A large scale discrete element framework for NVIDIA GPUs.

Nicolin Govender, Daniel Wilke, Schalk Kok

Govender.nicolin@gmail.com

CUDA™ RESEARCH CENTER

GTC 2015

University of Pretoria

UNIVERSITY OF JOHANNESBURG

GTC 2015

CSIR our future through science

Outline

- Particle Transport
- Discrete Element Method
- Physical Interaction
- BLAZE-DEM Framework
- Performance
- Conclusion

Particle Transport(1)

- Simulation of particle transport processes are required in many areas of research:
  - Elementary particles.
  - Nuclear particles
  - Molecular dynamics.
  - Dry chemical powders.
  - Granular media .
  - Natural phenomena.

Forces: Electromagnetic / Atomic/ Molecular.

BLAZE-DEM

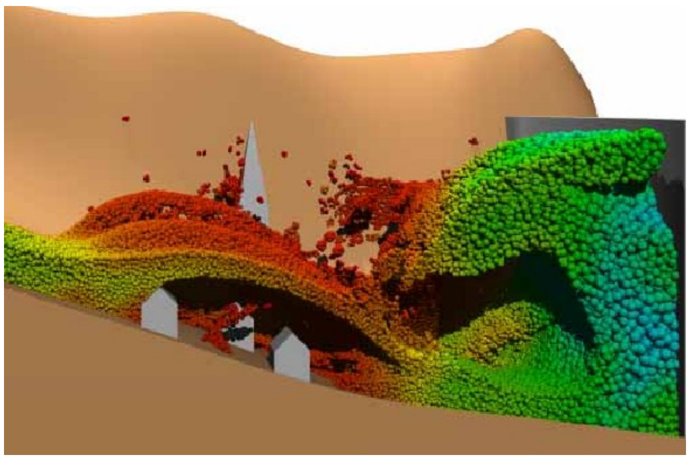
Forces: Gravitational/ Mechanical/Cohesion/electrostatic .

Particle Transport (2)

Particle Transport(3)

Two descriptions of particle transport:

Discrete



Continuum

Pictures: Simon Green (NVIDIA 2008)

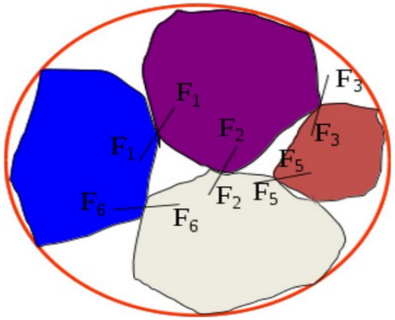
- Discrete is *physically correct* but *computationally expensive*.

- Continuum methods requires solution of a transport equation which describes system evolution. eg Navier-stokes (CFD).

Particle Transport(4)

- Discrete solutions most often can provide a solution by direction simulation of physics.
- The phase-space/trajectory of a particle is simulated in accordance with physical laws.
- Doesn’t require coupling of a system, physics simulated at each point.
- Since individual particles are simulated, well suited to parallel implementations.

Discrete Element Method

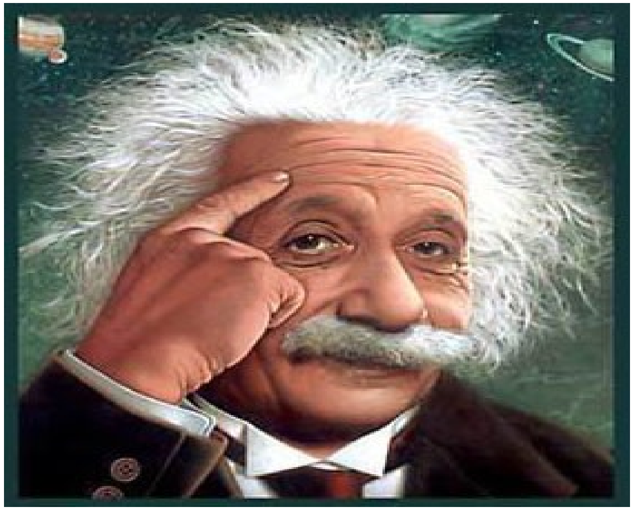


- Most popular and successful approach first described by "*CUNDALL : A discrete numerical model for granular assemblies. Geotechnique 29, (1979), 47–65.*"
- Particles most commonly treated as spheres.
- Motion of particle dependent on net sum of forces per time step.
- Similar forces and particle sizes.
- Binary Contact.
- Explicit integration.
- Embarrassingly parallel.

Physical Interaction(1)

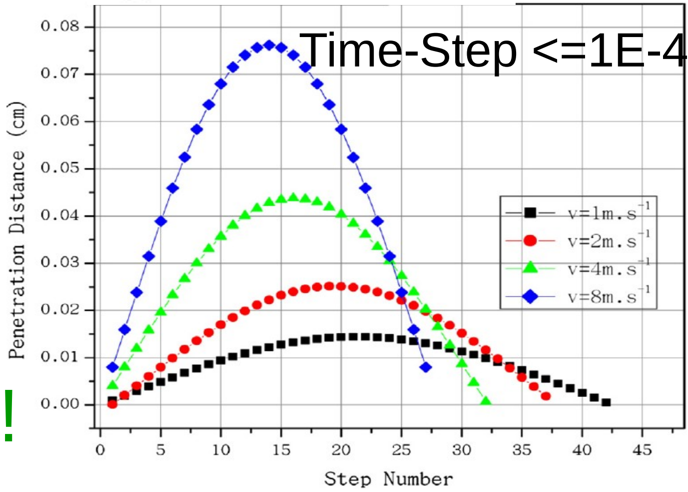
- After finding all contacting particles we need to determine their physical interaction.

- This is where gaming simulations diverges from physics.



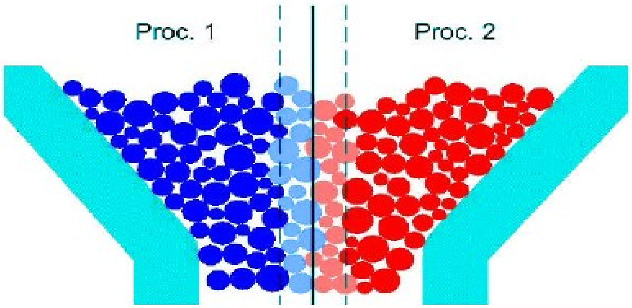
Physical Interaction(2)

- Gaming approximates contact duration crudely for impulse calculations. $\mathbf{v}^{new} = \mathbf{v}^{n} \pm \mathbf{j}/m, \quad \omega^{new} = \omega^{n} \pm \mathbf{I}^{-1}(\mathbf{r} \times \mathbf{j})$
- Physics simulations resolves the contact duration from constitutive contact models. $\mathbf{F}_{\mathrm{N}}^{\mathrm{elastic}} = (\mathrm{K_r}\delta^{\frac{3}{2}})\mathbf{n}$ $\mathbf{F}_{\mathrm{N}}^{\mathrm{diss}} = -\mathrm{K_D}\delta^{\gamma}\mathbf{vrel}_{\mathrm{n}}$



- Simple integration such as Euler.
- Velocity Verlet integration (2nd O)
- Contact is resolved in a single time-step!
- Contact is resolved over multiple steps!
- Gaming is qualitative and estimates visual acceptable behavior
- Physics simulations are quantitative and estimates physical quantities such as energy, impact and shear and normal forces.

