



The  
University  
Of  
Sheffield.

# Complex Systems Simulations on the GPU

Dr Paul Richmond

Talk delivered by Peter Heywood

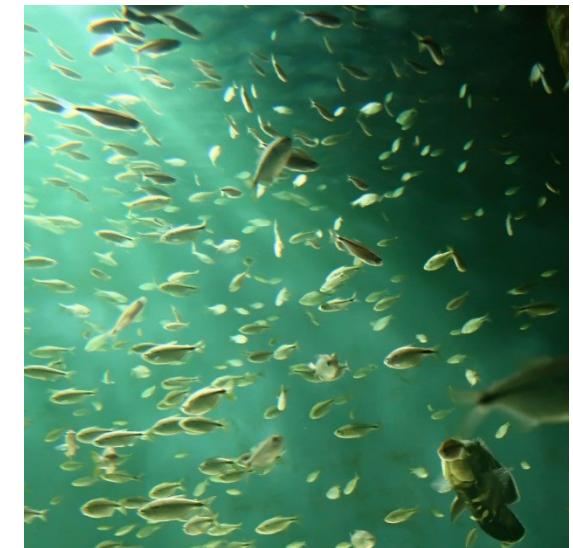
*University of Sheffield*

# Overview

- Complex Systems
- A Framework for Modelling Agents
- Benchmarking and Application Areas

# Complex Systems

- Many **individuals**
- **Interact** and behave according to simple rules
- System level behaviour **emerges**



# Agent Based Modelling

- A method for specification and simulation of a complex system
  - Model is a set of autonomous communicating agents
- Simulation helps to understand complex systems
  - Interventions and prediction
- Presents a computational challenge!
  - Especially for real time or faster



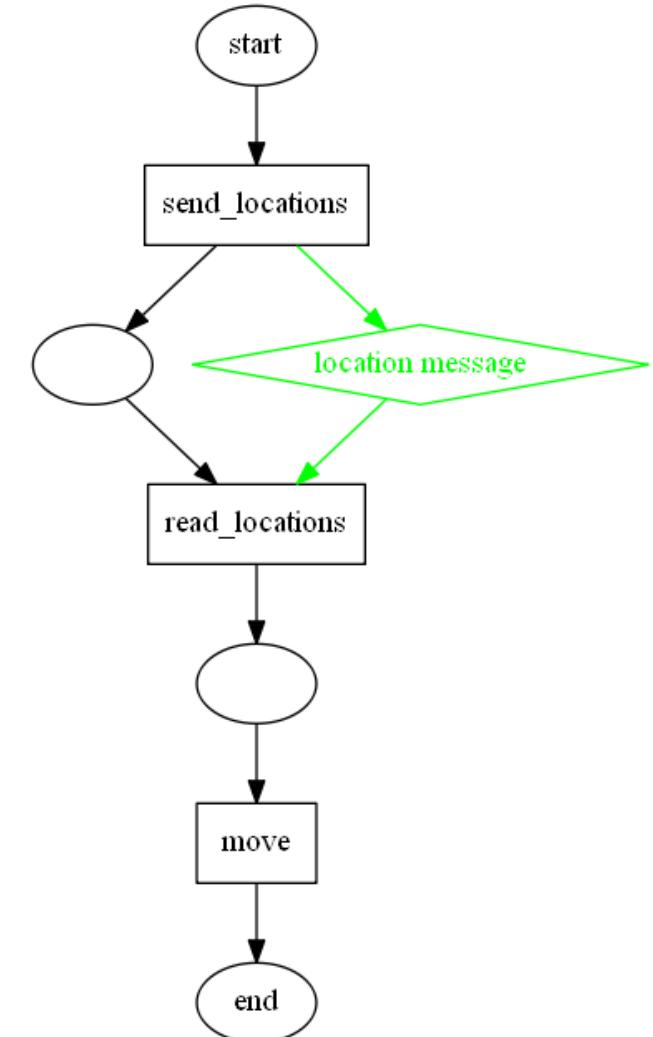
# Difficulties in Applying GPUs

- Agents are heterogeneous
  - i.e. **They diverge**
- Agents are born and agents die
  - Leads to **sparse populations** and non coalesced access
- Agents communicate
  - No global mechanism for GPU thread communication
- Agents don't stay still
  - Acceleration structures used for simulation need to be rebuilt

