BAE SYSTEMS

Using GPU Frameworks

Andy Page, ATC

Spark

- This talk sparked by using ray tracing library OPTIX (NVIDIA)
- Ability to build Straw-man in no time
  - Good SDK
- Extreme Performance
  - Possible to apply design processes
- Intrigued by need to just supply Kernels
  - Simpler than pure CUDA kernels.
  - Kernels can launch further rays.
- I am a Programmer at heart
  - So don’t necessarily want to take easiest option.
- C++ support
- Got me to thinking

BAE SYSTEMS

Examples - OPTIX SDK

- Shadowing
- Collision detection
- Glass refraction

BAE SYSTEMS

Also aware of Colleagues work

- GPU Flame – Particle (Agent) interactor
  - Even simpler – supply kernel partly in code and in text form
  - Particles in 2D cells.
- Colleague asking NVIDIA about volume ray tracing
  - Streamlines
  - Plume propagation
- Our in-house Smoothed Particle Hydrodynamic (SPH) code
  - Ask my self is this crying out for a similar Framework
  - Supporting particles in 3D cells
  - Local interactions trigger kernel.

BAE SYSTEMS

Definition & Span

- GPU Framework as far as this talk is concerned
  - Calls user supplied kernel for interaction.
  - I can specify state I require
  - Geometry is split in any fashion it thinks is sensible

- Span
  - Work carried out in a Physics modelling tool using OPTIX
  - End user of FLAME GPU
  - Looking ahead thinking what frameworks could be useful.

BAE SYSTEMS

Our Groups' Bread & Butter

- Historically HPC cell based simulation codes
  - Air flow over wings
  - Water flow
- HPC to GPU
  - We have bitten the bullet (pity PHI)
  - Kernels and independent loops
  - Proxy data that might reside CPU or GPU
  - Our own GPU cluster: 8 servers, 32 Fermi cards
- **However**, will there be a Mesh GPU Framework where you supply
  - Cell and Face kernels only.
- OP2 maybe be candidate
  - Oxford (Mike Giles) : http://www.oerc.ox.ac.uk/research/op2
- VTK a mesh framework
  - Supports some parallel processing.

A Ray Tracing tool

Based on OPTIX

Solving Physics important to the company

BAE SYSTEMS

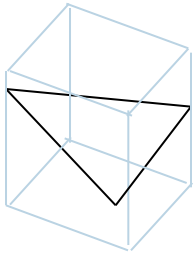
Physics important to the company

- Already invested in ray tracing tools
- Rays bounce statistics on fixed and moving geometries
- Infrared signature
- Lower Frequency Electromagnetic propagation
  - Radar
  - Wifi
- Convinced our partners that OPTIX can give a step change.

BAE SYSTEMS

OPTIX – Geometry

- Geometry agnostic
- Rather for each primitive you need to return a Bounding box.



Tri face Bounding Box

- Groups of elements can belong to different materials which have totally different colouring/physics routines

BAE SYSTEMS

OPTIX - State

- All state is stored in buffers that are mapped to the GPU
  - Automatically
  - And back again if need be.
- Kernel sees these as global multi dimensional arrays.

rtBuffer< float , 2 > out

Context[“out”]

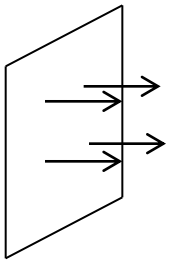
- C++ side links to these buffers using
  - A map accessed with strings
  - An issue but there is a very good validator.

- This state will include Geometry

BAE SYSTEMS

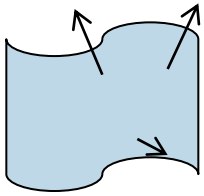
OPTIX – Ray Launch

- Rays are fired from 1 or 2d arrays, with associated camera model which can be
  - A image plane
  - Random positions over a geometry
  - Anything i.e. not tied to solving Image generation problems.



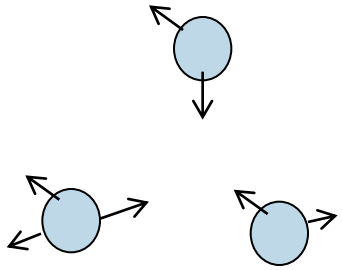
Orthogonal Camera

or



Rays randomly launched from Surface

or



Collison example earlier

BAE SYSTEMS

OPTIX - Intersection

- A launched Ray is tested against the bounding boxes.
- If it enters a bounding box, the intersection kernel is triggered
  - User supplied
  - If you indicate it intersects, OPTIX launches the relevant Colouring Kernel

BAE SYSTEMS

OPTIX - Colouring

- User supplied kernel
- Colouring can involve launching further rays

BAE SYSTEMS

OPTIX - Conclusions

- Extremely fast: 100 M rays a second
  - You could not code this yourself!
  - Vendor optimized
- Very flexible - Bug free
- Kernels are not complicated by concept of Warps
- Simple memory model
- SDK contains lots of examples
- Issues
  - We would rather have less rays per seconds and more ray depth
  - Complexity of Kernels limited.