Parallel computing in DEM



Parallel CPU: 3.0 GHZ x 12 cores
Intel® Xeon® Processor E7-8857
Cost: $3838
Power:130W

GPU: 1.0 GHZ x 26 SM 53284 threads
NVIDIA® Tesla® K80
Cost: $5000
Power:300W

Computation cost for 10 million particles ?

12 sub-domains = **83333** particles\core.

Each core will loop over 83333 particles in serial.

Thread scheduling is done automatically so we **launch 10 million threads** on the GPU.

CPU does 3X more computations per given cycle than the GPU.
Suppose it takes one second for a cycle.

The **CPU** will require **83333/3 = 27777 seconds for 10 million particles.**

GPU can execute **53284 parallel threads per cycle**. The **GPU** will require **1000000/53284= 18.76 seconds for 10 million particles.**

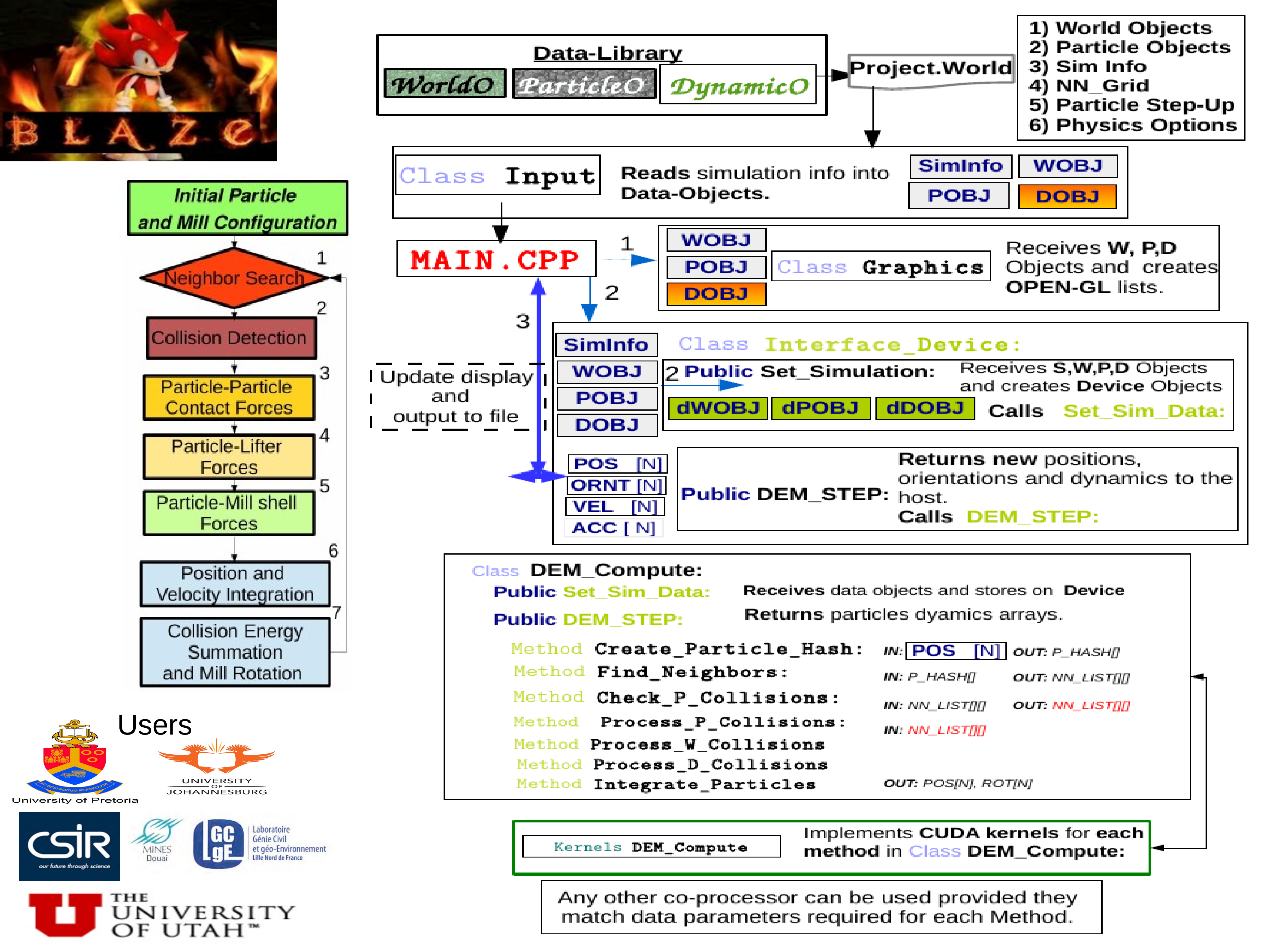
GPU is 1480X faster than the parallel CPU. Cost 1.3X , Power 2.3X.
So real-life gain is about 500X

Challenges

- *Discrete methods are computationally expensive thus limited in use.*

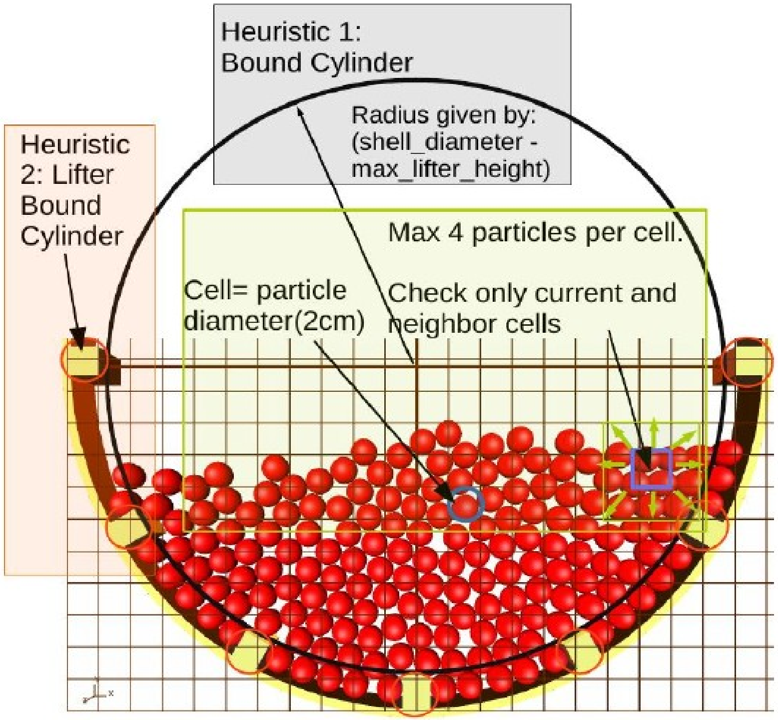
- *Approximations to make them more feasible only valid in few situations, generally not robust enough.*

