Universität Heidelberg
Institut für Technische Informatik

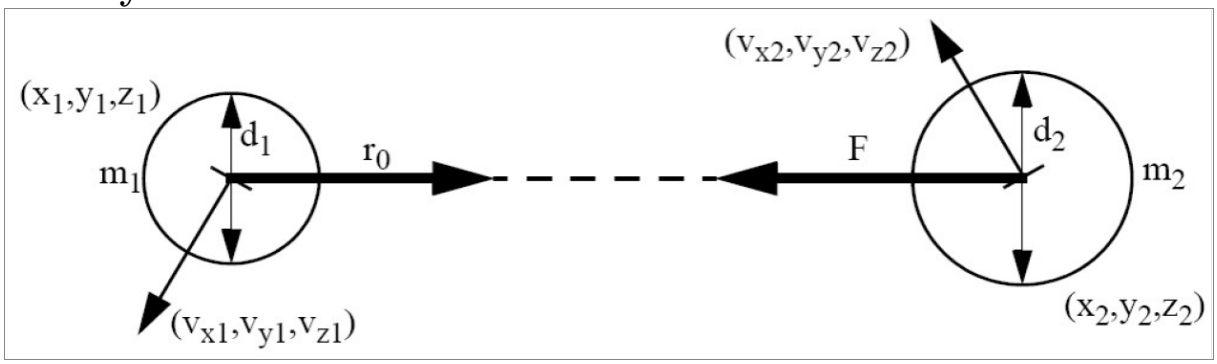
Holger Fröning
Lorenz Braun

GPU Computing
Term 2016/2017 (Winter)

Exercise 7

- **Return electronically until Thursday, 22.12.2016 14:00**
- **Include name on the top sheet. Hand in only one PDF.**
- **A maximum of three students is allowed to work jointly on the exercises.**
- **Hand in source code if the exercise required programming.**

n-Body Problem



Each object has the following data:

- Position p = (x,y,z) [m]
- Velocity v = (vx,vy,vz) [m/s]
- Mass m [kg]

Note that calculations are performed in 3D space. These are the most important formulas:

$$Force: F = -\gamma * \frac{m_1 \cdot m_2}{r^2} \cdot \overrightarrow{r_0}[N] \qquad \gamma = 6.673 \cdot 10^{-}11 \frac{Nm^2}{kg^2}$$

$$Acceleration: \overrightarrow{a} = \frac{\overrightarrow{F}}{m}[\frac{N}{kg}]$$

$$Velocity: \overrightarrow{v} = \overrightarrow{a} \cdot \Delta t[\frac{m}{s}]$$

$$Distance: \overrightarrow{s} = \overrightarrow{v} \cdot \Delta t[m]$$

Ensure the following:

1. The program accepts the number of objects as command line parameter.
2. Use single precision floating points for all object data.
3. Define the length of a time step with the constant DELTA_T.
4. Initialize positions and masses with random values, velocity with zero.
5. Perform a reasonable amount of time steps (iterations), for instance 100. Each iteration, update the velocity and position of all objects.

7.1 Naïve GPU implementation

Implement a naïve GPU version that primarily serves for testing the compute kernel. Use SOA without packed values for the data structures, and do not use shared memory. We will later report speedups based on this baseline performance. Do not include data movement over the PCIe interface in your measurements.

- Report performance numbers for an increasing body count and discuss!

(30 points)

7.2 Optimized GPU implementation

Based on the previous implementation, perform optimizations to maximize performance for this n-body computation. In particular:

- Consider if you prefer packed SOA or AOS data structures
- Make use of tiling to optimize for shared memory use
- Optional: investigate if loop unrolling is beneficial

Report performance numbers and discuss your optimizations. For which reason(s) is a certain optimization increasing performance?
(50 points)

Total: 80 points