partsival – Collision-based Particle and many-body Simulations on GPUs for Planetary Exploration Systems

Roy Lichtenheldt[1], Simon Kerler[1,2], Andreas Angerer[3] and Wolfgang Reif[2]

[1] *Institute of System Dynamics and Control, German Aerospace Center (DLR), Roy.Lichtenheldt@dlr.de*
[2] *Institute for Software & Systems Engineering, University of Augsburg, reif@informatik.uni-augsburg.de*
[3] *XITASO GmbH, andreas.angerer@xitaso.com*

Developing new or optimizing existing locomotion systems for planetary exploration faces two major challenges: First, environmental conditions like reduced gravity are hard to mimic in laboratories. Second, physical prototyping can be time-consuming and cost-intensive. To address both, virtual prototypes are simulated to verify new designs and concepts, e.g. for rover wheels, before first physical prototypes are constructed. Often a method to model soil is required to correctly simulate tool interactions. A common way is the Discrete Element Method (DEM) which discretizes soil volumes into particles. The software currently used for DEM computations at DLR is a CPU-bound, multi-threaded engine. Due to its high degree of parallelism, the DEM profits from being executed on graphics processing units (GPUs). In order to make absolute statements and predictions, high accuracy is required and therefore the penalty based contact formulation is preferred. However, most of the DEM implementations for GPUs employ constraint based, hard contact formulations [1], e.g. Project Chrono [2]. Therefore, this paper presents partsival – a collision-based particle simulation framework targeting graphics cards [1].

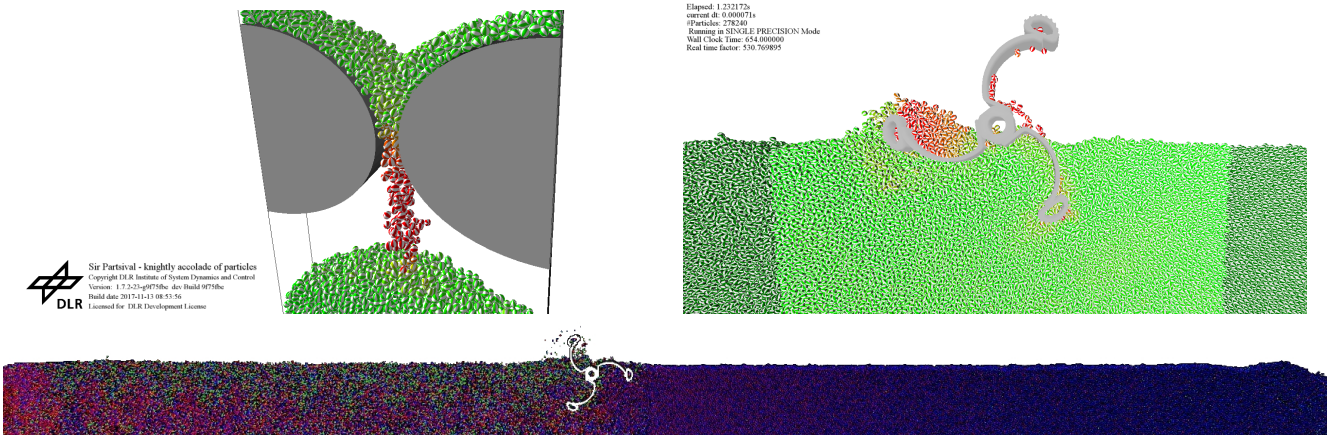

Fig. 1: Funnel piling application (left), single wheel test simulation of a rimless wheel [9] (right), 6 m single wheel test, ≈ 20 s of time (bottom)

GPU computing is performed via OpenGL 4.3+ compute shaders which ensures platform and vendor independence. Partsival integrates natively into OpenSceneGraph [3] and consequently features an object-oriented, hierarchical software architecture. A central particle system class is configured by adding one or more simulation steps, each of which defines a single action, like contact dynamics or integration. To facilitate the use of partsival, wrapper functions shift the focus from programming to configuration in user models. Instanced rendering is used for efficient on-line visualization of simulations. Since renderer and compute shaders have access to the same GPU memory, particle data is accessed directly without the need of copying between GPU and CPU, therefore avoiding PCIe (Peripheral Component Interconnect Express) and context switching latency.

Partsival currently features pre-defined simulation steps for state of the art [4] as well as advanced inter-particle and particle-mesh contact models [5, 6]. The framework is extendable either via additional steps derived from the existing classes or by an editable plugin shader class. In order to integrate the equations of motion, an integration scheme particularly developed for partsival is used [7]. Due to its novel approach, implicit time integration becomes available without renewed contact detection during corrector iteration. A further improvement in long dense packing simulations is gained by dynamic boundaries, i.e. activating and deactivating particles depending on the

tool position [8] (see the differently lighted particles of the right top image in Fig. 1). The engine also supports switching between single and double floating point precision at startup allowing to use fast single precision for relative statements (e.g. wheel-optimization scenarios) and double precision for absolute statements in prediction.

