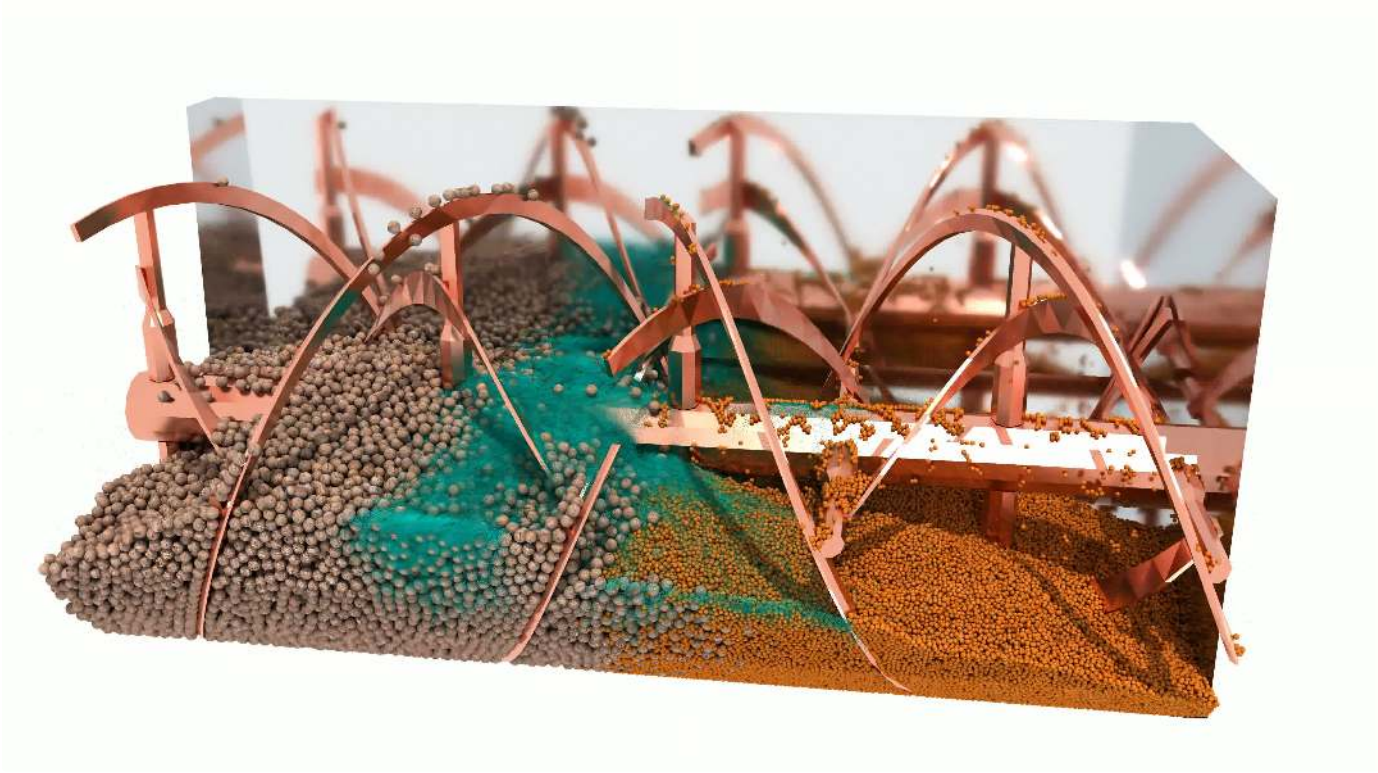
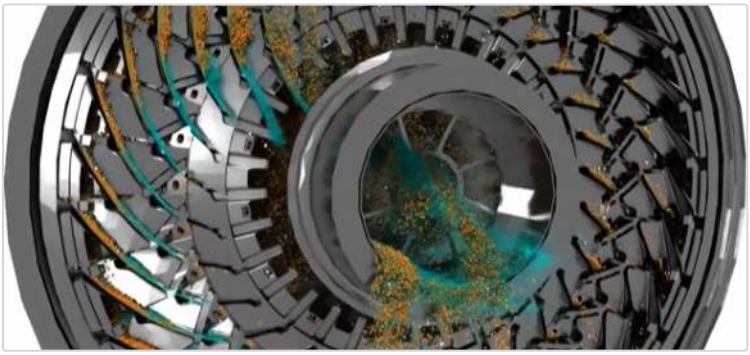
From atoms to particles, solids and fluids.

