Road Network Simulation Using FLAME GPU

Peter Heywood(✉), Paul Richmond, and Steve Maddock

Department of Computer Science, The University of Sheffield, Sheffield, UK
{ptheywood1,p.richmond,s.maddock}@sheffield.ac.uk

Abstract. Demand for high performance road network simulation is increasing due to the need for improved traffic management to cope with the globally increasing number of road vehicles and the poor capacity utilisation of existing infrastructure. This paper demonstrates FLAME GPU as a suitable Agent Based Simulation environment for road network simulations, capable of coping with the increasing demands on road network simulation. Gipps' car following model is implemented and used to demonstrate the performance of simulation as the problem size is scaled. The performance of message communication techniques has been evaluated to give insight into the impact of runtime generated data structures to improve agent communication performance. A custom visualisation is demonstrated for FLAME GPU simulations and the techniques used are described.

Keywords: Parallel agent based simulation · FLAME GPU · Transport network · Performance scaling · Visualisation

1 Introduction

With the increasing number of vehicles on road networks around the world and poor utilisation of existing road infrastructure capacity, there is a need for improved systems for planning and trialling proposed changes to traffic management system [16,26]. To avoid making costly or risky changes in the real world, computer modelling and simulation can be applied to evaluate proposed solutions. Transport simulation models can typically be classified into one of three categories: *macroscopic*, *mesoscopic* and *microscopic*. Macroscopic simulations (*top-down*) treat traffic as a liquid, modelling the overall system level behaviour. Mesoscopic simulations (*middle-out*) model platoons (groups of vehicles), using aggregate functions to calculate travel times and speeds [6]. Microscopic simulations (*bottom-up*) model the individual vehicles in the system and the interaction between them [23], and can be considered synonymous with Agent Based Simulations (ABS) [5]. In ABS behaviours are modelled at the individual level with agents interacting with both the local environment and other agents. Higher system level behaviours can emerge from the lower level behaviours and interactions modelled by the system.

Agent based modelling and microsimulation are being used in place of more traditional macro-level simulations as they offer a more natural method

S. Hunold et al. (Eds.): Euro-Par 2015 Workshops, LNCS 9523, pp. 430–441, 2015.
DOI: 10.1007/978-3-319-27308-2_35

of describing systems and allow for the emergence of more complex behaviour. Large scale and more-complex microscopic simulations are computationally expensive [3] when being processed in serial, but have been shown as suitable for acceleration through parallel methods [15,20]. Each agent in the system follows the same model, performing the same operations, as other agents of the same type, making aspects of microsimulation SIMD (Same Instruction Many Data) in nature and therefore an ideal task for General Purpose computing on Graphics Processing Units (GPGPU), which is well suited to SIMD problems. GPGPU computing has been shown to be effective for microsimulation of transport systems [24,27] and the associated tasks [7], showing that it can be used to provide computational speed-up or allow increased complexity for a similar execution time, while using low cost hardware.

This paper demonstrates the performance implications of applying the Graphics Processing Unit (GPU) to microsimulation of transport networks. It makes a unique contribution by evaluating performance scalability through the implementation of an artificial road network in which the size of the road network and the density of traffic within it can be effectively benchmarked. The Gipps car following model has been implemented using the *Flexible Large-scale Agent Modelling Environment for Graphics Processing Unit* (*FLAME GPU*) framework and the message communication techniques have been evaluated to give insight into the impact of runtime generated data structures to improve agent communication performance. A custom visualisation using GPU instancing has been developed which enables interactive simulation observation. The improved simulation performance described by this work is important in demonstrating simulations which are able to handle the increasing scale and complexity which are required to accurately model transport networks of increasing size. A coupled visualisation has the additional benefit of ensuring that traffic simulation is accessible to stakeholders involved in traffic management and infrastructure changes who are often non-modelling specialists [16].