- Complex Systems
- A Framework for Modelling Agents
- Benchmarking and Application Areas

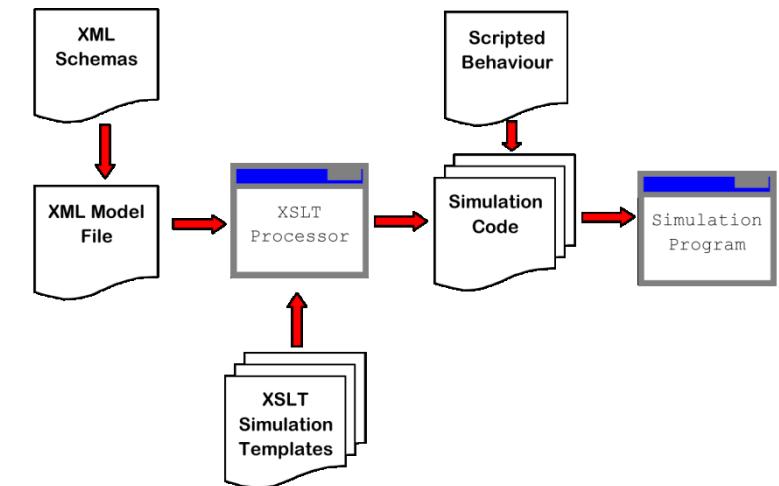
# A Formal Model of an Agent

- Abstract the underlying architecture
  - Let modellers write models not parallel programs
- Describe agents as a form of state machine (X-Machine)
  - Minimises divergence
- Describe state transition functions (agent functions) using high level script
- Describe communication as message dependencies between agent functions
  - Results in Directed Acyclic Graph
  - Identifies synchronisation points for scheduling



# FLAME GPU: A Code Generation Framework

- XML Model File
  - Describe Agents and Communication (messages) as a model in XML
- XSLT Templates
  - Code generate a simulation API from agent descriptions
- Scripted Behaviour
  - Scripted behaviour links with dynamic simulation API
- Simulation Program
  - Loads initial data and provides I/O or interactive visualisation



# Code Generation using XSLT

- Powerful technique for code generation from Declarative XML model
  - Full functional programming language

```
<xagents>
  <gpu:xagent>
    <name>Circle</name>
    <memory>
      <gpu:variable>
        <type>int</type>
        <name>id</name>
      </gpu:variable>
      <gpu:variable>
        <type>float</type>
        <name>x</name>
      </gpu:variable>
      <gpu:variable>
        <type>float</type>
        <name>y</name>
      </gpu:variable>
      <gpu:variable>
        <type>float</type>
        <name>z</name>
      </gpu:variable>
      <gpu:variable>
        <type>float</type>
        <name>fx</name>
      </gpu:variable>
      <gpu:variable>
        <type>float</type>
        <name>fy</name>
      </gpu:variable>
    </memory>
  ...
```



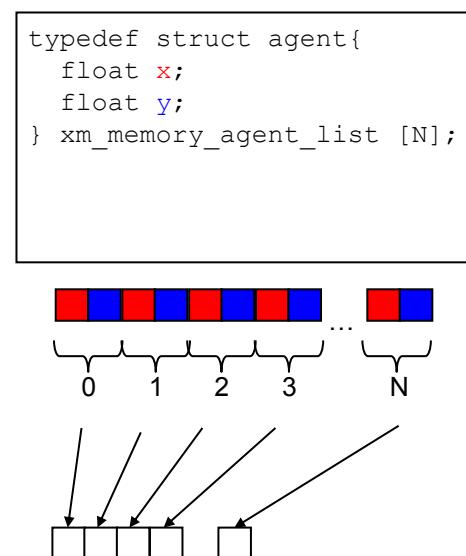
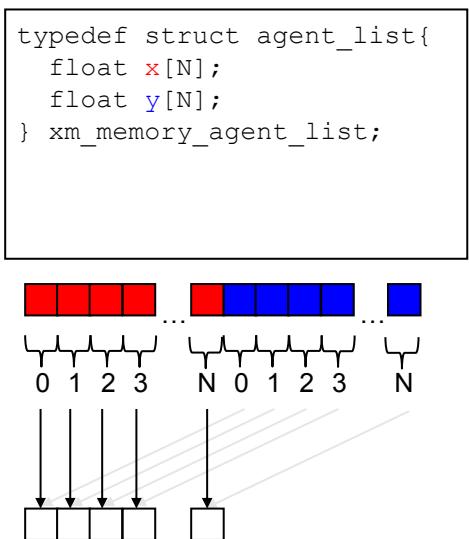
```
<xsl:for-each select="xagents/gpu:xagent">
  struct __align__(16) xmachine_memory_<xsl:value-of select="name"/>
  {<xsl:for-each select="memory/gpu:variable">
    <xsl:value-of select="type"/><xsl:text> </xsl:text><xsl:if test="arrayLength">*</xsl:if><xsl:value-of select="name"/>;
    </xsl:for-each>
  };
  </xsl:for-each>
```



```
struct __align__(16) xmachine_memory_Circle
{
  int id;
  float x;
  float y;
  float z;
  float fx;
  float fy;
};
```

