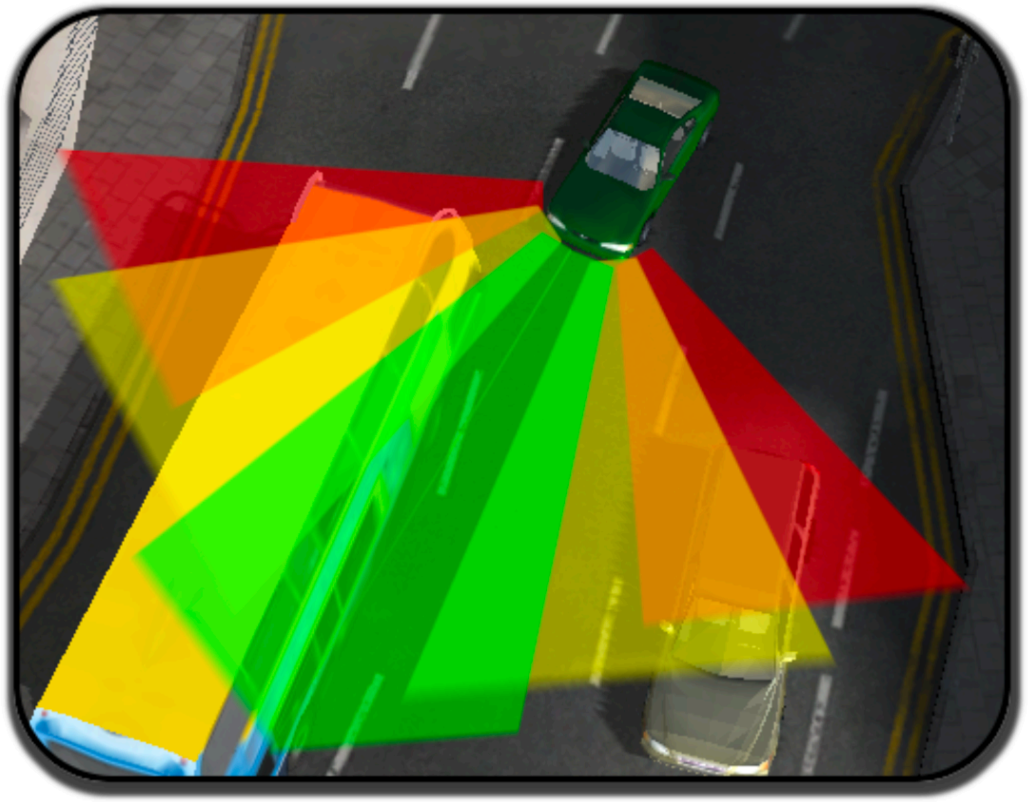
$$F_s = -k \cdot x = m \frac{\delta^2 x}{\delta t^2}$$

$$F_d = -c \cdot v = -c \cdot \frac{\delta x}{\delta t}$$

$$erp = \frac{t \cdot k}{t \cdot k + c}$$

$$cfm = \frac{1}{t \cdot k + c}$$

- *And avoid obstacles*
- Fuzzy navigation rules
- Too many parameters and unknowns