Fig. 1 depicts three example applications: The single wheel test is a typical application in planetary exploration needed to improve traction on loose sandy terrain, featuring dense particle packings. The funnel piling example is featuring both, loose particulate flow and dense packing including pile formation on the bottom. For validation of partsival, an existing example of the CPU code has been recreated. Thereby the angle of repose after 3 s of simulation, as well as the overall particle behaviour have been compared with good agreement as shown in Fig. 2 (right). Based on the funnel piling example a performance and scaling evaluation has been carried out for both, partsival and the previously used CPU code. For comparability, both used the LICHTENHELDT-jolt integration scheme [7] and identical contact models, as well as boundary conditions. The CPU code has been executed in

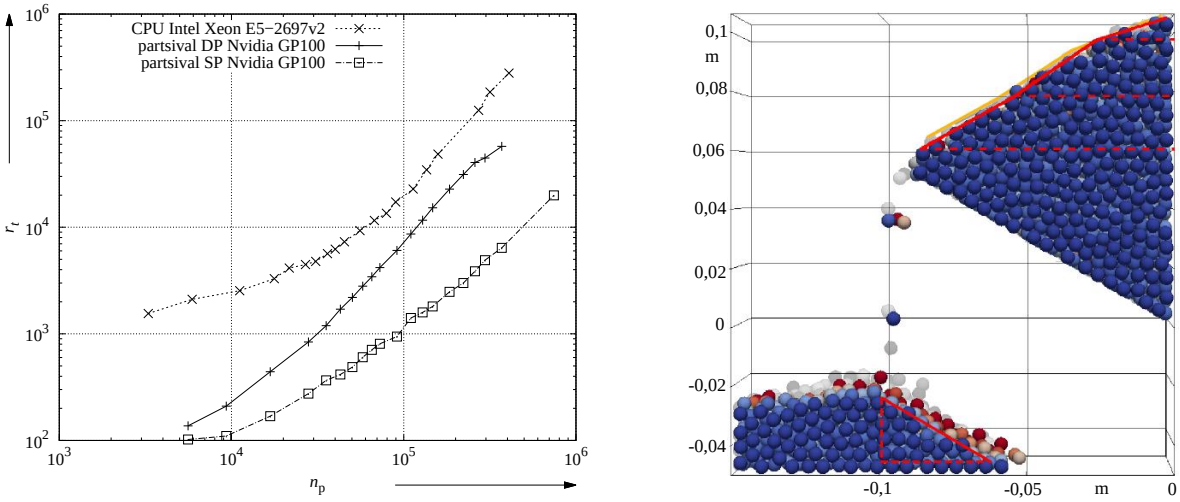

Fig. 2: Performance evaluation by comparing the real-time factor for different numbers of particles (lower is better) in double logarithmic scale (left); comparison of simulation results of partsival (colored particles and red lines) and the CPU code (grey translucent particles and orange line) (right)

double precision, whereas partsival is used in both, single and double precision. The results in Fig. 2 (left) indicate that partsival outperforms the CPU code for all tested numbers of particles (up to $\approx 7.5 \cdot 10^5$) on a single GPU. It has to be stated that the CPU code features an optimized $\mathcal{O}(n \log n)$ contact detection, whereas partsival is still $\mathcal{O}(n^2)$ in the first version. Hereby n denotes the number of particles. By speeding DEM and many body simulations up by orders of magnitude, partsival allows for faster computations as well as to simulate scenarios which would have not been feasible on the CPU. Such an example can be seen in Fig. 1 (bottom) while the simulation completed in 2 hours and 23 minutes, whereas a comparable CPU simulation would have required at least a full day.

References

[1] Kerler, S.; Collision-based Particle Simulations on GPUs for Planetary Exploration Systems, Masters Thesis, University of Augsburg, 2017

[2] Tasora A. et al.; Chrono: An open source multi-physics dynamics engine. In HPC in Sci. and Eng., Springer, 2016.

[3] Open Scene Graph; http://www.openscenegraph.org/, 12/05/2017

[4] Obermayr, M.; Prediction of Load Data for Construction Equipment using the DEM, Dissertation, Stuttgart, 2013

[5] Lichtenheldt, R., Schäfer, B.; Planetary Rover Locomotion on soft granular Soils - Efficient Adaption of the rolling Behaviour of nonspherical Grains for Discrete Element Simulations, Particle-based Methods III, S. 807-818, ISBN 978-84-941531-8-1, 2013

[6] Lichtenheldt, R.; Lokomotorische Interaktion Planetarer Explorationssysteme mit weichen Sandböden - Modellbildung und Simulation, Verlag Dr.Hut, ISBN 978-3-8439-2704-8, 2016

[7] Lichtenheldt, R.; A stable, implicit time integration scheme for Discrete Element Method and contact problems in dynamics, Particle-based Methods V, ISBN: 978-84-946909-7-6, CIMNE, 2017

[8] Lichtenheldt, R.; Covering shock waves on Mars induced by InSight's HP3-Mole , Coupled Problems VII, ISBN: 978-84-943928-3-2, Artes Grficas Torres S.L., 2017

[9] Stubbig, L., Lichtenheldt, R., Becker, F., Zimmermann, K.; Model-based development of a compliant locomotion system for a small scout rover, 59th Int. Scientific Colloquium, Ilmenau, 2017