Visualising Real Time Large Scale Micro-Simulation of Transport Networks

Peter Heywood, Paul Richmond & Steve Maddock

Department of Computer Science, The University of Sheffield

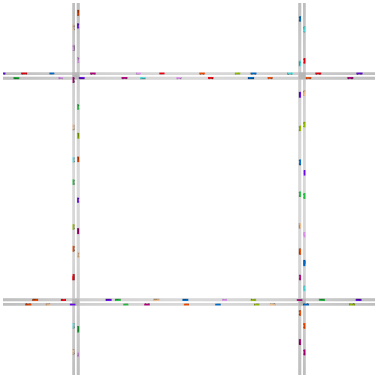
Overview

Introduction

Road Traffic Network Simulation

Visualisation & Performance

Conclusions



Introduction

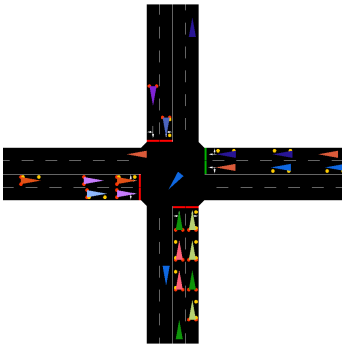
Why **Simulate** Transport Networks?

- Increasing traffic demand globally
  - UK projected increase between 2010 & 2040: [Dep15]
  - Up to 42% increase of car ownership
  - 19% to 55% growth in UK road traffic
- Poor utilisation of existing infrastructure
- Need for improved road simulation systems [NFN*15, UK 15]
- Used for planning & trialling road network changes
- Cheaper & less disruptive than real world trials

Rush hour traffic on the M6 motorway
(Mat Fascione - CC BY-SA 2.0)

Why **Visualise** Transport Network Simulations?

- Decision makers are often not modelling specialists [NFN*15]
- Visualisation increases accessibility of simulations
- Improves decision making



An example of traffic microsimulation visualisation (`sumo-gui`)

Our Aims

- Use the GPU for Agent Based Simulation of Road Network
  - Using FLAME GPU (Flexible Large Scale Agent Modelling Environment for the GPU)
  - Large-scale simulation of a road network
  - Car following behaviour on an artificial road network
- Demonstrate performance of road network simulation using FLAME GPU
  - Described in forthcoming paper “Road Network Simulation using FLAME GPU” [HRM15]
- **Develop custom visualisation for the simulation**
  - **Enable interactive simulation observation**
  - **Minimal impact on performance**

Road Traffic Network Simulation

Microsimulation, Agent Based Modelling & the GPU

Microsimulation & Agent Based Modelling (ABM)

- *Bottom up* simulations - *individual level* with *local interactions* [SYGD08]
- **ABM** provides a natural method for describing agents and behaviours
  - allows emergence of more complex behaviour
- Good for modelling congested transport networks

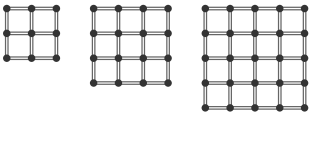
Why *General Purpose computing on Graphics Processing Units* (*GPGPU*)?

- Increased performance due to massively parallel architecture
- Microsimulation is *well suited* for GPGPU computing [SN09, WS12]
  - However it is **not** embarrassingly parallel

The Simulation

Artificial Road Network

- Scales consistently unlike real world networks
- Uniform grid of N^2 junctions & $4N(N-1)$ roads



Gipps' Car Following Model [Gip81]

- *Safety distance* car following model
- Considers driver and vehicle limitations
- Extensively used [CPM12]

FLAME GPU

- "*Template based simulation environment*" for agent based simulation on GPU architecture [Ric11]
- Provides a high level interface for describing agents, abstracting the CUDA programming model [Ric14]
- State-based agent representation
- Message-based communication