- *Current Parallel implementations, require expensive clusters and software .*

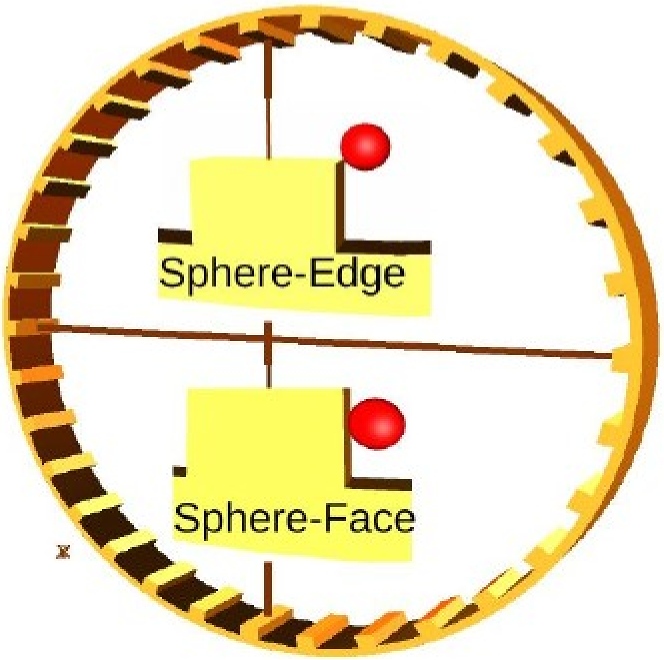

BLAZE
Initial Particle and Mill Configuration
Neighbor Search
1
Collision Detection
2
Particle-Particle Contact Forces
3
Particle-Lifter Forces
4
Particle-Mill shell Forces
5
Position and Velocity Integration
6
Collision Energy Summation and Mill Rotation
7
Users
University of Pretoria
UNIVERSITY OF JOHANNESBURG
CSIR our future through science
MINES Douai
Laboratoire Génie Civil et géo-Environnement Lille Nord de France
THE UNIVERSITY OF UTAH
Data-Library
WorldO
ParticleO
DynamicO
Project.World
1) World Objects
2) Particle Objects
3) Sim Info
4) NN_Grid
5) Particle Step-Up
6) Physics Options
Class Input
Reads simulation info into Data-Objects.
SimInfo
WOBJ
POBJ
DOBJ
MAIN.CPP
1
2
3
WOBJ
POBJ
DOBJ
Class Graphics
Receives W, P,D Objects and creates OPEN-GL lists.
Update display and output to file
SimInfo
WOBJ
POBJ
DOBJ
Class Interface_Device:
2 Public Set_Simulation:
Receives S,W,P,D Objects and creates Device Objects
dWOBJ
dPOBJ
dDOBJ
Calls Set_Sim_Data:
POS [N]
ORNT [N]
VEL [N]
ACC [N]
Public DEM_STEP:
Returns new positions, orientations and dynamics to the host.
Calls DEM_STEP:
Class DEM_Compute:
Public Set_Sim_Data:
Receives data objects and stores on Device
Public DEM_STEP:
Returns particles dyamics arrays.
Method Create_Particle_Hash:
IN: POS [N]
OUT: P_HASH[]
Method Find_Neighbors:
IN: P_HASH[]
OUT: NN_LIST[][]
Method Check_P_Collisions:
IN: NN_LIST[][]
OUT: NN_LIST[][]
Method Process_P_Collisions:
IN: NN_LIST[][]
Method Process_W_Collisions
Method Process_D_Collisions
Method Integrate_Particles
OUT: POS[N], ROT[N]
Kernels DEM_Compute
Implements CUDA kernels for each method in Class DEM_Compute:
Any other co-processor can be used provided they match data parameters required for each Method.

Collision Detection (1)

Broad Phase



Narrow Phase

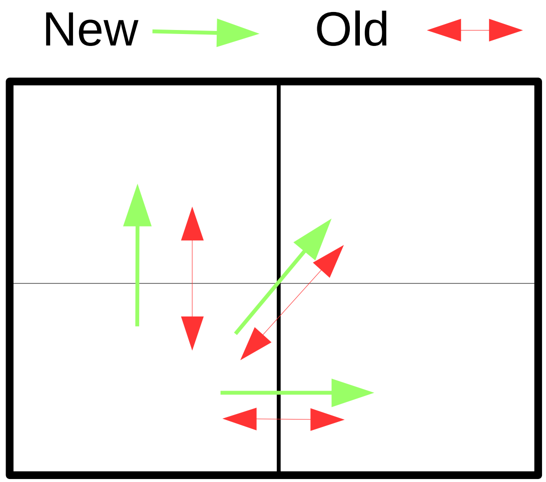


Collision detection between particles and boundaries takes ~90% of simulation time.

Collision Detection (2)

- Multi-Phase approach for code flexibility and performance.

- Spatial decomposition to search for Nearest Neighbors (NN).
  - Each particle gets a grid position based on location of COM. Stored as a hash based on spatial location.
  - Similar sized particles (1/4) ratio so can use a single grid based on largest size. (problem specific).
  - In other GPU simulations each particle checks its 27 neighboring cells for potential NN particles (Sphere test). Could not exploit symmetry on the GPU.