2 Related Work

Transport microsimulation models have been used to analyse and study a broad range of driver behaviours, infrastructure layouts and the effects of traffic [4,9,12,19], showing that there are real world uses for microsimulation of transport systems for both analysis and planning without the need for costly and potentially dangerous real world experimentation.

For any study of transport systems at the microscopic level, driver behaviours such as car following [10,25], lane changing [11], interaction with road signals or traffic calming measures [13], amongst others, need to be modelled to achieve a valid, fully functional system which can be used to study other aspects of transport network behaviours.

For the purposes of this paper an example road user behaviour model is required for implementation, car following is chosen as it is arguably the most important behaviour when modelling road networks at the microscopic level.

Car following is the behaviour that on a single lane road, or within an individual lane on sections of road with multiple lanes, vehicle drivers wish to drive at their desired speed without colliding with the vehicle ahead [11].

The car following theory is that a driver reacts to the behaviour of the car(s) in front. Early models were mainly concerned with the trailing car's acceleration being proportional to the relative speeds of each car at an earlier time, or a 'history' of relative speeds [14]. Later models took into account the reaction time of the vehicle driver [10], as there is a delay between an individual receiving stimuli and performing the correct action.

The performance limits of both the driver and vehicle also affect car following behaviour, as safety concious drivers will ensure there is a gap large enough in which to stop between them and the vehicle in front (limited by an acceptable level of braking). Vehicles also have limits of acceleration performance, and so a trailing car will accelerate with no more than the vehicle's maximum acceleration until it nears the desired speed (or the speed of the vehicle in front) at which point the rate of acceleration will decrease to zero [10]. Models which are based on the principal of following at a safe distance such as Gipps' model are known as *safety distance* models.

Other models and categories of car following models have been developed, such as the *Intelligent Driver Model* proposed by Treiber et al. [25]. At it's core this model bases the acceleration of the trailing vehicle on the ratio between the "desired minimum gap" and the actual gap to the vehicle in front, with special cases for scenarios such as when traffic is in equilibrium where drivers will keep a gap to the vehicle in front which is dependant on velocity. *Psycho-physical* car following models are an alternate category in which models adjust the reactions of the driver based on the state of the vehicle [22].

3 Model

In order to demonstrate the scalable performance of Agent Based Simulations using an accelerator such as the GPU, Gipps' Car Following Model [10] was selected for implementation as it is one of the most extensively used car following models [8].

The model aims to "mimic the behaviour of real traffic", have "parameters which correspond to obvious characteristics of drivers and vehicles" to avoid elaborate calibration procedures and "should be well behaved when the interval between successive recalculations of speed and position is the same as the reaction time"[10]. The model combines limits on the performance of the driver and vehicle with the assumption that a following driver will drive in such a manner that they can safely stop if the vehicle ahead stops suddenly.

The model states that the vehicle n should not exceed its driver's target speed and the vehicle's free acceleration should increase with speed then decrease as the desired speed is approached [10]. The inequality in Eq. 1 combines these two constraints:

$$v_n(t+\tau) <= v_n(t) + 2.5a_n\tau(1 - v_n(t)/V_n)(0.025 + v_n(t)/V_n)^{\frac{1}{2}} \quad (1)$$

Table 1. Notation for variables used by Gipps' car following model

a_n	the maximum acceleration of vehicle n
b_n	the most severe braking that the vehicle n will undertake
s_n	the effective size of vehicle n, including a margin
V_n	the target speed of vehicle n
$x_n(t)$	the location of the front of vehicle n at time t
$v_n(t)$	the speed of vehicle n at time t
τ	constant reaction time for all vehicles

The terms of Eq. 1 are explained in Table 1. The limitation of braking safely can be represented by the inequality shown in Eq. 2. This takes into consideration the following driver's reaction time, and a margin of error on the driver's behalf $\theta = \tau/2$ to avoid maximum braking and an estimate of the leading driver's most severe braking $\hat{b}$ which cannot be estimated by direct observation [10].