FLAME GPU

www.flamegpu.com
github.com/flamegpu

Visualisation & Performance

General Purpose computing on Graphics Processing Units

- Massively parallel architecture
- Perform same operation on many items of data (SIMD)
  - *Kernels* (GPU functions) execute same code in parallel using many threads
- Multiple memory spaces
  - Memory access pattern is important for performance
- Dedicated cards connected over PCI bus
  - *Host-Device* memory transfers are relatively slow

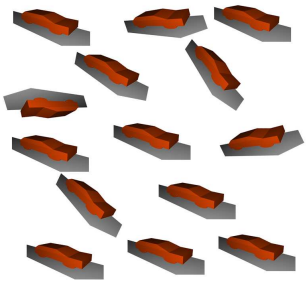
Nvidia Tesla C2075 (Source - CC0 1.0)
CC0 1.0

What is **Geometry Instancing**?

- Rendering multiple copies of the same geometry
- Vertex data is copied but modified to reduce repetition
  - Position
  - Colour
  - Animation state
- Data needs to be accessible on the GPU
  - OpenGL Buffers
- Requires fewer API calls [Khr08]

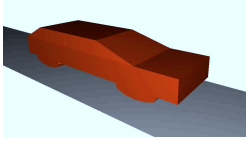
What is **Geometry Instancing**?

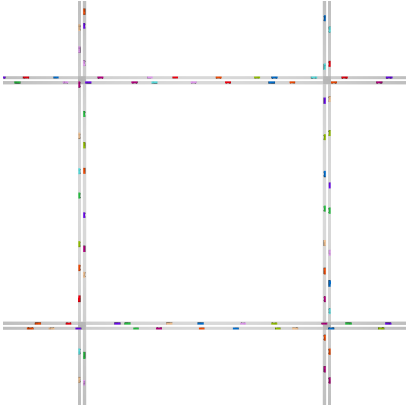
- Rendering multiple copies of the same geometry
- Vertex data is copied but modified to reduce repetition
  - Position
  - Colour
  - Animation state
- Data needs to be accessible on the GPU
  - OpenGL Buffers
- Requires fewer API calls [Khr08]



Interactive Visualisation

- Cross platform `C++`, `OpenGL` & `libSDL`[SDL]
- Mouse & Keyboard controls (*no-clip*)
- Simulation updated per frame (currently)
- Geometry loaded from wavefront (`.obj`) files
- Flat shading

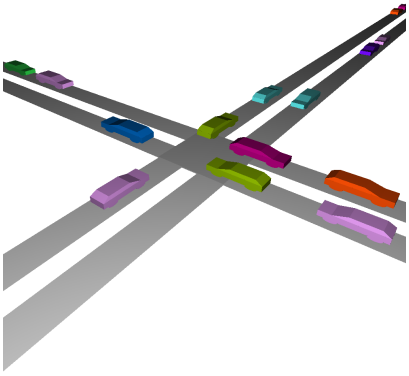




Overview of visualisation

Interactive Visualisation

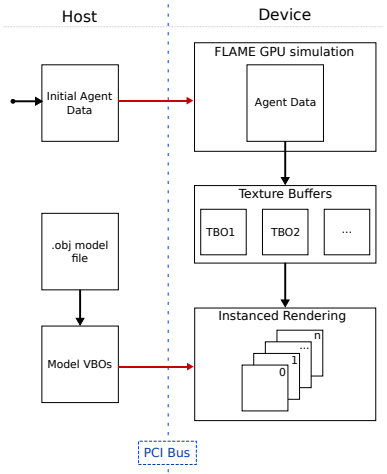
- *OpenGL Texture Buffers* populated with agent data via *CUDA OpenGL Interop* [Nvi15]
- *Geometry Instancing* [Khr] used to apply data to models
  - gvec4 texelFetch(gsamplerBuffer sampler, int P); [Khr14]
  - Reduced number of API calls [Khr08]
  - Minimises host-device memory transfers
- Fragment shader used to differentiate vehicles & apply lighting model