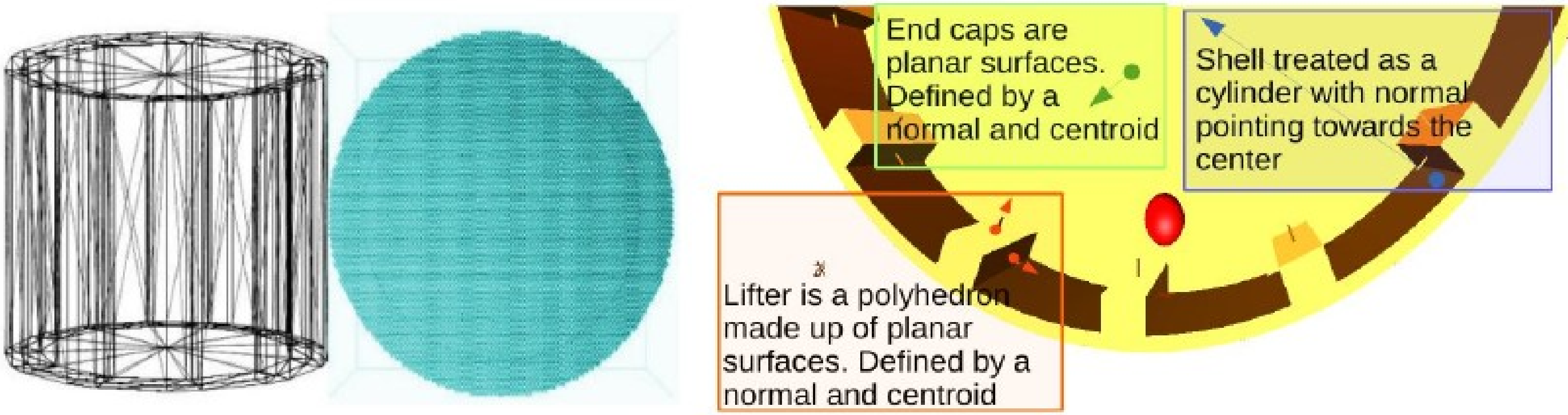


Thus N Checks are required not 2N. We do te same amount of computations as typical CPU implementations.

We also use symmetry for force updates (atomic operations). Total speed up 40% so memory over head is only 10%.

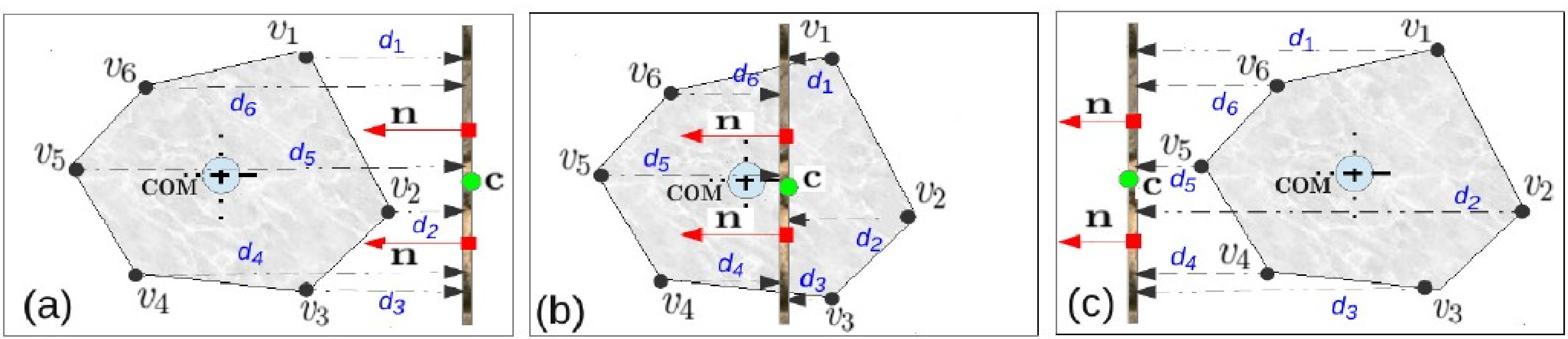
Collision Detection (3)

- Current methods use triangulation/particles which requires thousands of checks to determine collision.
- We use ray-tracing which does not require a mesh and is very efficient on the GPU