True convex polyhedra with volume based contact, SPH fluids and melts, bonded Elements that crack and CFD coupling, all native on the GPU. Models train on the live simulation through our CUDA bridge, and the solver physics self heals their predictions.

Ground truth for Physical AI.





Ore and slurry in a SAG mill pulp lifter.



Lath polyhedra in a twin screw feeder.



Polyhedral particles in an OpenFOAM pipe bend.

Atoms (atomic)	Particles (meso)	Solids (continuum)	Fluids (continuum)
Crystal properties computed with machine learned potentials.	Grains, powders and ore as true polyhedra.	Bonded Elements that deform and crack.	SPH and CFD coupled both ways.

One solver, every physics

Particles, fluids, solids and coupling run natively on the GPU in one solver, on Linux and Windows, from Turing (RTX 20 series) to the newest GPUs, in single, mixed and double precision builds.

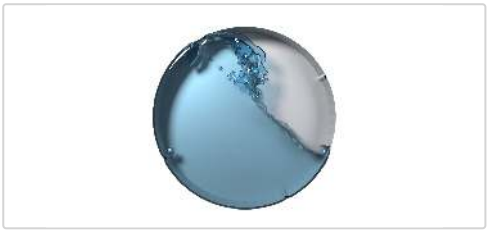


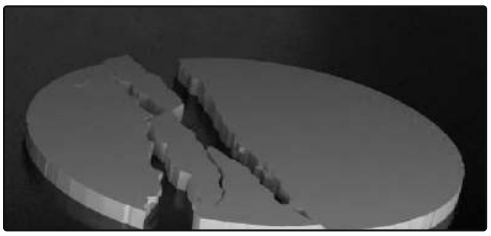
Particles: true shapes, real contact

- True convex polyhedra with volume based contact, not sphere clusters
- Rigid clumps up to 128 spheres, and flexible clumps
- Hertz or linear contact, Mindlin tangential, cohesion (Linear, BlazeNonLinear, JKR)

SPH liquids and polymer melts

- Native SPH with two way SPH-DEM coupling, inlet and outlet flow, slurry
- Polymer melt: power law or Cross-WLF viscosity, Maxwell viscoelastic stress, thermal melting
- Pellets start as DEM spheres or polyhedra and turn into SPH as they melt



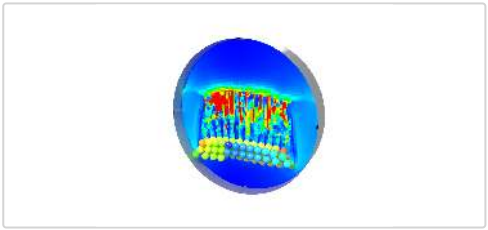


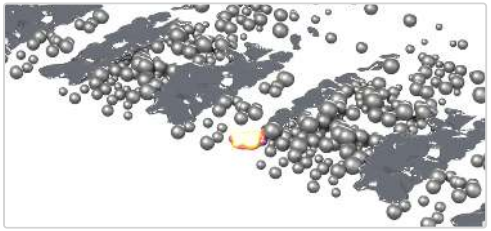
Bonded Elements: solids that crack

- Deformable Element meshes (tet, wedge, hex and OBJ) with vertex and face bonds and damage
- Cracks follow the bonds into fragments of any shape, over several generations
- Replacement breakage for spheres and polyhedra (Vogel-Peukert)

CFD-DEM: OpenFOAM and Kratos

- BlazeFOAM: two way coupling with OpenFOAM v2606, unresolved (Model A, Model B) or resolved immersed boundary
- DEM mapped onto unstructured meshes on the GPU; polyhedra in CFD-DEM, parallel (MPI), heat exchange
- Kratos coupling through Python



