# Molecular Dynamics with CUDA

meta-project

Enrico Verdolotti

January/February 2020

This project will be based on a previous work of Emily Crabb, publicly available at: `https://github.com/ejc44/MD`. The main objective is to successfully implement the same algorithms and specifications, harnessing GPU multi-threading capability and in particular, exploiting CUDA toolkit. Moreover will be done some review of the original code and introduced new stuffs as described below.

The envisioned steps are:

1. Review and rewrite **serial** version;
2. Write **GPU multi-thread** version with **CUDA**;
3. Review and rewrite **CPU multi-thread** version;
4. **Benchmark** and comparisons;
5. New **animated visualization**.

Two years are passed from the original work so the revision of **serial** version will be done mainly for better understanding of algorithm, data structures involved, entire workflow and, possibly for old-code revision using new constructs that maybe was not available in the previous version of Julia, taking the opportunity for refactoring and finding bottlenecks (e.g. the "find forces" function, re-allocate for each time step the entire forces "array" and maybe this can be avoided.)

In the original work, the **GPU version** implementation was left incomplete due to the complexity of main functions which contains several conditionals check. That make hard to write GPU code using just built-in functions and CuArrays. The idea for addressing this problem is to write an ad hoc kernel function capable to run on GPU along with the classic CuArrays usage.

It would be nice to have a **CPU multi-thread** version more similar to GPU version for comparisons and even if the common workflow for developing multi-threaded code is to write first CPU multi-thread version and then the GPU one, in this case will be done in reverse due to two reasons: First, the multi thread CPU code is already available and working and, second, to adapt a kernel function to CPU or GPU from the respective counterpart is relatively easy.

As ever, will be mandatory to test if a real speedup has been achieved. For this reason, **BenchmarkTools** package will be used to compare both current results and original ones.

Lastly, a more expressive **visualization** of particle motion will be explored using classic Plot library or Makie for 3D animations.

In conclusion some of the original project, such as distributed computing, part will not be considered even if these was written with `@parallel` macro that is no longer supported and this would need a code rewriting. That choice has been done due to the low score obtained for distributed version in original code and lack of time but that doesn't mean it cannot be resumed in future.

*"We adore chaos because we love to produce order." – M.C. Escher*