Nearby view of visualisation

How we use the GPU

- Road network stored in CUDA Constant Memory
  - Does not change during kernels
  - Maximum size to 64Kb currently -> CUDA Read-only Memory
- Geometry Instancing & CUDA interop
  - Avoids unnecessary host-device transfers
- FLAME GPU
  - One thread per agent
  - State-based representation minimises branching
  - Synchronisation points defined by message dependence
  - Transparently converts between AoS & SoA
  - Minimal transfer of data to host (CPU)



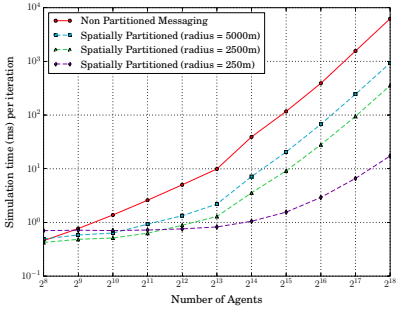
Instanced rendering memory transfers

Demonstration

Performance Impact

- Instanced visualisation has minimal performance impact
- $N = 8$, length $1000m$, 8192 vehicles & 1000 iterations
- NVIDIA GeForce GTX 660

Console	15079ms
Visualisation	16291ms
Run-time Increase	**1.08x**



Performance comparison between FLAME GPU message partitioning schemes

What Next?

Procedural Instancing

- Increase variation of instanced vehicles
- Procedurally generate data at runtime to modify instances
- Use simulation data such as vehicle length / type
- Applicable to many types of agents
  - Vehicles
  - Pedestrians
  - Environment

Other Future Work

- Analyse and visualise aggregate data using the GPU to increase accessibility
- Further performance optimisations for large populations

Conclusions

Conclusions

- Highlighted difficulties of large scale GPGPU microsimulation of transport networks
  - Expensive host-device memory transfers
  - Number of GPU draw calls
- Described & demonstrated techniques used to combat these issues
  - CUDA OpenGL Interoperability
  - Geometry Instancing
- Demonstrated minimal performance impact for an example visualisation

Thank You

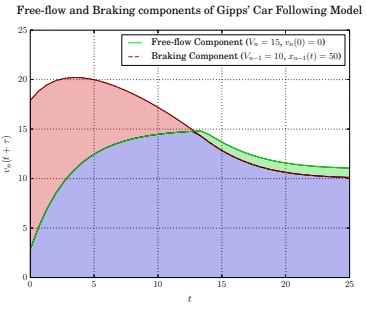
ptheywood.uk
ptheywood1@sheffield.ac.uk

Additional Slides

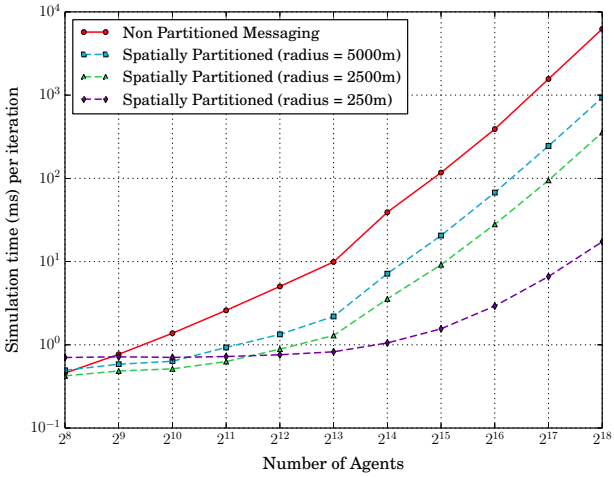
Gipps' Car Following Model Equation

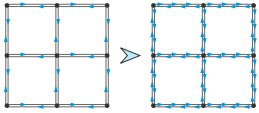
$$v_n(t+\tau) = \min\Big\{ v_n(t) + 2.5a_n\tau(1 - v_n(t)/V_n)(0.025 + v_n(t)/V_n)^{\frac{1}{2}},$$
$$b_n\tau + \sqrt{{b_n}^2\tau^2 - b_n[2[x_{n-1}(t) - s_{n-1} - x_n(t)] - v_n(t)\tau - v_{n-1}(t)^2/\hat{b}]} \Big\}$$