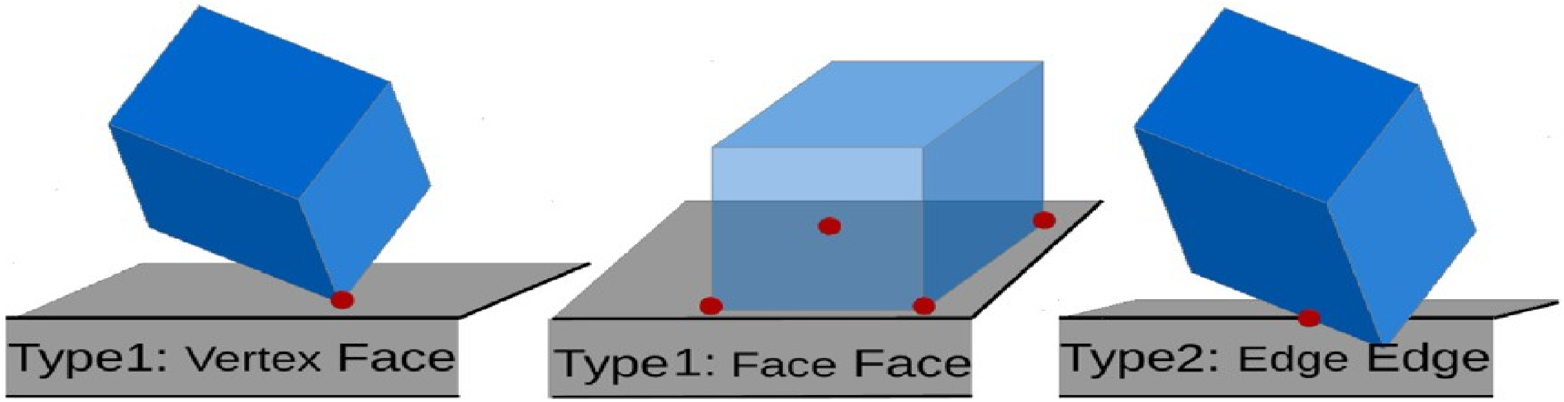


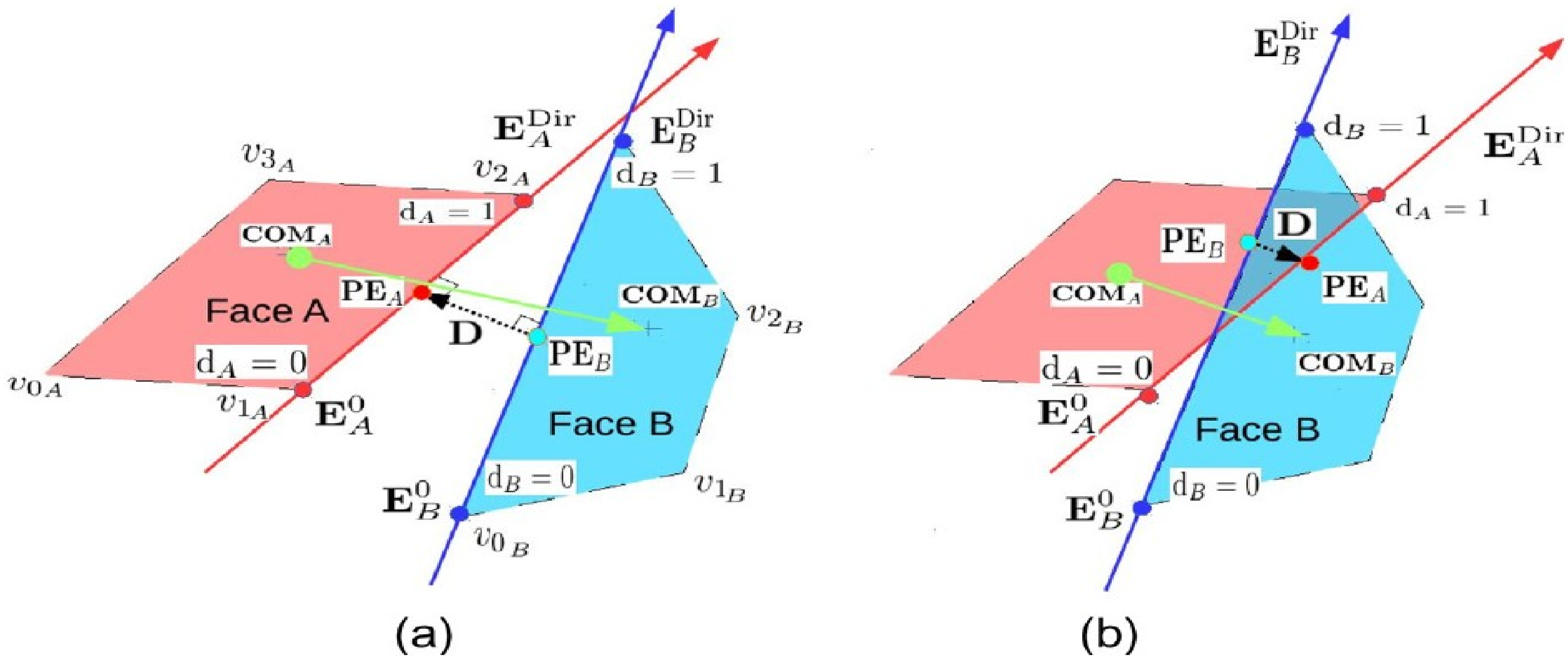
(a) (b)

$$d = \mathbf{n} \cdot (\mathbf{v} - \mathbf{c})$$



Collision Detection (4)



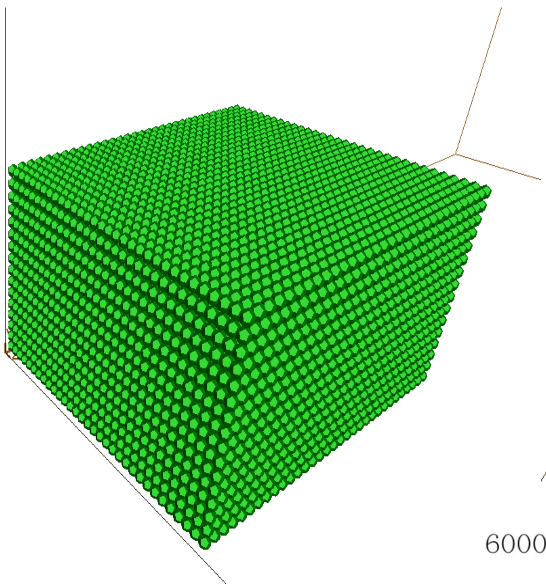


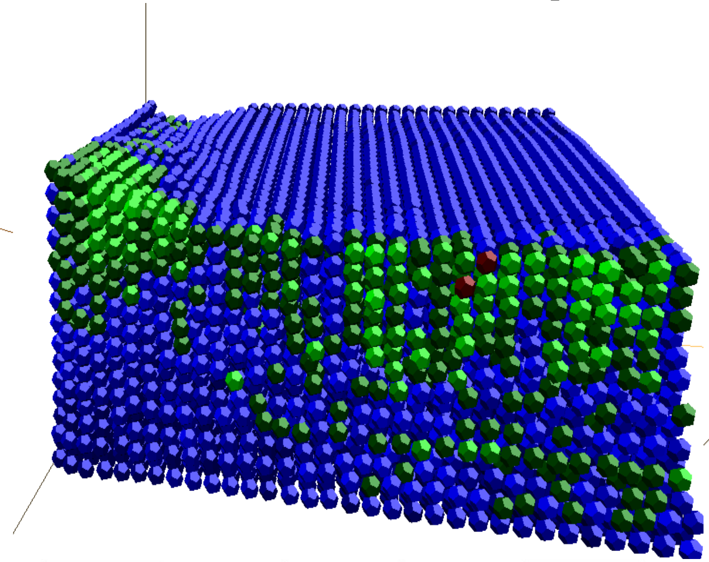
$$\mathbf{PE}_A = \mathbf{E}_A^0 + \mathrm{d}_A \mathbf{E}_A^{\mathrm{Dir}}$$

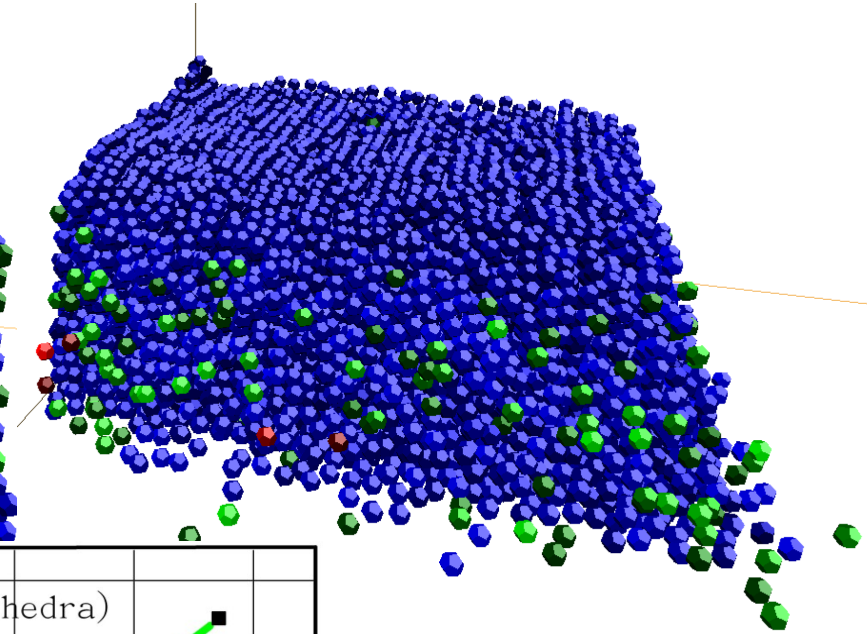
(a) Non penetrating Type 2 contact, (b) Penetrating Type 2 contact.