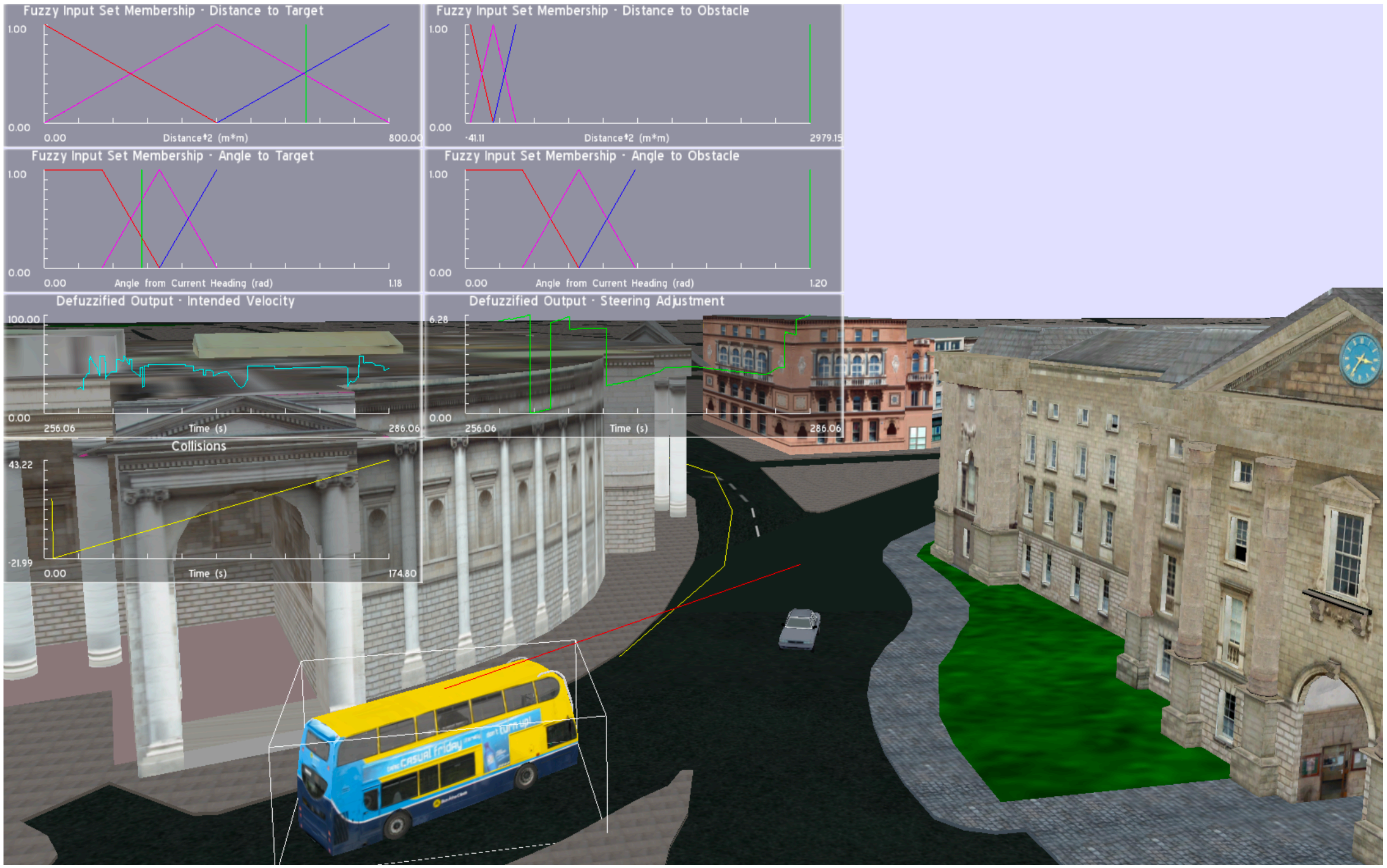

Fuzzy Input Set Membership - Distance to Target
Fuzzy Input Set Membership - Distance to Obstacle
Fuzzy Input Set Membership - Angle to Target
Fuzzy Input Set Membership - Angle to Obstacle
Defuzzified Output - Intended Velocity
Defuzzified Output - Steering Adjustment
Collisions
Distance*2 (m*m)
Angle from Current Heading (rad)
Time (s)

Figure 5.3: A coordinating agent embedded in the road has planned a path (yellow line) for the bus based on its abstract representation of the road; a linked list of segment along the road and the occupancy and state of each segment. The overlay shows the bus' agent handling smooth steering and that it is watching the nearest moving obstacle (red line pointing to the car) should it need to react to avoid colliding with it.

Evolutionary Algorithms (EA)

- "*I don't know what's best - randomly tweak stuff and **converge** to solution.*"
- *Randomishly* initialise a set of new steering rules
    - call this input data a **chromosome**
    - base rules on some biology-inspired ideas {clone, mutate, cross-over}
- Set up a test course with some random conditions
- **Repeat** many many times
- Grade success on some **fitness function**
- Promote successful rule-sets to seed next **generation**