$$v_n(t+\tau) <= b_n\tau + \sqrt{{b_n}^2\tau^2 - b_n(2[x_{n-1}(t) - s_{n-1} - x_n(t)] - v_n(t)\tau - v_{n-1}(t)^2/\hat{b})} \quad (2)$$

Assuming that if the driver travels as fast as safely possible considering vehicle limitations, the new speed can be given by Eq. 3.

$$v_n(t+\tau) = \min\Bigg\{ v_n(t) + 2.5a_n\tau(1 - v_n(t)/V_n)(0.025 + v_n(t)/V_n)^{\frac{1}{2}}, \\ b_n\tau + \sqrt{{b_n}^2\tau^2 - b_n[2[x_{n-1}(t) - s_{n-1} - x_n(t)] - v_n(t)\tau - v_{n-1}(t)^2/\hat{b}]} \Bigg\} \quad (3)$$

The main restriction of Gipps' car following model is that the time-step of the simulation needs to be set to the driver reaction time τ. The model also relies on assumptions that drivers drive in a safe manner (which is often not the case) and that they are capable of estimating observable characteristics of the leading vehicle with some degree of accuracy.

4 Implementation

This section consists of four parts: the artificial road network, an overview of FLAME GPU, how Gipps' car following model is implemented using FLAME GPU & the visualisation.

4.1 Scalable Artificial Road Network

For the purposes of this paper a simple, single lane, artificial grid network is generated for agents to navigate. Road networks in the real world are designed

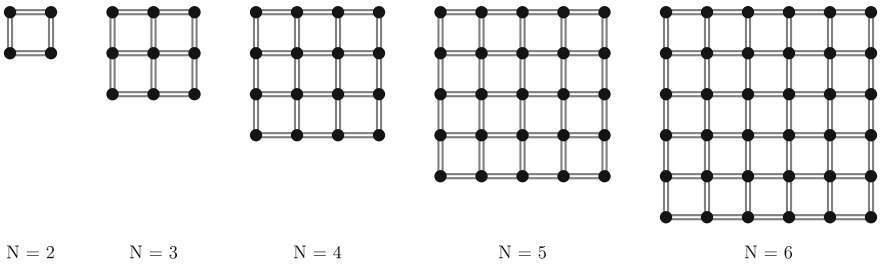


Fig. 1. Examples of the artificial road network with varying values of N. Circles represent junctions and lines represent sections of road

with real world limitations such as terrain or the location of existing roads, buildings and other spatial constraints. An artificial road network is preferable for performing controlled studies as the properties of the road network are consistent at different scales. The artificial road network is a uniform grid network, generated based on the target grid size N where $N >= 2$, as shown in Fig. 1. The grid contains N rows and N columns of junctions, with two sections of road between each adjacent node, hence the road network contains N^2 junctions and $4N(N-1)$ one-way sections of road.

4.2 FLAME GPU

FLAME GPU [21] is a "template based simulation environment" for agent based simulation on Graphics Processing Unit (GPU) architecture. Agents are represented as X-Machines which can communicate via globally accessible message lists. Messages are crucial for interaction between agents and they can be partitioned to "ensure the most optimal cycling of messages" [21].

There are currently three defined message partition schemes in FLAME GPU: non-partitioned, discrete 2D space partitioning and 2D/3D spatially partitioned space. *Non partitioned messaging* involves no filtering to reduce the number of messages each agent must iterate; it is merely a brute force approach. *Discrete partitioned messages* can be used only when sending from non mobile discrete agents, and is therefore best suited to discrete systems using Cellular Automata (CA) based approaches. *Spatially partitioned messages* originate from agents in continuous space (such as a 2D or 3D environment). The partitioning scheme requires a radius and environment bounds. The radius dictates the range to which the message iteration will extend to from its originating point, while the bounds are used to limit the area of partitioning.