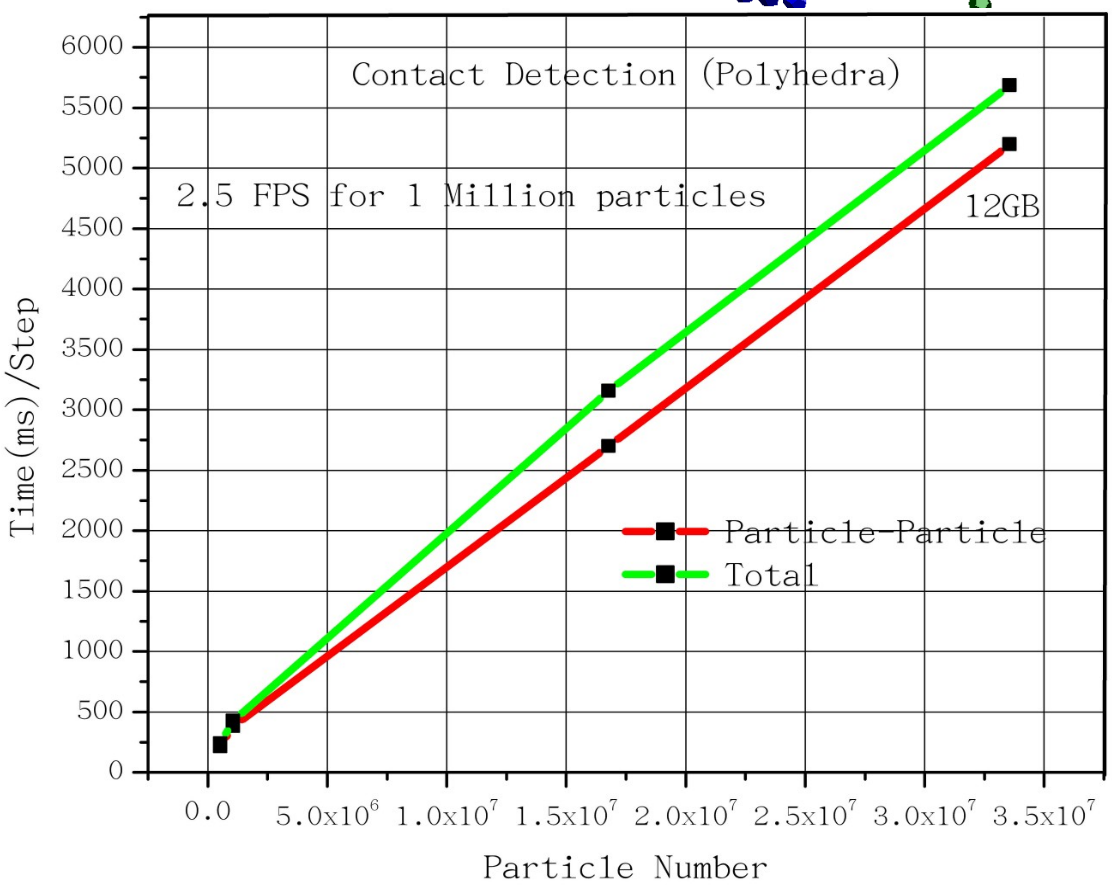
$$\mathbf{PE}_B = \mathbf{E}_B^0 + \mathrm{d}_B \mathbf{E}_B^{\mathrm{Dir}}$$

Performance (Polyhedra)