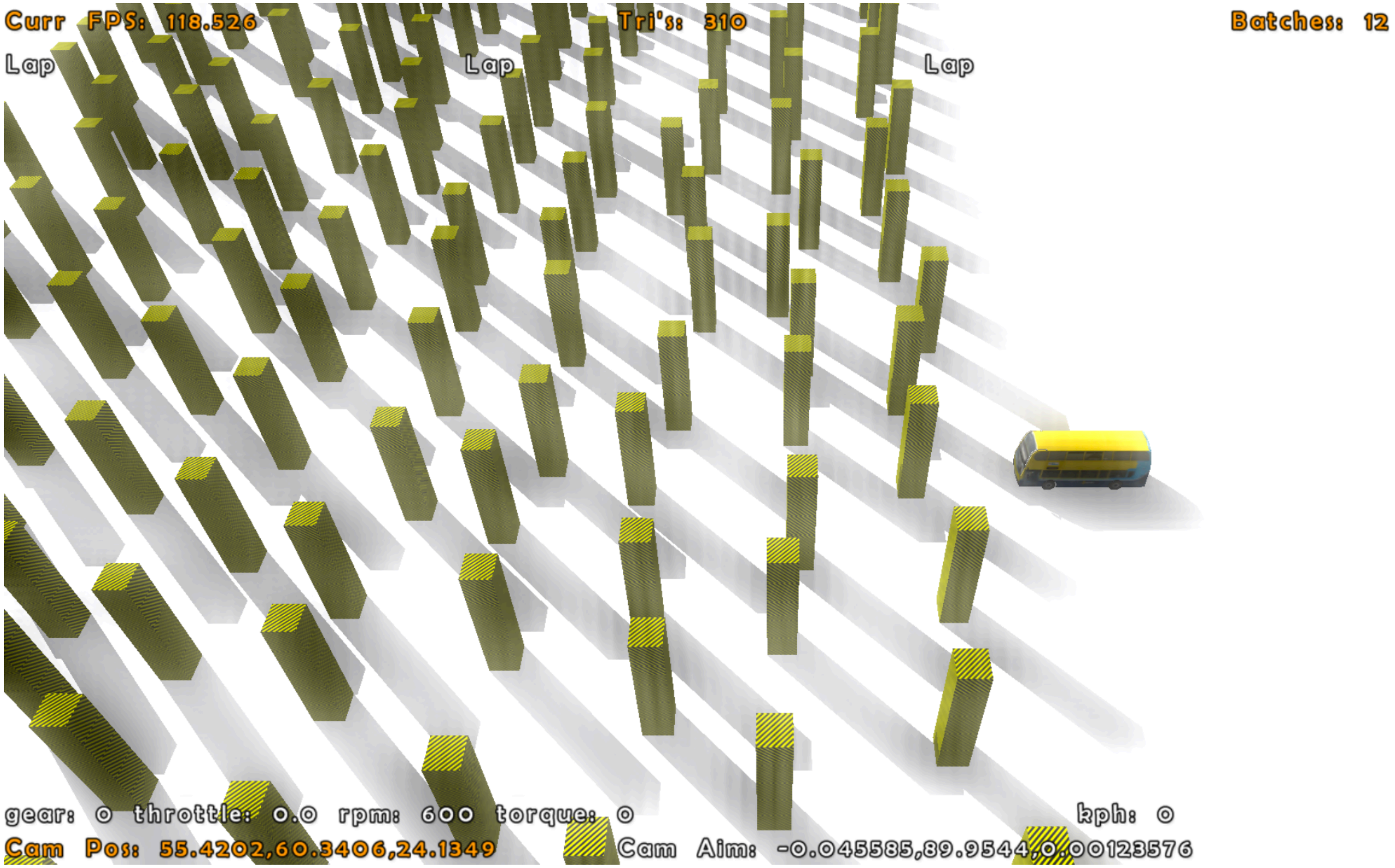
