# Simulation in Computer Graphics - Exercises

Computer Graphics - Computer Science Department - University of Freiburg

# Particles

The goal of this exercise is to simulate and visualize a set of moving particles under gravity in a cuboid.

### Coin3D

We provide a framework that employs Coin3D for visualization purposes. Therefore, Coin3D has to be installed. Coin3D can be downloaded from https://bitbucket.org/Coin3D/coin/downloads.

*Windows specific:* Download `Coin-3.1.3.zip` and `SoWin-1.5.0.zip`. The repositories contain `sln` files for `msvc6` to `msvc9`. The solution file in the `msvc9` folder should work up to Visual Studio 2008. Before building the solution, a system environment variable `COINDIR` has to be set to the directory where the libraries are finally copied to. Libraries for release mode (`coin3.lib, sowin1.lib, coin3.dll, sowin1.dll`) and for debug mode (`coin3d.lib, sowin1d.lib, coin3d.dll, sowin1d.dll`) can be built.

*Linux specific:* `Precisetimer` can be safely removed as it is only used for performance measurements.

### ParticleViewer

- Download `sim_exercise_particles.zip`, install `ParticleViewer`, and get familiar with the code. Please focus on the interaction of simulation and visualization. E.g., Coin's idle sensor is used to trigger a simulation step, particle positions are copied to Coin's scene graph after a simulation step.
- Generate a set of particles. Specify mass, initial (random) position and initial (random) velocity.
- Implement Euler, Euler-Cromer, Verlet, and Heun.
- Implement the computation of the kinetic and potential energy of all particles to check if the total energy is preserved during the simulation.
- Compare and analyze the numerical integration schemes. E.g., initialize four particles with similar initial positions, the same initial velocity, the same mass, but different integration schemes. Vary `TIMESTEP` or `WORLD_ELASTICITY`.

Please note that Verlet might require changes in the collision handling with the wall (`worldCollision`). Think about the validity of $\mathbf{x}(t-h)$ after collision handling. In terms of Heun, try to implement it as accurate as possible. Heun has to compute forces at predicted positions with predicted velocities. While `addForces` does not depend on position and velocity in our simplistic example, it generally does.