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
$\hat{b}$	estimate of leading vehicles most severe braking



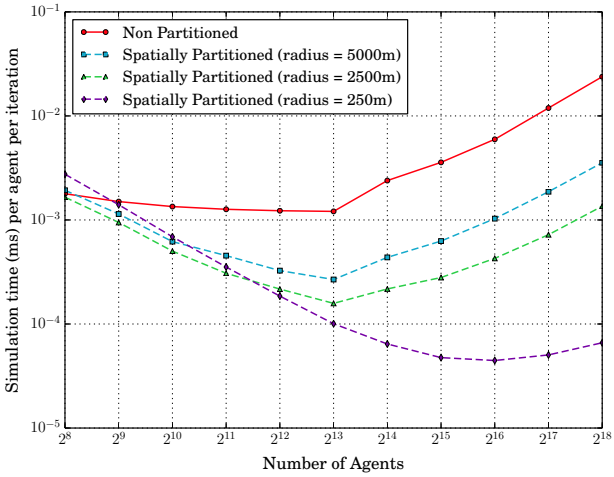
Results: Fixed Grid Network





- Spatially partitioned messaging outperforms non-partitioned messaging
- Smaller radii outperforms larger radii beyond overhead

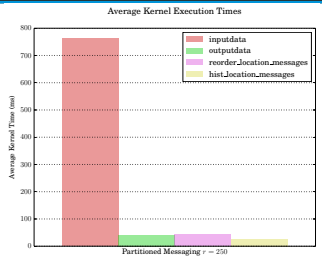
Results: Fixed Grid Network - Per Agent



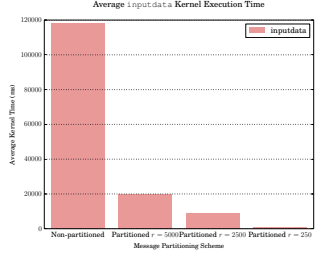
- Distinct gradient change at 2^{13} agents - **hardware utilisation** vs **larger message lists**
- Maximum message count

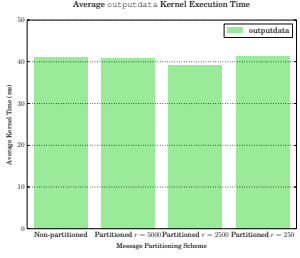
Non-partitioned	262144
Partitioned $r = 5000$	19662
Partitioned $r = 2500$	9720
Partitioned $r = 250$	309

Results: Fixed Grid Network - Kernel Profiling

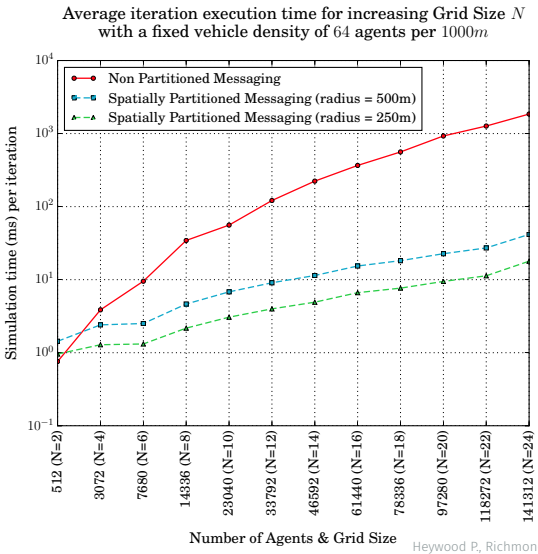


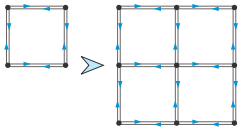
- Kernel times averaged over 10 iterations
- 32768 Agents
- `inputdata` kernel is dominant





Scaled Grid Network





- Spatially partitioned messaging outperforms non-partitioned beyond overhead
- Up to 103x performance increase for spatial partitioning than non-partitioned

Heywood P., Richmond P. & Maddock S.