# Mapping an Agent to the GPU

- Each agent function corresponds to a single GPU kernel
  - Each CUDA thread represents a single agent instance
- Agent functions use a dynamically generated API
- Agent Data is transparently loaded from Structures of arrays



```
--FLAME_GPU_FUNC__ int read_locations(
xmachine_memory_bird* xmemory,
xmachine_message_location_list* location_messages)
{
    /* Get the first message */
    xmachine_message_location* location_message =
        get_first_location_message(location_messages);

    /* Repeat until there are no more messages */
    while(location_message)
    {
        /* Process the message */
        if distance_check(xmemory, location_message)
        {
            updateSteerVelocity(xmemory, location_message);
        }

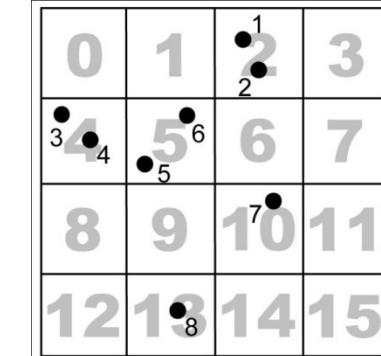
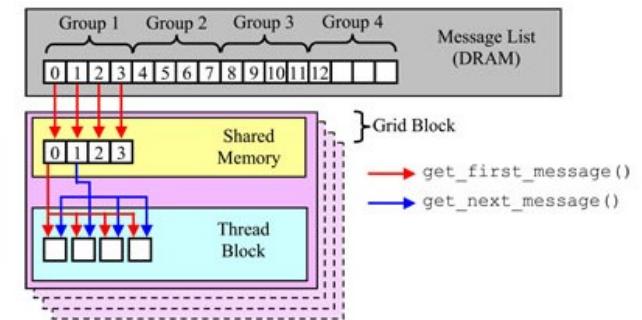
        /* Get the next message */
        location_message =
            get_next_location_message(location_message,
                location_messages);
    }

    /* Update any other xmemory variables */
    xmemory->x += xmemory->vel_x*TIME_STEP;
    ...

    return 0;
}
```