Of these partitioning schemes both non partitioned messaging and spatially partitioned messaging are of interest with potential performance implications. Non partitioned messaging is computationally expensive ($O(n^2)$ message iteration loop when reading messages) but has little overhead cost with respect to dynamic construction of data structures compared to partitioned messaging schemes. Simulations using partitioned messaging generally lead to less expensive

message iteration loops but an increased overhead cost as long as the partitioning system is set-up appropriately. Both non partitioned messaging and spatially partitioned messaging will be evaluated and the performance difference compared.

4.3 Implementing Gipps' Car Following Model Using FLAME GPU

Each vehicle in the system is represented by an agent. The initial values of each agent are generated by a Python script and stored in a FLAME GPU XML file for simulation state data to be parsed at runtime and copied into device memory.

The road network is stored in CUDA constant memory (via FLAME GPU's Simulation Constants) as it will not change over the course of the simulation and each agent in the simulation interacts with the same road network. This does however impose a limit on the number of nodes and roads possible within the system, due to the current limit of 64kB of constant memory being available. An alternative would be to store the road network in the CUDA Read-Only Data Cache [17] but this would restrict the GPU architectures which could be used to CUDA Compute Capability 3.0 (Kepler) and greater.

For each step in the simulation, each agent first outputs it's observable properties (location, velocity, etc.) as a message using the selected message partitioning scheme. Each vehicle then iterates through the list of relevant messages based on the partitioning scheme to find the message from the lead vehicle. Gipps' car following model (Sect. 3) is applied to the current vehicle, using the *Forward Euler Method* to calculate the vehicle's new location and velocity. Higher-order integration methods could be considered, with further related study on accuracy and performance trade-offs. Agents which have reached the end of their current segment of road randomly select a new segment from the roads connected at their current junction.

4.4 Visualisation and Graphics Techniques

A custom, cross platform visualisation is implemented to visualise the agents moving about the 3D world using OpenGL and Simple DirectMedia Layer [2], written in C++. FLAME GPU stores agent data on the GPU and it is best to avoid unnecessary and unneeded copying between device memory and host memory so CUDA *OpenGL Interoperability* [18] is used to populate *OpenGL texture buffers* with the location, length and orientation of each agent within the simulation via a CUDA kernel. *Instanced rendering* [1] allows primitives to be rendered at the correct world location for each agent, correctly orientated via custom vertex shaders. Example output of the visualisation is shown by Fig. 5.

5 Experimental Results

Two sets of benchmarking experiments are carried out to evaluate (1) the performance of the simulation as the number of agents within the system is increased,

Table 2. Suggested model parameters proposed by Gipps

a_n	sampled from the normal distribution $N(1.7, 0.3^2)$ m/sec^2
b_n	$-2.0a_n$
s_n	sampled from the normal distribution $N(6.5, 0.3^2)$ m
V_n	sampled from the normal distribution $N(20.0, 3.2^2)$ m/sec
τ	2/3 s
$\hat{b}$	the minimum of -3.0 and $(b_n - 3.0)/2$ m/sec^2

and (2) the overall scale of simulation while maintaining a fixed vehicle to road ratio. The two message partitioning schemes are compared and the performance difference highlighted for each set of experiments. Agent populations are generated using the model parameters defined in Table 2 and randomly distributed throughout the road network. The simulation has been implemented in FLAME GPU 1.4 for CUDA 7.0. Results are obtained from an Intel Core i7 4770 K machine using an NVIDIA TESLA K20c GPU with CUDA 7.0.

5.1 Vehicle Scaling for Static Road Network

The first experiment illustrates the performance scaling of the agent based simulation with respect to the number of agents in the system. A fixed grid network of size $N = 16$ and road length of $10000\,m$ is used and the number of agents varies from 2^8 up to 2^{18}, using both message partitioning techniques.

Figure 2 shows the performance against the number of agents, where performance has been measured by averaging the simulation time over 100 iterations. At very small numbers of agents, non partitioned messaging demonstrated similar performance to partitioned messaging, but with greater numbers of agents the gap in performance increases dramatically with each increase in agent population size. Spatially partitioned messaging shows a similar level of scaling with each radius, but the lower radius consistently shows higher levels of performance.