Bibliography

References I

[CPM12] Ciuffo B., Punzo V., Montanino M.:
Thirty years of gipps' car-following model.
Transportation Research Record: Journal of the Transportation Research Board 2315, 1 (2012), 89–99.

[Dep15] Department for Transport:
Road traffic forecasts 2015.
`https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/260700/road-transport-forecasts-2013-extended-version.pdf`, Mar. 2015.

[Gip81] Gipps P. G.:
A behavioural car-following model for computer simulation.
Transportation Research Part B: Methodological 15, 2 (1981), 105–111.

[HRM15] Heywood P., Richmond P., Maddock S.:
Road network simulation using flame gpu.
In *Euro-Par 2015: Parallel Processing Workshops*, Lecture Notes in Computer Science. 2015. forthcoming.

[Khr] Khronos Group:
OpenGL SDK glDrawArraysInstanced manpage.
`https://www.opengl.org/sdk/docs/man/html/glDrawArraysInstanced.xhtml`.

[Khr08] Khronos Group:
GL_ARB_draw_instanced specification.
`https://www.opengl.org/registry/specs/ARB/draw_instanced.txt`, 2008.
Last accessed 2015-08-04.

References II

[Khr14] KHRONOS GROUP:
OpenGl SDK - texelFetch.
`https://www.opengl.org/sdk/docs/man/html/texelFetch.xhtml`, 2014.
Last accessed 2015-09-15.

[NFN*15] NEFFENDORF H., FLETCHER G., NORTH R., WORSLEY T., BRADLEY R.:
Modelling for intelligent mobility.
`https://ts.catapult.org.uk/documents/10631/169582/Modelling+Intelligent+Mobility,+Feb+2015/73b7c9f9-d05a-4fca-ad9f-0e226e48d6b7`, Feb. 2015.

[Nvi15] NVIDIA C.:
Cuda c programming guide.
`http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf`, Mar. 2015.
Last accessed 2015-03-30.

[Ric11] RICHMOND P.:
FLAME GPU technical report and user guide.
Tech. rep., technical report CS-11-03. Technical report, University of Sheffield, Department of Computer Science, 2011.

[Ric14] RICHMOND P.:
Resolving conflicts between multiple competing agents in parallel simulations.
In *Euro-Par 2014: Parallel Processing Workshops*, Lopes L., Žilinskas J., Costan A., Cascella R., Kecskemeti G., Jeannot E., Cannataro M., Ricci L., Benkner S., Petit S., Scarano V., Gracia J., Hunold S., Scott S., Lankes S., Lengauer C., Carretero J., Breitbart J., Alexander M., (Eds.), vol. 8805 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 383–394.

[SDL] Simple DirectMedia Layer (libSDL).
`https://www.libsdl.org/`.

References III

[SN09] Strippgen D., Nagel K.:
Multi-agent traffic simulation with cuda.
In *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on* (2009), IEEE, pp. 106–114.

[SYGD08] 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* (2008), ACM, pp. 41–48.

[UK 15] UK Department for Transport:
Quarterly Road Traffic Estimates: Great Britain Quarter 4 (October - December) 2014.
`https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/402989/road-traffic-estimates-quarter-4-2014.pdf`, Feb. 2015.

[WS12] Wang K., Shen Z.:
A gpu based trafficparallel simulation module of artificial transportation systems.
In *Service Operations and Logistics, and Informatics (SOLI), 2012 IEEE International Conference on* (2012), IEEE, pp. 160–165.