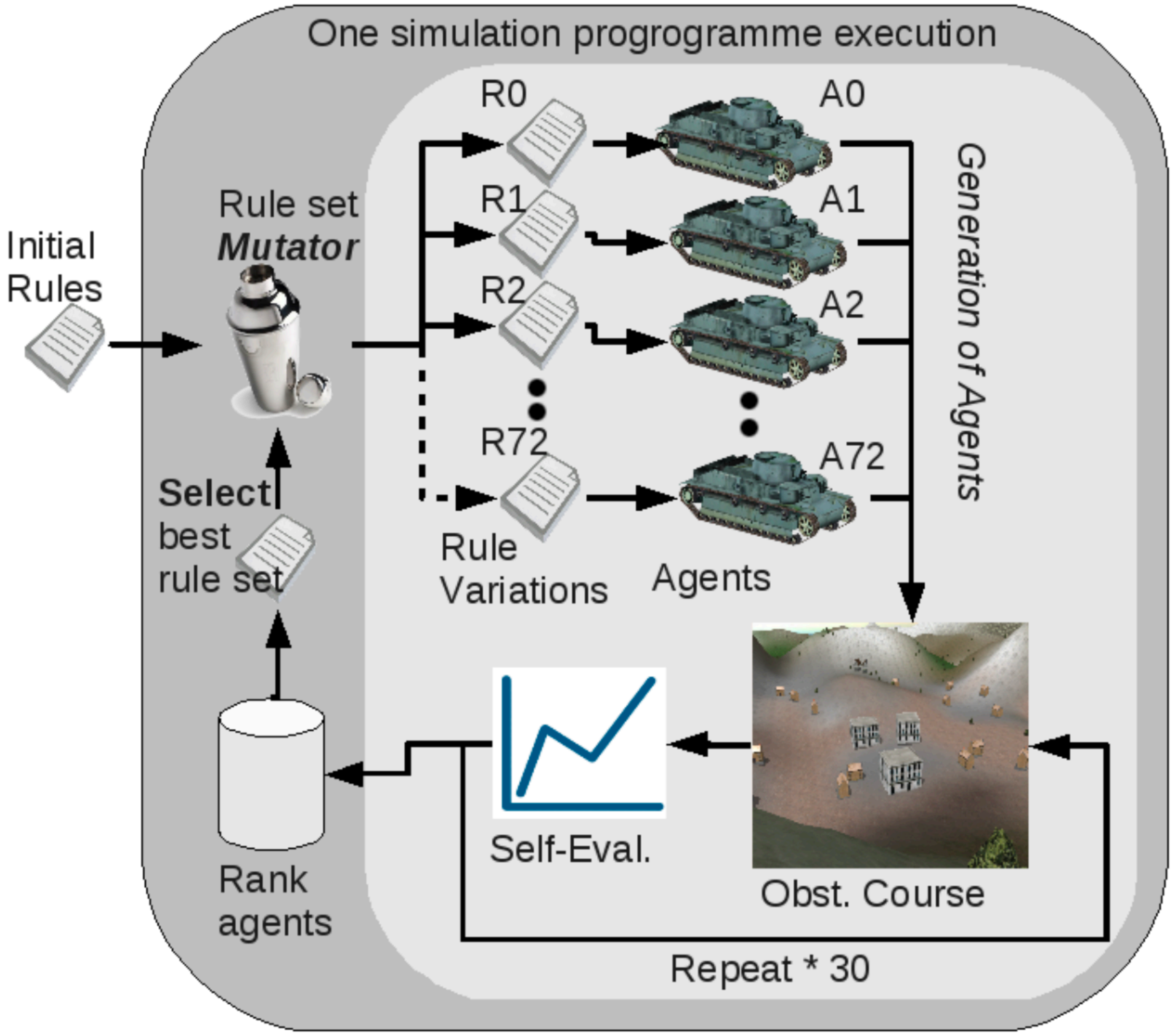