Figure 3 shows the average agent iteration performance (calculated as average iteration time/population size). With small agent counts ($<= 2^{13}$), as the population size increases the simulation performance increases as hardware utilisation increases. Larger populations ($> 2^{13}$) show a decrease in agent performance as agent population size increases due to the increased quantity of messages each agent reads. Non partitioned messaging shows significantly lower per-agent performance as the brute force approach causes each agent to read a message from each agent in the population. As the population is increased the density of agents increases on the fixed size road network. This causes agents using spatially partitioned communication to, on average, receive an increased amount of messages, but still less than would be received using the brute force approach.

The larger population sizes show a decrease in per agent performance for each increase in agent population, with a significant decrease in performance for non partitioned messaging when compared to spatially partitioned messaging.

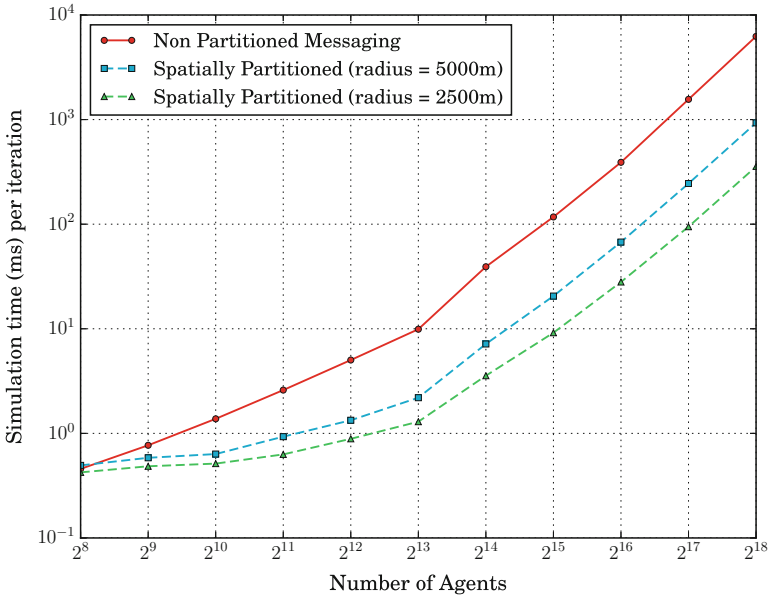


Fig. 2. Average iteration execution time for fixed grid of size $N = 16$ against agent population size, averaged over 100 iterations

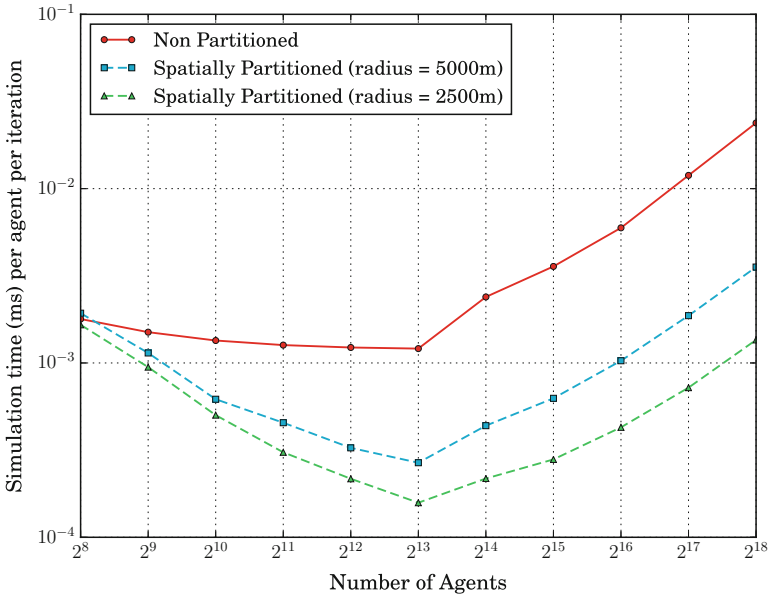


Fig. 3. Average iteration execution time per agent for fixed grid of size $N = 16$ against agent population size, averaged over 100 iterations