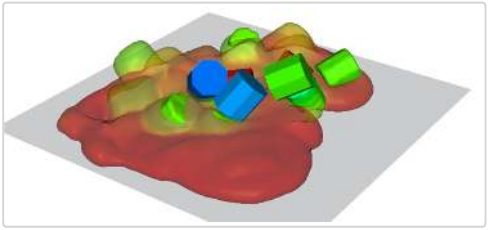


Laser powder bed fusion

- Laser heat input along tracks or hatch patterns
- Recoater spreading of the powder
- Melt and solidify, with surface tension, radiation and evaporation

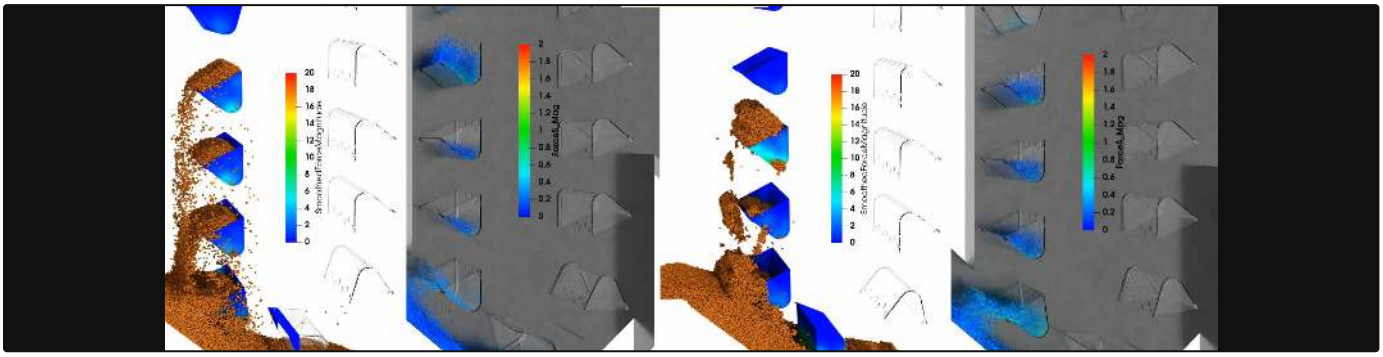
Heat, coating and wear

- Thermal conduction and heat flux; pellet beds melting on a hot plate
- Carrier coating with bonded parcels; wall wear per triangle from the material hardness
- Power balance, energy spectra, tallies, residence time, damage and force chains



Train machines on real physics.

A twin that reveals the underlying mechanics predicts beyond the conditions it was trained on. Models train directly on the live GPU simulation through our CUDA bridge, with WL3 hypergraph metrics computed by the solver itself during training. The solver physics runs alongside inference, self healing their predictions.

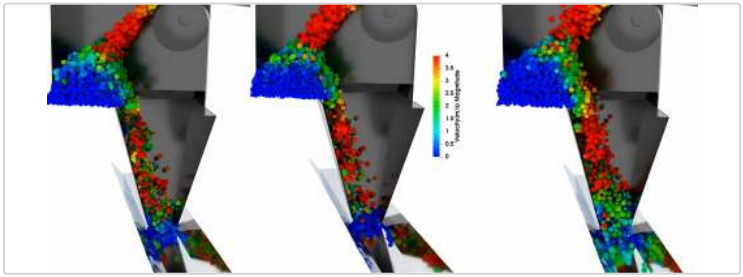


Bucket excavator DEM and live machine signals.

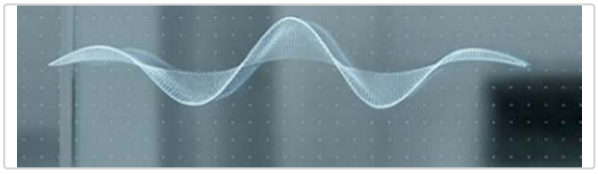
Stage	What the agent does
1. Calibrate and learn	The DEM is calibrated against the real machine; cohesion cases (clay binder, freezing) build a database of sensor and motor signals to DEM states.
2. Two phase training	Pre-training on recorded DEM runs, then live co-simulation: the agent acts (speed change, jog, reverse) and learns the causal effect.
3. Read the machine	From torque, tension, drive power and speed the agent infers contacts, payload, force chains and kinetic energy.
4. Check before acting	A predicted high tension state is verified by a DEM simulation built from the start state force chains.