Curr FPS: 118.526
Tri's: 310
Batches: 12
Lap
Lap
Lap
gear: 0 throttle: 0.0 rpm: 600 torque: 0
kph: 0
Cam Pos: 55.4202,60.3406,24.1349
Cam Aim: -0.045585,89.9544,0.00123576

Figure 12.1: This obstacle course was designed to bear a resemblance to the “forest” obstacle course from the OpenSteer library. The vehicle, pictured at its starting position, must move 120m autonomously to the other side of a lattice field of randomly scattered cuboid “tree” obstacles. Each obstacle measures 2 × 2 × 10m and weighs 1000kg. Physically simulating the obstacles means that heavier vehicles can learn to push through if they are slowed less by this than by moving around.

Genetic Algorithms (GA)

- Like most EAs - completely unnecessary complexity.
- Never gives the best solution $fitness_x = t - d$
- Improves slowly in one obvious direction
- Success depends on fitness function
- Too many variables to tweak and it takes far too long
    - took <u>days</u> to collect results
    - small mistake = start again


One simulation progrogramme execution
Initial Rules
Rule set *Mutator*
R0
R1
R2
R72
Rule Variations
A0
A1
A2
A72
Agents
Generation of Agents
Obst. Course
Self-Eval.
Repeat * 30
Rank agents
Select best rule set

Control System and Rules

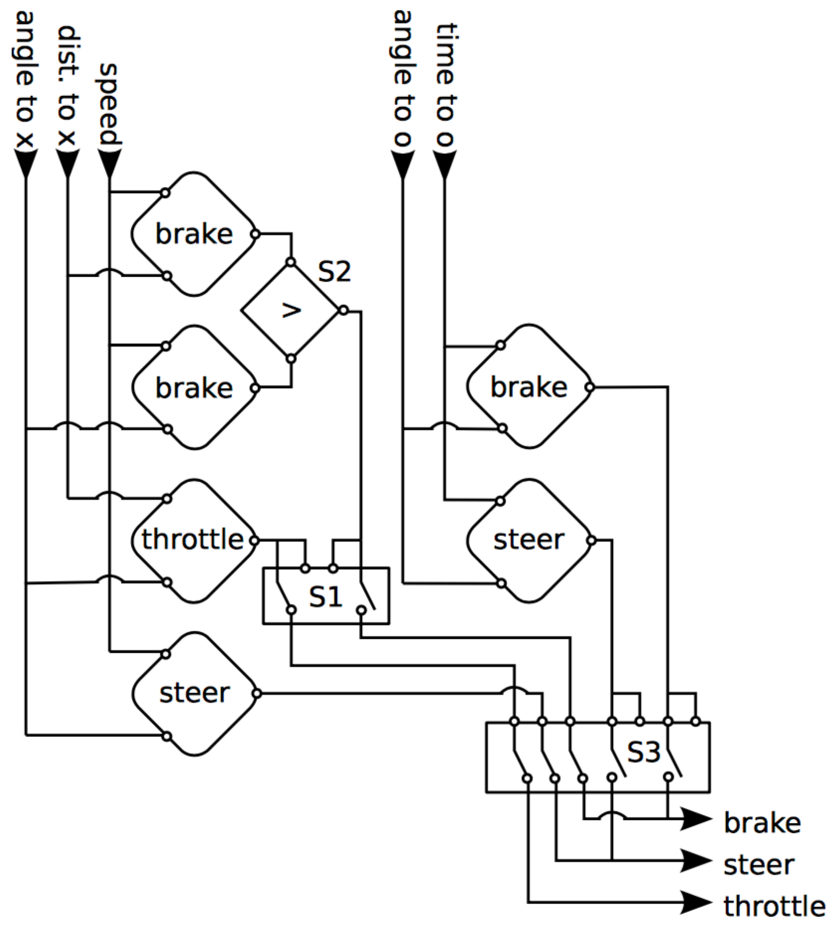


Figure 12.4: A mixed function motion control system with algorithmic switches. A reactive collision avoidance system has been added with two controllers that consider estimated time (distance / speed) to and angle to nearest obstacle o. Switch $S3$ flips between seeking and avoidance outputs and activates with any significant collision avoidance output.