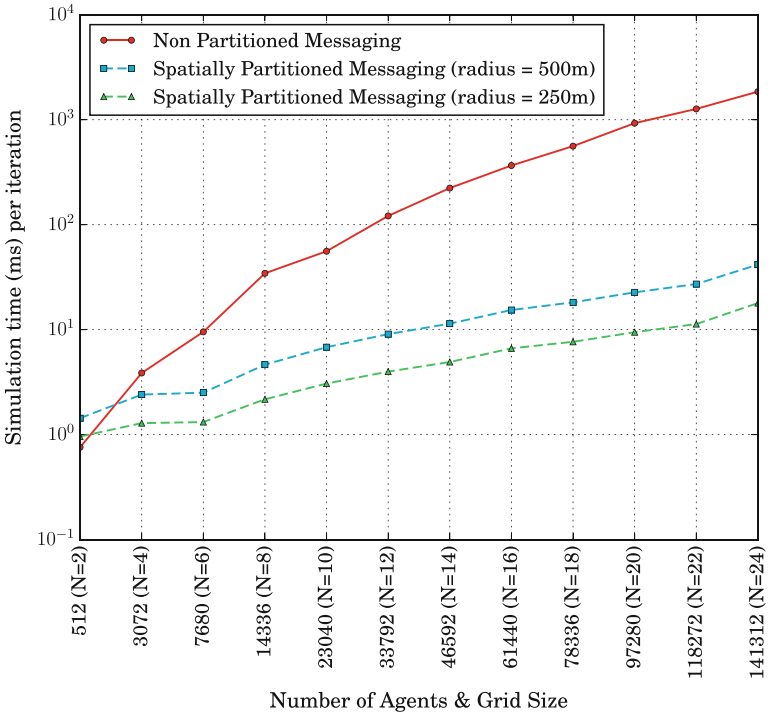
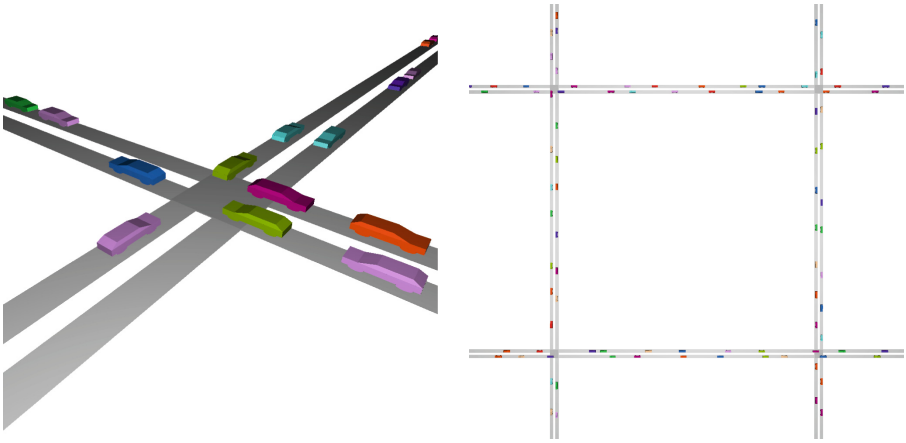


Fig. 4. Average iteration execution time for increasing Grid Size N with a fixed vehicle density of 64 agents per 1000 m, averaged over 100 iterations



(a) Nearby (b) Overview

Fig. 5. Screen captures of the custom visualisation running

5.2 Scaling of Vehicle and Road Network

The second experiment illustrates the performance scaling of the agent based simulation with respect to the problem size as a whole - the number of agents and size of grid generated have been scaled in unison, maintaining a constant vehicle count per section of road (64 vehicles per $1000\,m$ of road) from $N = 2$ & 512 agents to $N = 24$ & 141312 agents, with a road length of $1000\,m$. The vehicle to road ratio is kept constant to illustrate the performance difference from increasing the size of the problem as a whole, rather than just increasing the density of vehicles. Both message partitioning techniques are used and the performance difference highlighted, including using different spatial partitioning radii ($250\,m$ and $500\,m$).