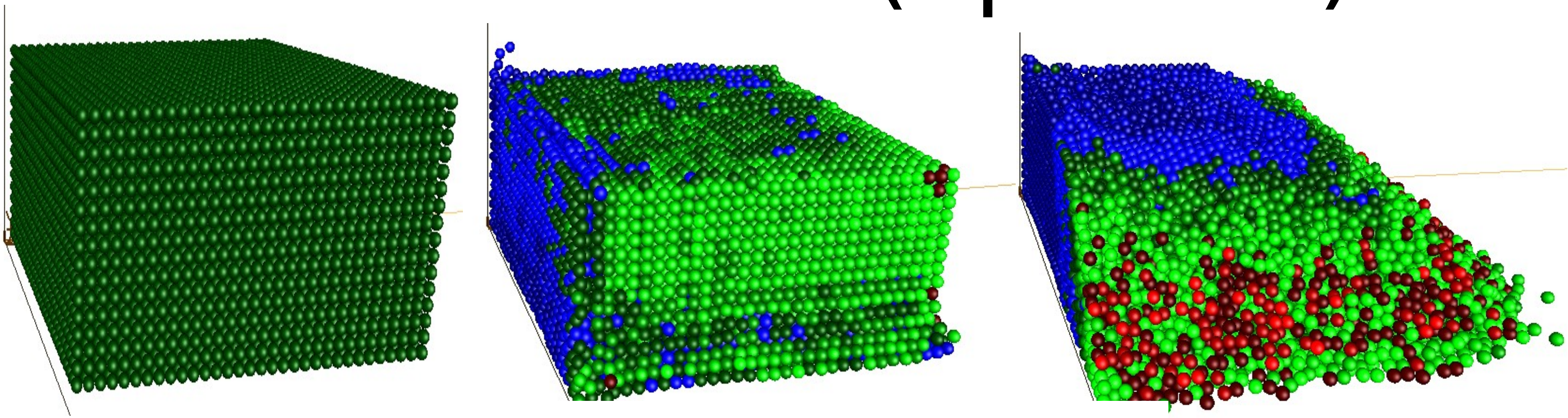


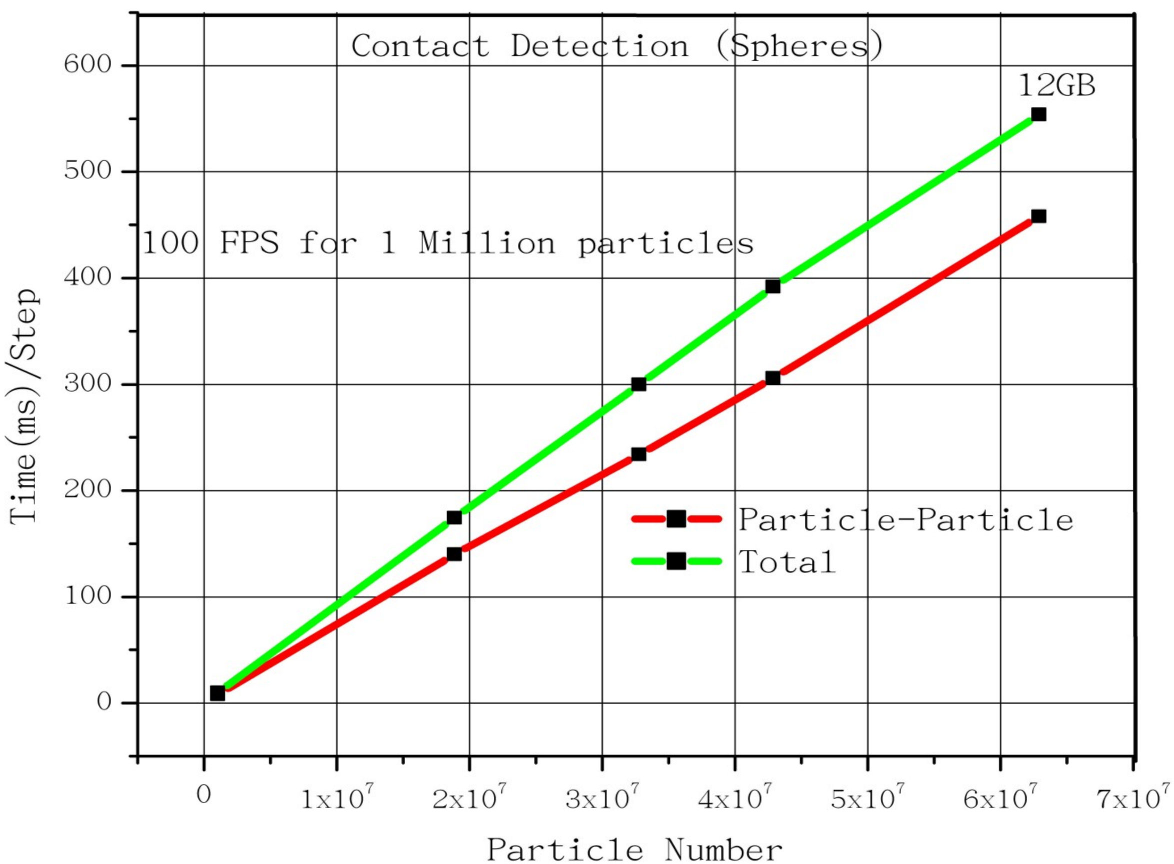






Performance (Spheres)





Performance vs Others

Author	Shape	Physics Fidelity	N particles	C Number
Harida et.al (2008 gpu gems)	Clumped	Low	65536	2.0×10^6
Longmore et.al (2013 Jpowder tech)	Clumped	High	256000	1.49×10^6
XPS (2015 GTC Poster)	Sphere	High	20×10^6	20×10^6
Nivida SDK (2014)	Sphere	Low	2.50×10^5	125×10^6
BLAZE-DEM (2014)	Sphere	High	60×10^6	100×10^6

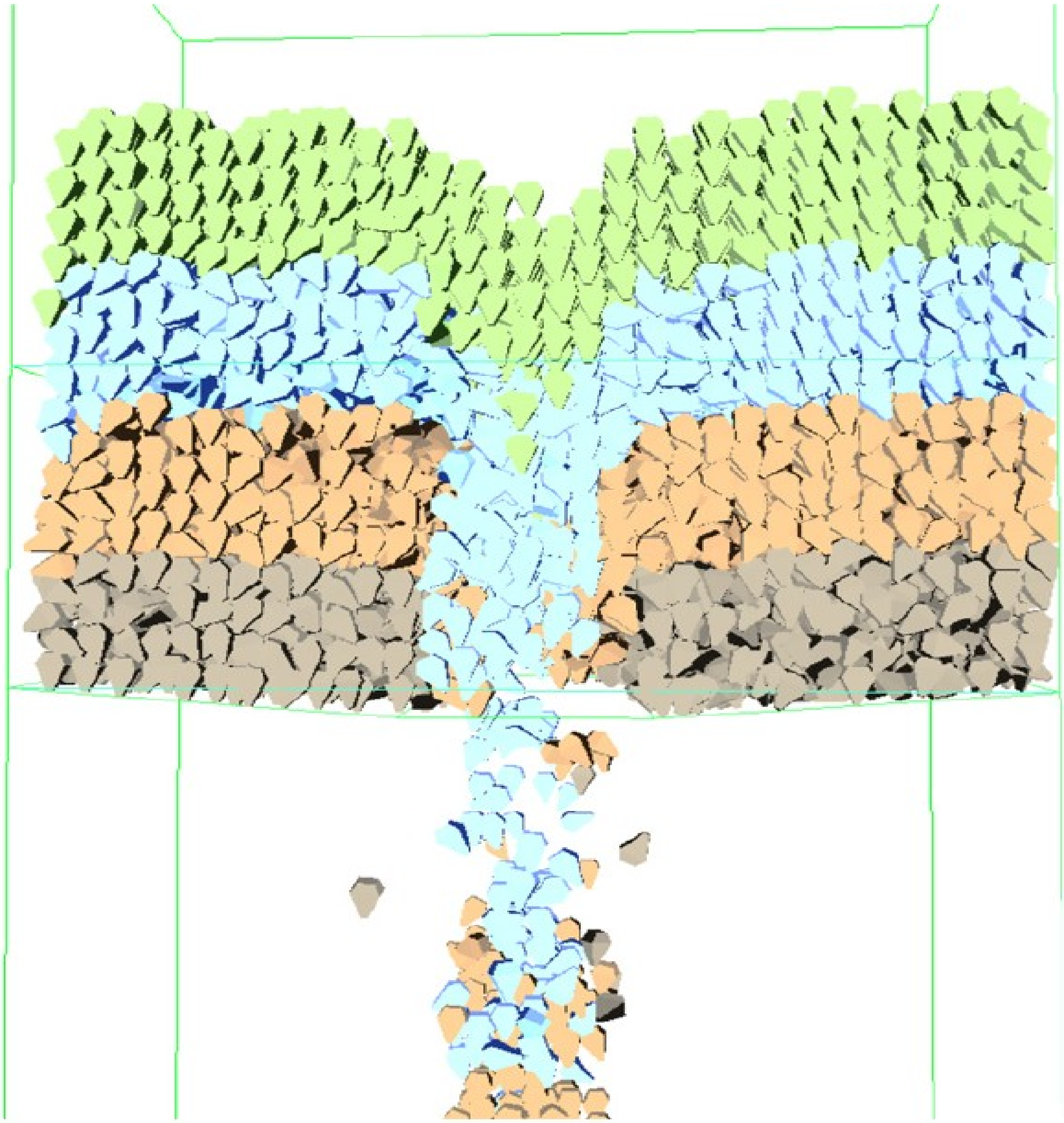
5X Faster than GPU DEM , 25% Slower than gaming simulations.