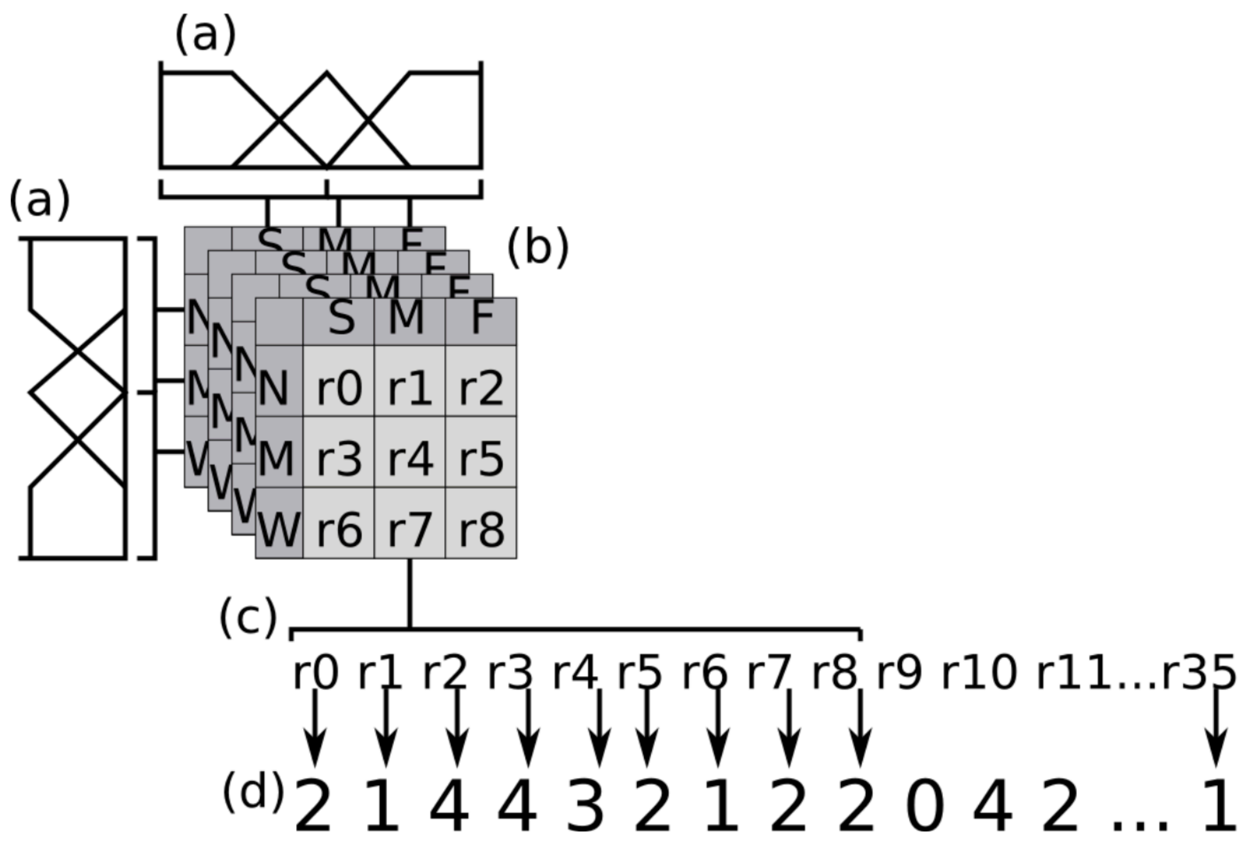


Figure 10.6: The illustration shows how fuzzy input values (a) are mapped to a 3x3 fuzzy rule tables (b) and finally to a chromosome as a string of digits (d).

Results

Generation	Chromosome
5	003240122044101410201004000020402030
6	003020002032013220200030100000103030
7	002031013031004011110133100003211030
8	012011000042004010100022300002110210
9	012011000042004010100022300002110210
10	014110000242004010400002310004000230
11	204100100141014001011010001000313200
12	204000100042002010101001000022011101
13	114020000042102210101000101222012101
14	014000000032002000130003100012200001
15	014000000032002000130003100012200001
25	102011020140001020011000100002103000
50	200000000100030000230010300020200000
75	021011000000000000000000001031001000
100	012000000100001000000000001010001001
200	000000100000000110000100021000000000
500	000000000001000100000010011000000000
999	000001000000000100000000000001000010

Table 10.3: The most-fit chromosomes from selected generations in an evolutionary run. The chromosomes are converging towards an optimal individual of all zeros.