# Agent Communication

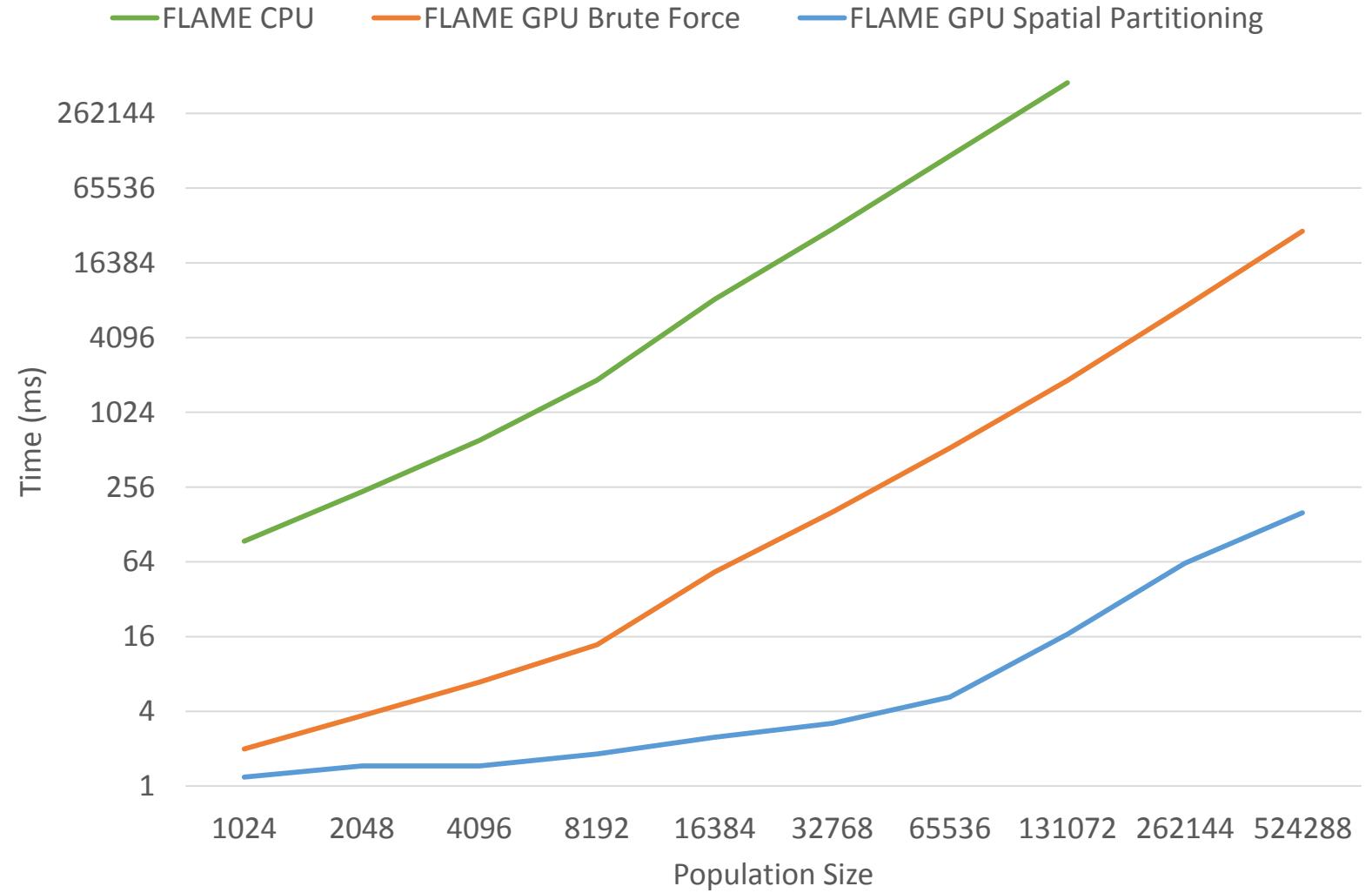
- Brute force communication
  - Tile message lists into shared memory to reduce global memory access
  - Each agent serially reads data from shared memory block
- Continuous Space Limited Range Communication
  - Sort messages according to position within partitioned space
  - Save partition boundaries
  - Agents check only neighbouring boundaries
- Discrete Space Limited Range Communication
  - Either load all messages into shared memory or use the texture cache