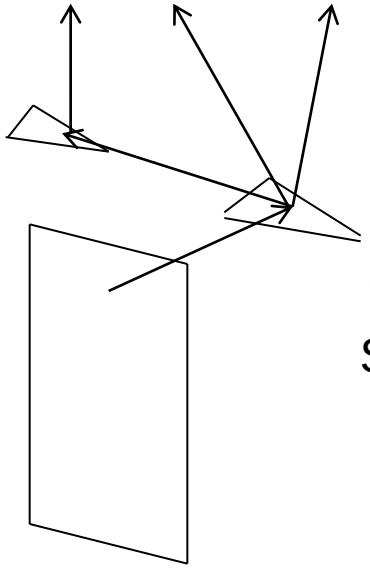
Application example: Infrared

- Signatures Important

- To build Temperature Image
- Fire rays from viewed surfaces
  - In diffuse and specular fashion
  - To any depth
  - Collect temperature from hit surfaces
  - Collection Cut-off
  - Concave regions get hotter

- We can upgrade tools with 60 X

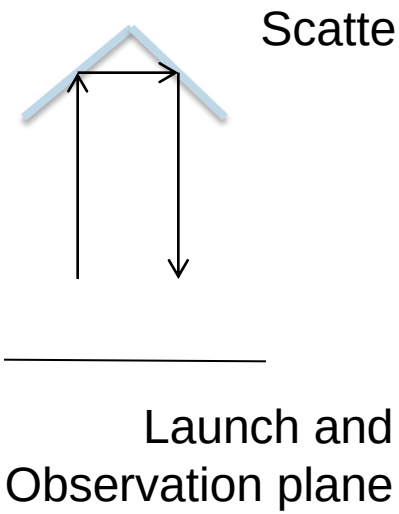
- Starting to think about real time analysis



BAE SYSTEMS

Application example: Electromagnetic Propagation

- Shooting Bouncing Ray method => EM Signature
  - Integrate rays hitting observation plane
  - Take into account absorption along the way.



  - In process of updating existsing tools

BAE SYSTEMS

Modelling Agents
Using Flame GPU

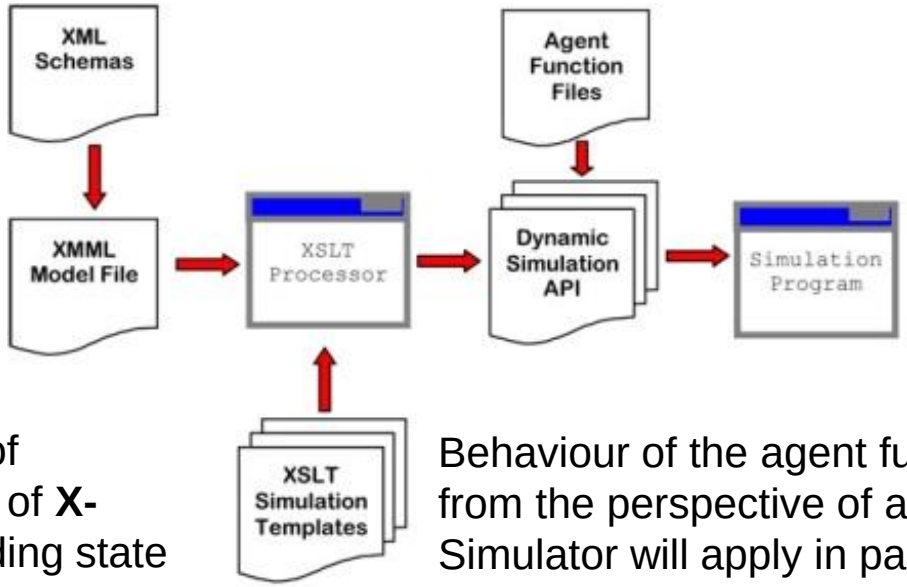
BAE SYSTEMS

Flame GPU

- FlameGPU
  - Is a data parallel implementation of FLAME using CUDA
  - Consider divergence
  - Is cost effective solution for high performance ABM simulation
  - Offers real time visualisation

- FLAME
  - Is a Flexible Large-scale Agent Modelling Environment
  - Uses **XML** Model specification based on agents as **X-Machines** (extended state machine)
  - Is a template system for generating simulation code

FLAME GPU – High level Overview



XMML model file consists of

- a definition of a number of **X-Machine agents** (including state and memory information and a set of agent transition functions)
- a number of **message** types (each of which has a globally accessible message list)
- a set of simulation layers and define the execution order of agent functions for a single simulation iteration

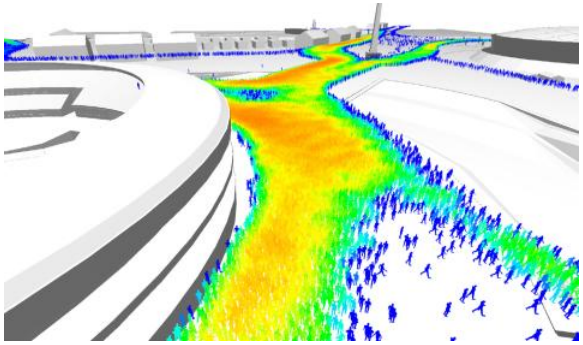
Behaviour of the agent function is described from the perspective of a **single agent**. Simulator will apply in parallel the same agent function code to each agent which is in the correct start state.