Figure 4 shows the performance against the scale of simulation, where performance has been measured by averaging the simulation time over 100 iterations. This shows that as the overall size of simulation is scaled up the simulation time also increases. Non partitioned messaging shows greater performance at very small simulation sizes compared to partitioned communication. This is due to the additional computational overhead of spatial partitioning. Larger problem sizes show the benefits of using a partitioned messaging scheme to reduce the number of messages each agent must iterate through, as the performance impact of a greater problem size is significantly less for larger problem sizes. For a grid size of $N = 24$ average iteration time was approximately 44.4 & 103.4 times quicker using spatially partitioned communication with radii of $500\,m$ & $250\,m$ respectively than non partitioned messaging.

This experiment was initialised with uniform vehicle distribution. Whilst uneven distributions then occur as the simulation progresses, future work needs to consider this aspect and any associated performance implications in more detail.

5.3 Visualisation

Running the simulation with a visualisation has a negative impact on the performance of the simulation. For a simulation with grid size $N = 8$, road length $1000\,m$ and 8192 vehicles, 1000 iterations of simulation takes 15079 ms without visualisation compared to 16291 ms with visualisation using an *NVIDIA GeForce GTX 660*. This shows a 1.08x increase in simulation time when using the GPU for both simulation and visualisation.

6 Conclusions

Two experiments have been carried out, demonstrating that GPUs are suitable for Agent Based Simulation of road networks and are capable of scaling to handle the increasing demands of road network simulations, and the performance difference between message partitioning schemes is highlighted.

Both experiments show that FLAME GPU can successfully be used for agent based simulations of road networks and can process large amounts of agents in

a reasonable time frame. The experiments also highlight the performance difference between the two message partitioning techniques. Non partitioned messaging only outperforms spatially partitioned messaging when the number of agents in the system is low enough that the additional overhead of the partitioning scheme outweighs the performance improvements and so for agent based simulations where a generally large populations of agents are used spatially partitioned messaging is more suitable, but the choice of spatial partitioning parameters need to be selected carefully. Smaller partitioning radii generally increases the performance but the radius needs to be relevant to the target simulation so that messages from relevant agents still reach each other.

Future work developing alternate message partitioning techniques specific to networked systems should allow further increase in performance compared to spatially partitioned messaging. On a road network a driver is only concerned with vehicles on the same section of roads and connected sections of roads at junctions. Developing a messaging scheme which limits the incoming agent messages to the road network structure should reduce the cost of the message iteration loop, without too large of a performance overhead. An example visualisation has also been described, demonstrating some of the techniques which can be used to create custom visualisations for FLAME GPU based simulations.