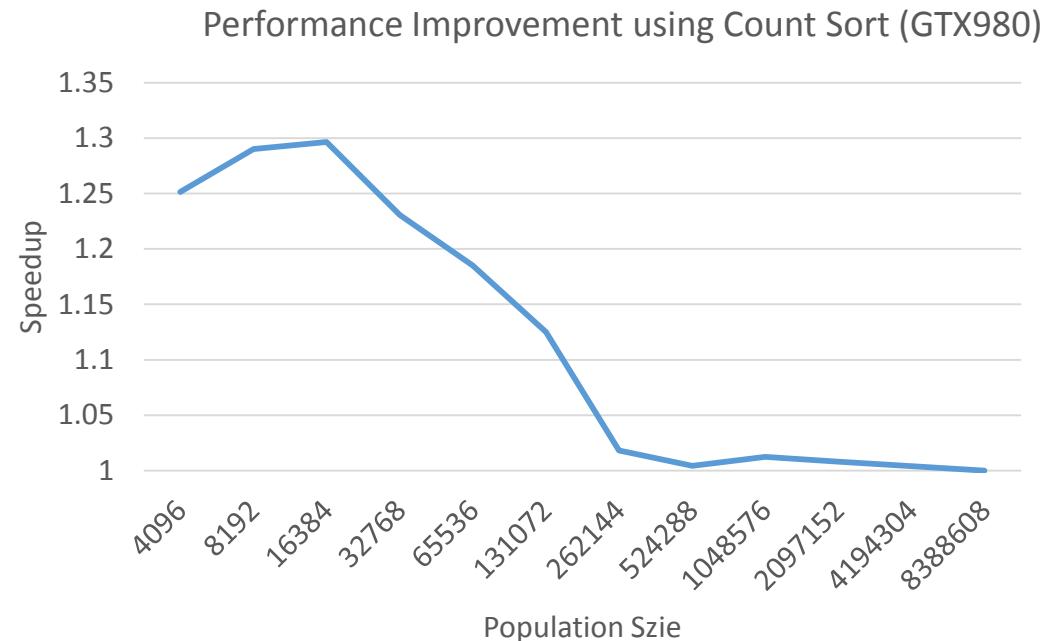
Partition	First agent	Last agent
0		
1		
2	1	2
3		
4	3	4
5	5	6
6		
7		
8		
9		
10	7	7
11		
12		
13	8	8
14		
15		

- Complex Systems
- A Framework for Modelling Agents
- Benchmarking and Application Areas

- Circles benchmark model
  - Iterative force resolution of randomly located points
- GPU results from NVIDIA K40 GPU using FLAME GPU 1.4
- 700x faster than FLAME CPU II cluster using MPI and vector instructions

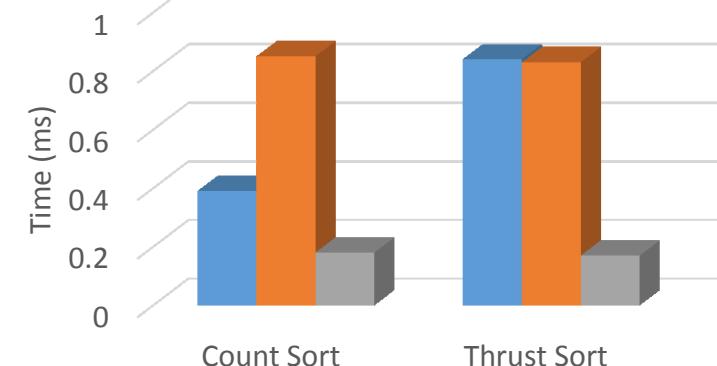


# Recent Performance Improvements

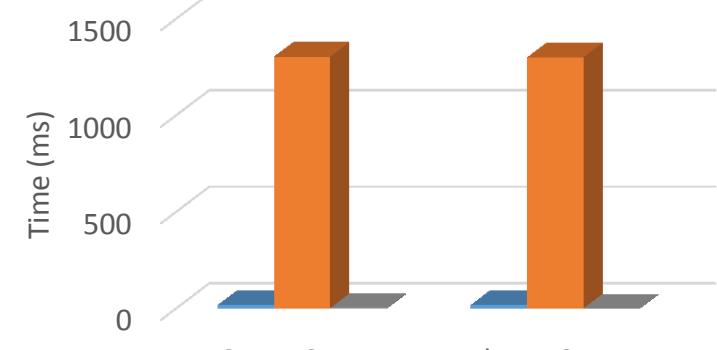


- Introduction of Counting Sort for spatial binning
- More noticeable improvement in smaller populations
  - Otherwise message reading becomes the bottleneck