Results

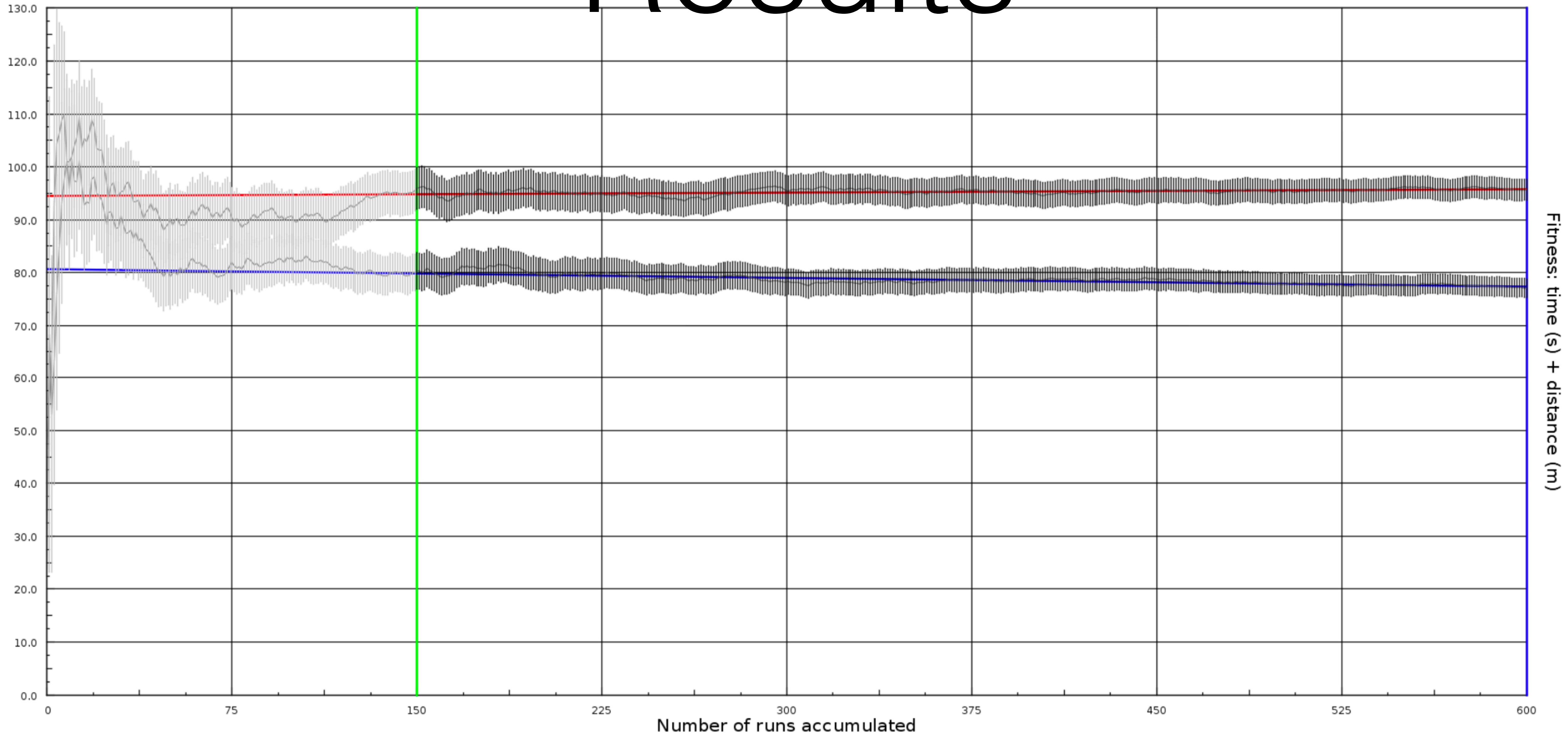


Figure 12.5: This graph gives us accumulated fitness scores and uncertainty measurements for two different vehicles being driven by the fuzzy motion controller from Figure 12.4. The first thing that the graph shows us is that the fitness measurement for both vehicles stabilises after about 120-150 runs have been collected. We can fit straight lines to data collected after the 150 run threshold (illustrated by the horizontal blue and red fits) so we can say for the simulation used that 150 runs is a safe threshold to stop evaluating with a reliable estimate of fitness. We can also see a clear distinction between the result from both vehicles (the error bars do not overlap), which indicates that the fitness function makes a course enough evaluation to distinguish between different motion control results. The simulation appears to have a fixed spread of results of ± 5 fitness points due to the robust (randomised) starting conditions.

Summary

- Directed graph network handy for traffic/routes
- Intersections are vertices (nodes)
- Edges are linked lists to control lane flow/direction
- Fuzzy logic handy for simplifying steering rules
- Evolutionary algorithms are ridiculous complexity (but fun) - simpler solution would be better
- Visualisation tools and interactive tweaking - v useful