References

1. OpenGL SDK glDrawArraysInstanced manpage. https://www.opengl.org/sdk/docs/man/html/glDrawArraysInstanced.xhtml
2. Simple DirectMedia Layer (libSDL). https://www.libsdl.org/
3. Algers, S., Bernauer, E., Boero, M., Breheret, L., Di Taranto, C., Dougherty, M., Fox, K., Gabard, J.F.: Review of micro-simulation models. Institute for Transport Studies (1997)
4. Bell, M.C., Galatioto, F., Giuffrè, T., Tesoriere, G.: Novel application of red-light runner proneness theory within traffic microsimulation to an actual signal junction. Accid. Anal. Prev. **46**, 26–36 (2012)
5. Bonabeau, E.: Agent-based modeling: methods and techniques for simulating human systems. Proc. Nat. Acad. Sci. **99**(suppl 3), 7280–7287 (2002)
6. Charypar, D., Axhausen, K.W., Nagel, K.: Event-driven queue-based traffic flow microsimulation. Transp. Res. Rec. J. Transp. Res. Board **2003**(1), 35–40 (2007)
7. Charypar, D., Nagel, K.: Generating complete all-day activity plans with genetic algorithms. Transportation **32**(4), 369–397 (2005)
8. Ciuffo, B., Punzo, V., Montanino, M.: Thirty years of gipps' car-following model. Transp. Res. Rec. J. Transp. Res. Board **2315**(1), 89–99 (2012)
9. García, A., Torres, A.J., Romero, M.A., Moreno, A.T.: Traffic microsimulation study to evaluate the effect of type and spacing of traffic calming devices on capacity. Procedia-Soc. Behav. Sci. **16**, 270–281 (2011)
10. Gipps, P.G.: A behavioural car-following model for computer simulation. Transp. Res. Part B Methodological **15**(2), 105–111 (1981)
11. Gipps, P.G.: A model for the structure of lane-changing decisions. Transp. Res. Part B: Methodological **20**(5), 403–414 (1986)
12. Keenan, D.: M 62-m 1 interchange- a combined transyt/vissim micro-simulation assessment. Traffic Engi.+Control **46**(5), 178–187 (2005)

13. Kesting, A., Treiber, M., Helbing, D.: General lane-changing model mobil for car-following models. Transp. Res. Rec. J. Transp. Res. Board **1999**(1), 86–94 (2007)
14. Lee, G.: A generalization of linear car-following theory. Oper. Res. **14**(4), 595–606 (1966)
15. Nagel, K., Rickert, M.: Parallel implementation of the transims micro-simulation. Parallel Comput. **27**(12), 1611–1639 (2001)
16. Neffendorf, H., Fletcher, G., North, R., Worsley, T., Bradley, R.: Modelling for intelligent mobility (Feb 2015)
17. Nvidia, C.: CUDA kepler tuning guide - read-only data cache. http://docs.nvidia.com/cuda/kepler-tuning-guide/#read-only-data-cache
18. Nvidia, C.: Cuda c programming guide (2015). http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf. Accessed 30 Mar 2015
19. Panis, L.I., Broekx, S., Liu, R.: Modelling instantaneous traffic emission and the influence of traffic speed limits. Sci. Total Environ. **371**(1), 270–285 (2006)
20. Raney, B., Cetin, N., Völlmy, A., Vrtic, M., Axhausen, K., Nagel, K.: An agent-based microsimulation model of swiss travel: first results. Netw. Spat. Econ. **3**(1), 23–41 (2003)
21. Richmond, P.: Flame gpu technical report and user guide. Technical report CS-11-03, Technical report. University of Sheffield, Department of Computer Science (2011)
22. Schulze, T., Fliess, T.: Urban traffic simulation with psycho-physical vehicle-following models. In: Proceedings of the 29th conference on Winter simulation, pp. 1222–1229. IEEE Computer Society (1997)
23. Sommer, C., Yao, Z., German, R., Dressler, F.: On the need for bidirectional coupling of road traffic microsimulation and network simulation. In: Proceedings of the 1st ACM SIGMOBILE workshop on Mobility models, pp. 41–48. ACM (2008)
24. Strippgen, D., Nagel, K.: Multi-agent traffic simulation with cuda. In: International Conference on High Performance Computing & Simulation, HPCS 2009, pp. 106–114. IEEE (2009)
25. Treiber, M., Hennecke, A., Helbing, D.: Congested traffic states in empirical observations and microscopic simulations. Phys. Rev. E **62**(2), 1805 (2000)
26. UK Department for Transport: Quarterly Road Traffic Estimates: Great Britain Quarter 4 (October - December) 2014, Februry 2015. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/402989/road-traffic-estimates-quarter-4-2014.pdf
27. Wang, K., Shen, Z.: A gpu based trafficparallel simulation module of artificial transportation systems. In: 2012 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), pp. 160–165. IEEE (2012)