CUDA bridge and training data

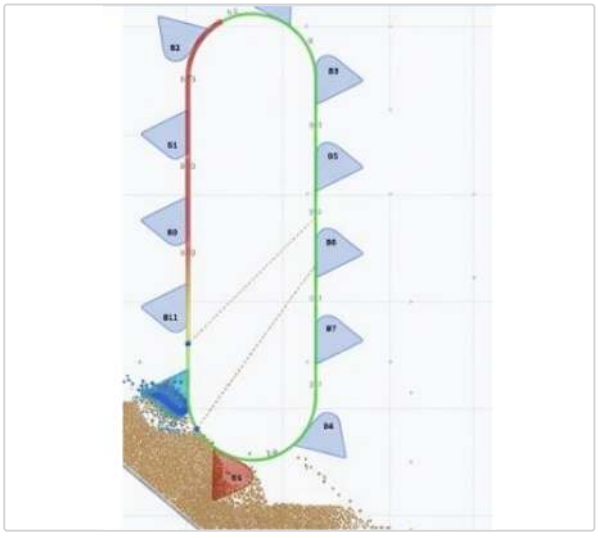
- CUDA bridge: the solver library runs inside the Python process, and CUDA device pointers to positions, orientations, velocities, forces, polyhedron state and Element stress, are used as PyTorch and CuPy tensors in place; Python kernels can change particles in the middle of a time step.
- The solver itself computes WL3 hypergraph metrics of the full contact network during training; the high fidelity data trains a hypergraph neural network that learns how the process itself works, such as mixing. Training frames go to HDF5; live coupling needs no files.
- Self healing inference: the solver physics runs alongside the model, and Jacobi relaxation pushes its predictions back to physical states on the GPU.
- For AI physics surrogates, robotics training, reinforcement learning and digital twins.



Hopper discharge with DEM and ML in the loop.



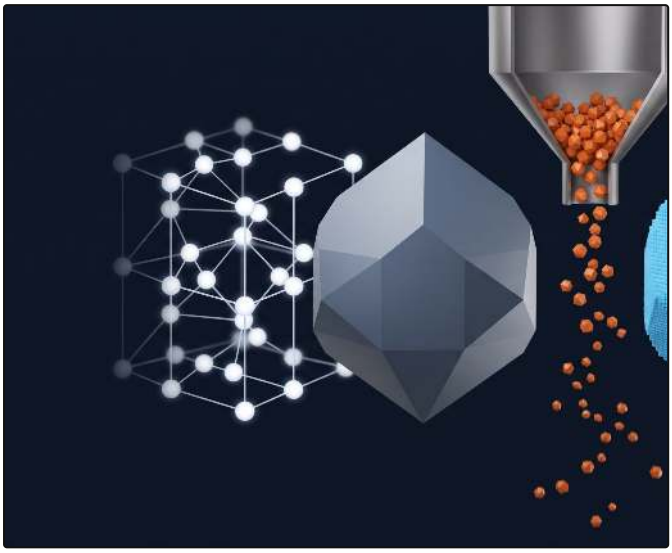
Learned drive torque signature.



DEM verification from start state force chains.

Bonded Elements that carry the crystal

Each bonded Element carries the full anisotropic stiffness (21 constants) and a crystal orientation, with several materials in one body, lattice misfit as eigenstrain and cohesive fracture with a fracture energy. Checked against exact elasticity within 1 %.



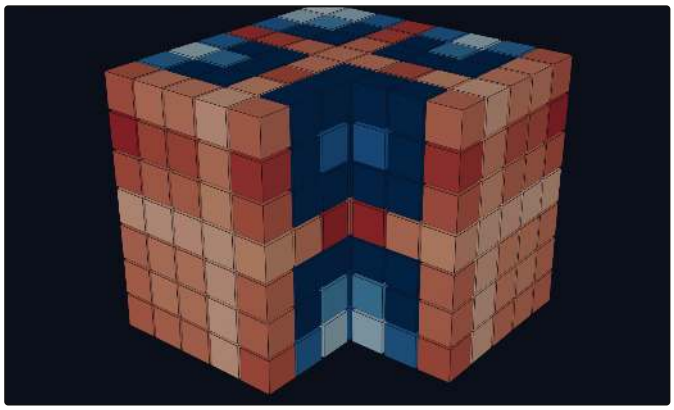
Atoms to Material: from the lattice to the process

BYOL: Bring Your Own Lattice.