Author	Shape	Physics Fidelity	Max particles	(Time N=5×10^5)
BLOCKS (2014, PhD thesis U Illinois)	***Poly**	Highest	5000	186 days
iDEM (2014, PhD thesis U Illinois)	***Poly**	Low	500000	2.8 days
BLAZE-DEM (2014)	**Poly**	High	32×10^6	28 min

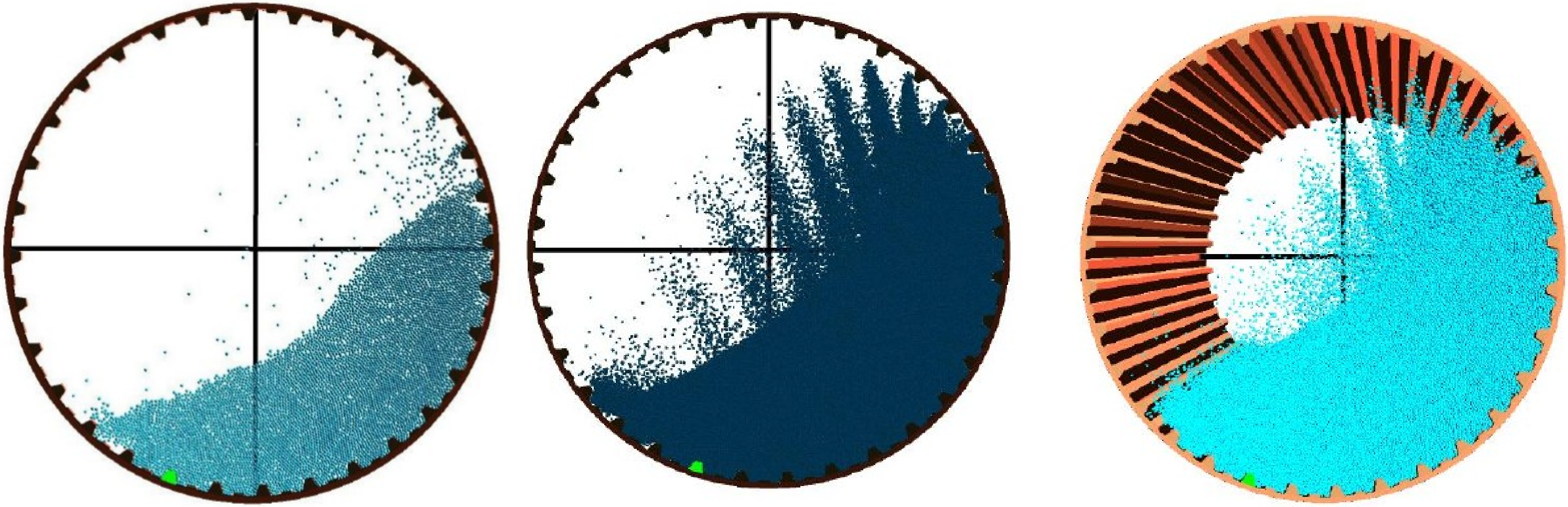
9000X Faster than DEM CPU, 144X Faster than impulse DEM

*=CPU CODE

Why is shape important



Why do we need more particles ?



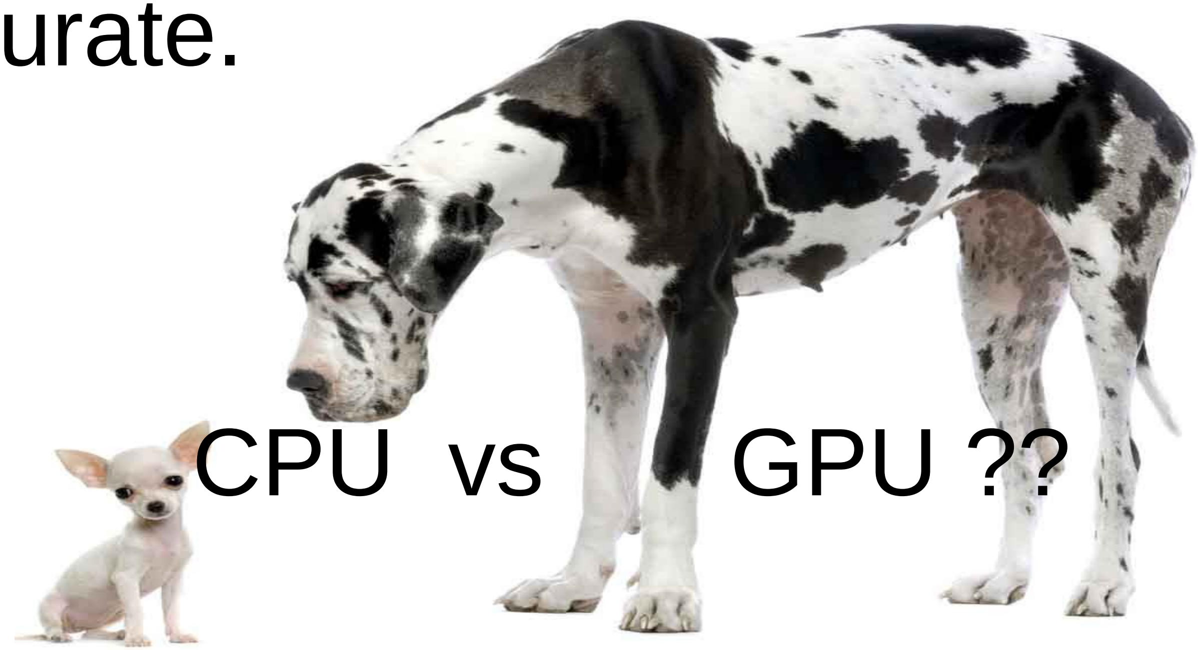
(a) (b) (c)

Conclusions

- 5X Faster than current physics GPU codes.

- 60 million spheres, 34 million polyhedra on K40 (12GB).

- Physically accurate.
-



- CPU vs GPU ??

Acknowledgments

- NVIDIA for generous Hardware donations (www.nvidia.com/cuda) .

Universities of Johannesburg and Pretoria for financial contributions.

- More Details: https://research.nvidia.com/content/university-johannesburg-crc-summary
- BLAZE-DEM will be hosted in the near future on github