Performance Breakdown for 16k agents



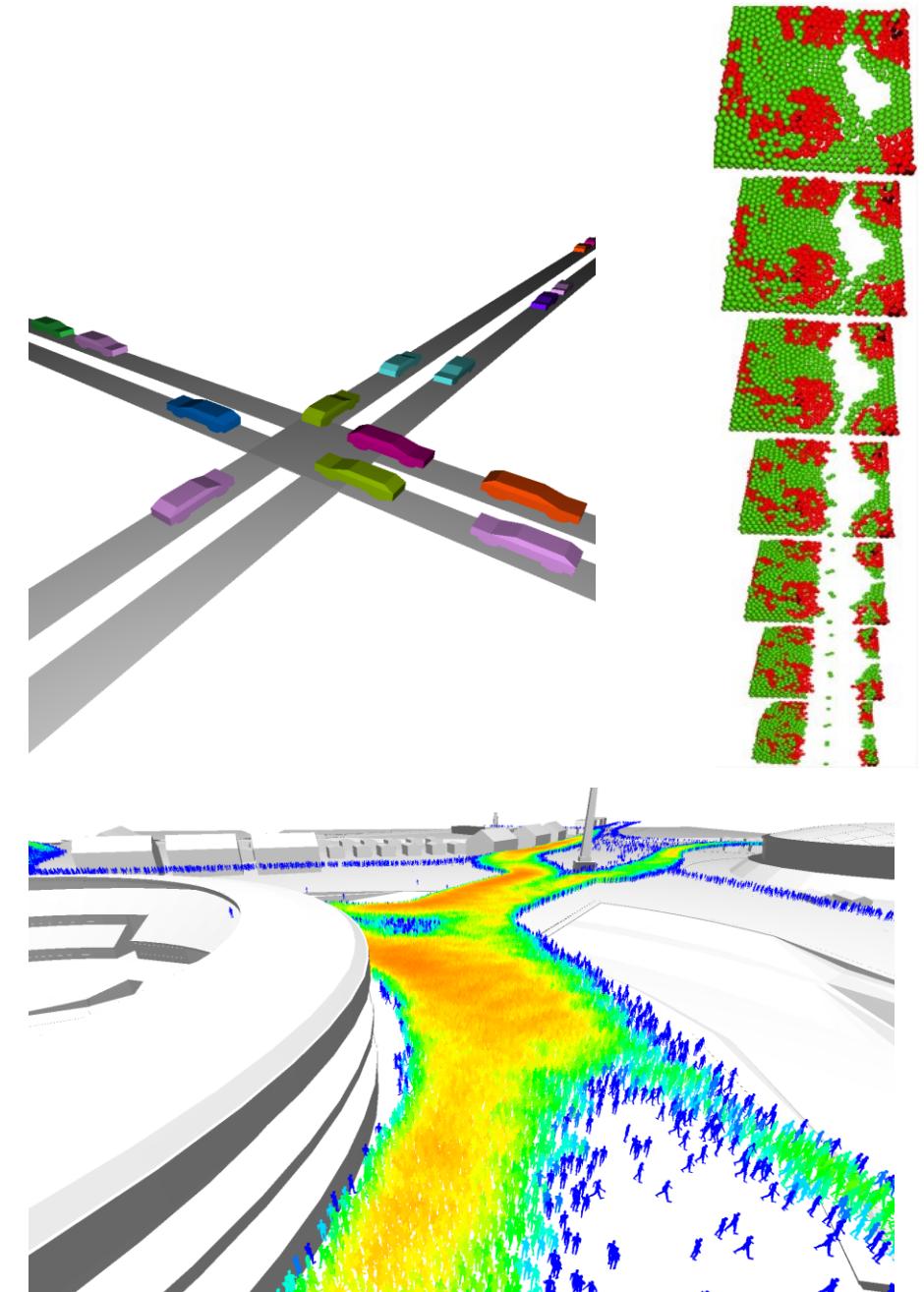
Performance Breakdown for 4M agents



█ send\_locations  
█ read\_locations  
█ move

# Application Areas

- Pedestrian dynamics
  - Social Repulsion (Social Forces)
  - Vector Fields and Navigation graphs
- Computational Biology
  - Cellular simulations (e.g. epitheliome tissue formation)
  - Molecular modelling of pathways
- Traffic and Transport Networks
  - Car following



# Conclusions

- Agent based modelling can be used to represent complex systems at differing biological scales
- FLAME GPU is a framework for model description and CUDA code generation
- Using state based representation avoids divergence and allows parallelism within a model to be exploited
- Visualisation is extremely cheap

# Thank You

Get the code for free from:

<http://www.flamegpu.com>

[www.github.com/FLAMEGPU](http://www.github.com/FLAMEGPU)



Any further questions then please contact me:

[p.richmond@sheffield.ac.uk](mailto:p.richmond@sheffield.ac.uk)

<http://www.paulrichmond.staff.shef.ac.uk>