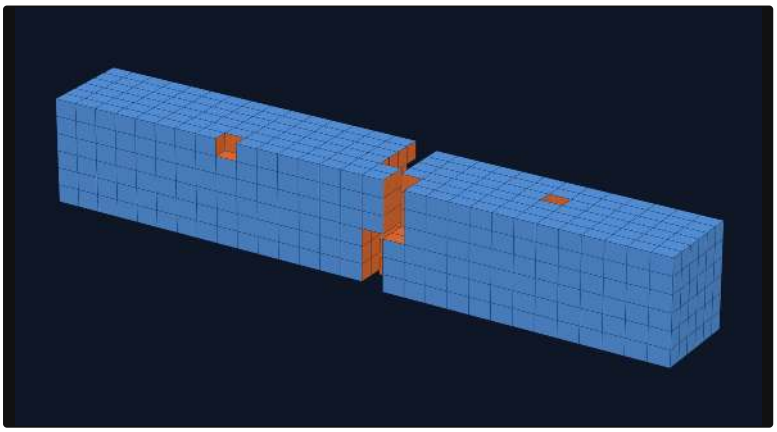
Established methods (MLIP, DFT) and the Atoms to Material workflow turn a crystal structure into simulation ready material properties: stiffness, crystal shape, and cohesion and friction per crystal face, every value labelled computed or calibrated.



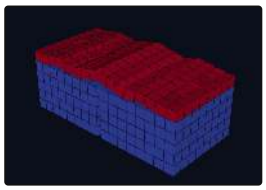
Alloy elastic ratio against the pair spring limit.



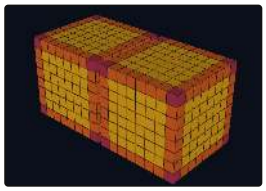
Coherency stress in a two phase nickel alloy.



Crystal lath of bonded Elements cracking through defects.



Brittle coating cracking on an alloy.