Agent function scripts are defined using a simple C based syntax with the agent function definitions, and more specifically the function arguments dependant on the XMML function definition.

BAE SYSTEMS

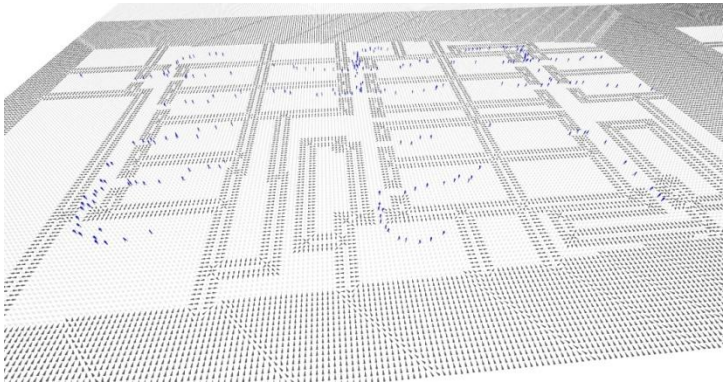
Agent Based Modelling (of multi agent systems)

- Agents are **individuals** with their own memory and perception
- Agents change their behaviour depending on **interactions** with neighbours
- Distance filters used to specify neighbours.
- Complex system level behaviour naturally **emerges** as a result of many interacting individuals



Agent Based modelling: applications

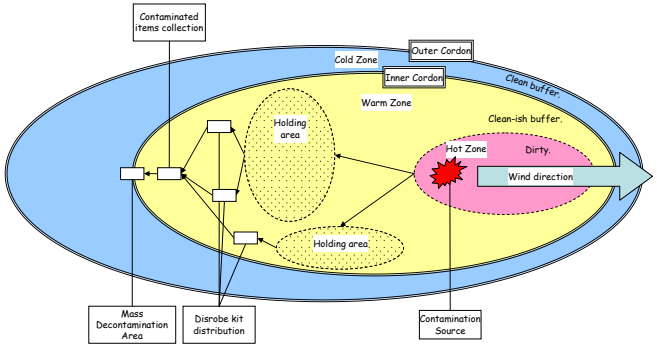
- Maritime

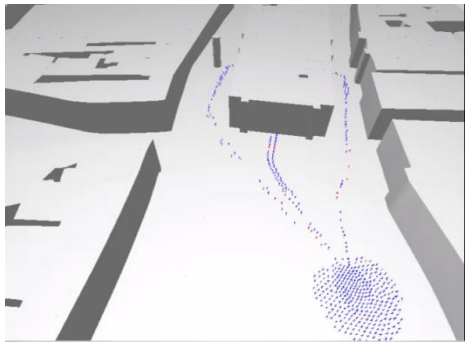


Live Traffic Modelling

- Civil: CBRN Incident

Aim: Modelling of large numbers of persons who must move towards and through decontamination structures as a results of an incident. All the configurations are run simultaneously, faster than real time and ranked to provide meaningful outputs to help in the decision making process.





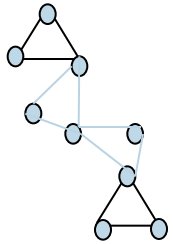
BAE SYSTEMS

Mesh GPU Frameworks

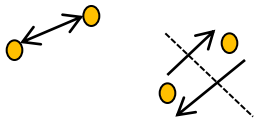
BAE SYSTEMS

Our experience – returning to our Bread and Butter

- Need Kernels to be independent
  - Colour elements
    - Implies writes to points are independent and safe
    - Downside elements in a colour group might be small



  - Do twice as much work
    - In SPH this corresponds to every particle looking at all neighbours
    - Even though computation is symmetrical.



- Programmed both these schemes and have running on our own Hardware.
- OPLUS2 alternative
  - Keen to find time to contrast solutions and performances.

BAE SYSTEMS

Conclusions

- OPTIX performance made it easy for us to adopt
  - Vendor tuned to Hardware
- GPU Flame
  - Versatile, cost effective and allows real time simulation for planning and in the field live operation
- Like to stimulate people to think about SPH framework
  - combination FLAME and OPTIX
- Watching brief on OP2