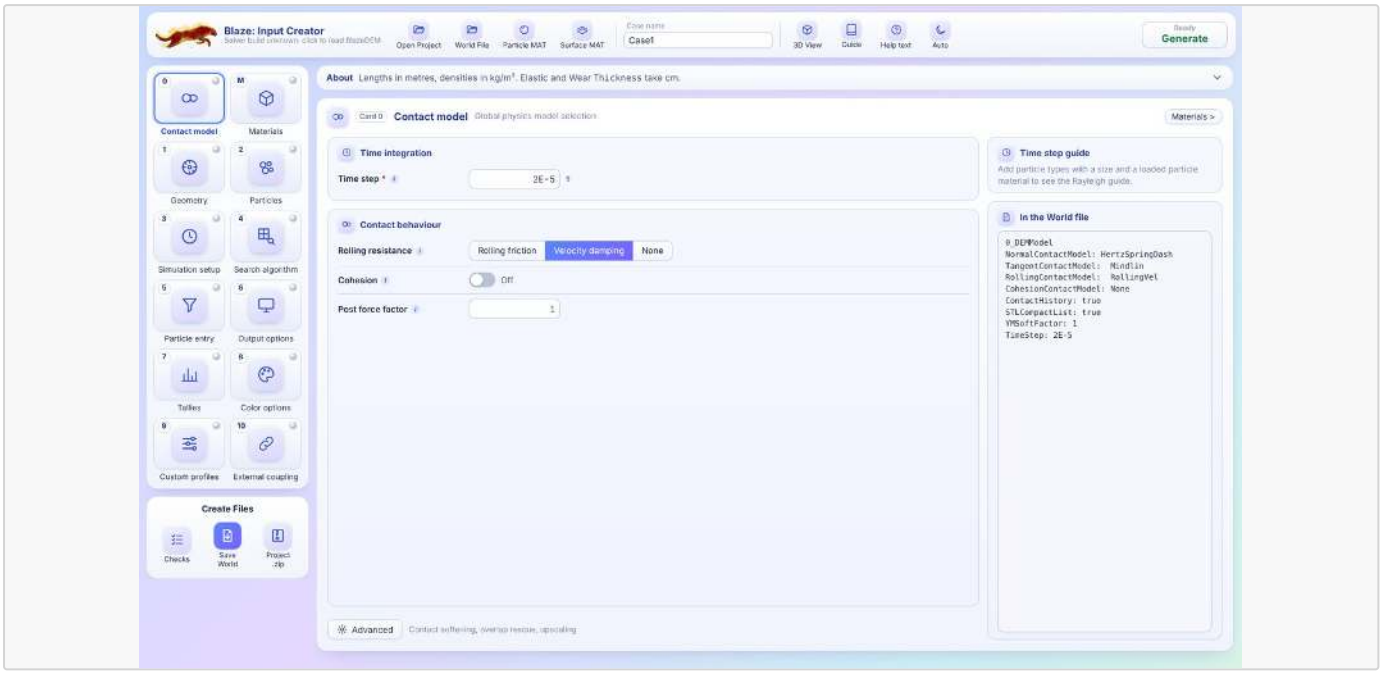


Crack running through a central void.

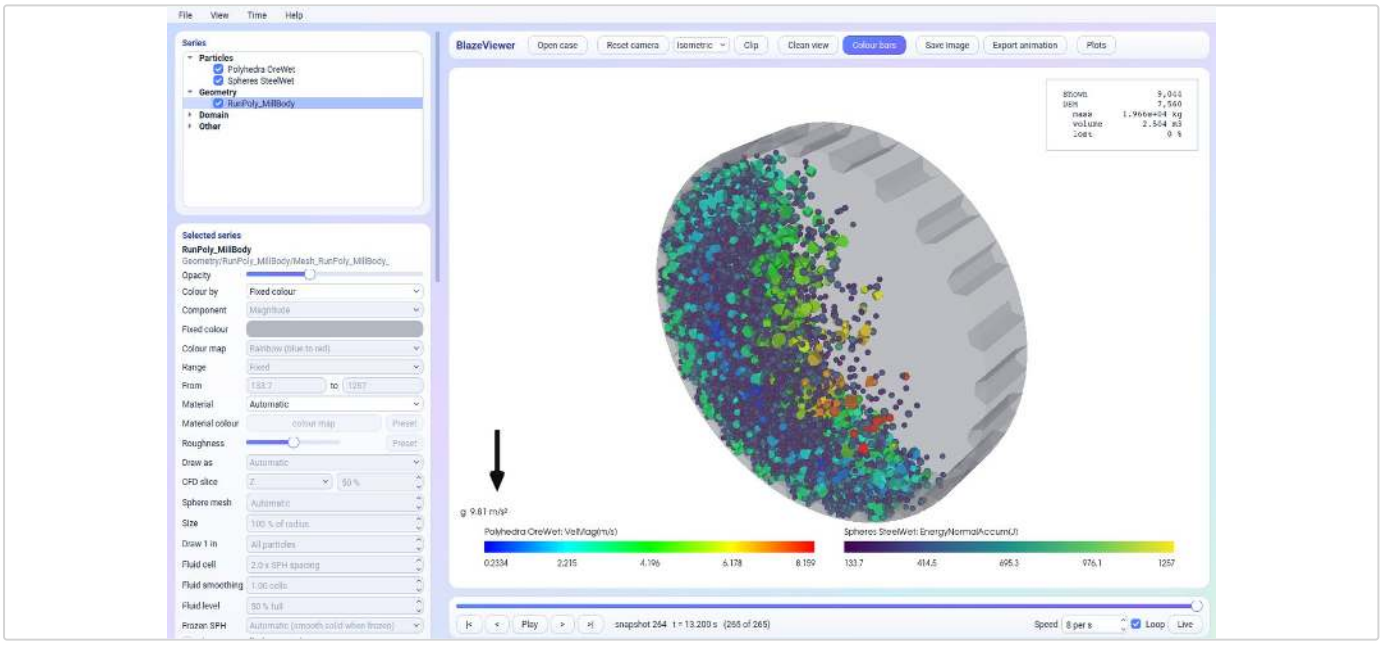
Fracture strength and fracture energy of bonded Elements are calibrated once per material class. Details in the Atoms to Material slide deck.

The tools in the package

Every program ships compiled for Linux and Windows; only an NVIDIA GPU driver is needed for runs. Example results open at once in BlazeViewer.



Input Creator browser interface for case setup.



BlazeViewer 3D particle and multiphysics results viewer.

Component	Role in the package
BlazeLauncher	Run queue per GPU with a memory check before each run, live monitor and plots.
Blaze SDK	Custom forces against the solver library.
CUDA bridge	Models train on the live GPU simulation with solver computed WL3 hypergraph metrics; Kratos through Python.

Why Blaze

Capability	BlazeMultiPhysics	Commercial enterprise DEM	Open source or CPU DEM
1. Hardware Efficiency GPU memory execution	✔ **High Single-GPU Efficiency** Runs entirely on the GPU with direct in-memory execution.	◐ **Multi-GPU Available** Multi-GPU solvers stall on CPU/PCIe transfers during multiphysics.	✘ **CPU Cluster Bound** Severe memory bandwidth bottlenecks across distributed CPU nodes.
2. Particle Shapes Geometry & breakage	✔ **True Polyhedra & Elements** True convex polyhedra & deformable Element meshes.	◐ **Clumps / Polyhedra** Multi-sphere clumps or older volume formulations; limited native crack breakage.	◐ **Spheres & Bonds** Primarily spherical particles and bonded sphere clumps used to approximate irregular shapes.
3. Fluids & Melt Hydrodynamics & thermal	✔ **Native GPU SPH & Melt** Native GPU SPH fluids, slurry, viscoelastic melt, and thermal phase change.	◐ **External CFD Add-Ons** Requires paid external CFD add-ons and expensive co-simulation licences.	◐ **Custom User Routines** Requires manual patching of separate CFD codes or custom user routines.
4. CFD Co-Simulation Mesh mapping & drag	✔ **Direct GPU Mesh Mapping** Native GPU unstructured mesh mapping and compiled kernels (OpenFOAM); two-way CFD-DEM coupling with Kratos through Python.	✔ **Turnkey CFD Suites** Established commercial CFD suites, but couplings stall on CPU cell search.	✔ **CFDEMcoupling** Established open-source CFD-DEM coupling on CPU; memory-bound scaling.
5. AI Integration Live tensors & GNN datasets	✔ **Direct VRAM Tensors** CUDA bridge to the simulation tensors and pointers (**MLLive**), solver computed WL3 hypergraph metrics & HDF5 graphs (**MLTrainGNN**).	◐ **Python / Sockets** Python scripting and socket interfaces; cannot share live GPU arrays mid-step.	◐ **File Dumps & Hooks** File-based I/O or custom C++ memory hooks; high latency for live RL loops.
6. Ecosystem & Support Ecosystem maturity	◐ **Specialised GPU Architecture** Agile specialist solvers and Python bridge; focused multiphysics workflow.	✔ **Broad Global Ecosystem** Extensive CAD interfaces, mature multi-physics suites, and global support.	✔ **Fully Inspectable Open Source** Zero licence cost, community-driven code, and fully open academic research base.

Heritage

Over ten years of peer reviewed GPU DEM development: mill charge, breakage, CFD-DEM coupling and ML.

BLAZE-DEM, to our knowledge the first published GPU DEM framework for convex polyhedral particles: Govender, Wilke, Kok, Els, Journal of Computational and Applied Mathematics 270 (2014) 386-400.

Open source GPU DEM for polyhedral particles: Govender et al., "Blaze-DEMGPU: Modular high performance DEM framework for the GPU architecture", SoftwareX 5 (2016) 62-66.

Volume based polyhedral contact: Govender, Wilke, Kok, Applied Mathematics and Computation 267 (2015) 810-829.

Evaluation and academic licences

BlazeMultiPhysics for Linux and Windows, single, mixed and double precision builds, GPU architectures from Turing to the newest.





NVIDIA CUDA Research Center since 2013 (Blaze-DEM, University of Johannesburg); NVIDIA Applied Research Accelerator Program; NVIDIA Inception Program member

Blaze Computing LTD

128 City Road, London, United Kingdom
director@blazecomputing.co.uk
www.blazecomputing.co.uk

Ground truth for Physical AI.