



University of Stuttgart  
Germany



Linus Bantel,  
Alexander Strack,  
Prof. Dr. Dirk Pflüger

# Programming Project Galaxy-Crash

# Who are we?



**Linus Bantel**

[Linus.Bantel@ipvs.uni-stuttgart.de](mailto:Linus.Bantel@ipvs.uni-stuttgart.de)



**Alexander Strack**

[Alexander.Strack@ipvs.uni-stuttgart.de](mailto:Alexander.Strack@ipvs.uni-stuttgart.de)



**Prof. Dr. Dirk Pflüger**

[Dirk.Pflueger@ipvs.uni-stuttgart.de](mailto:Dirk.Pflueger@ipvs.uni-stuttgart.de)

# Target - Phase 1

Simulation of two colliding galaxies.

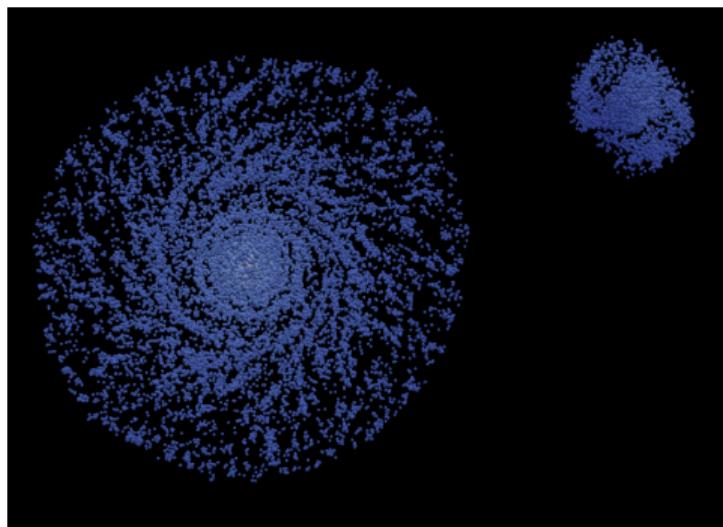
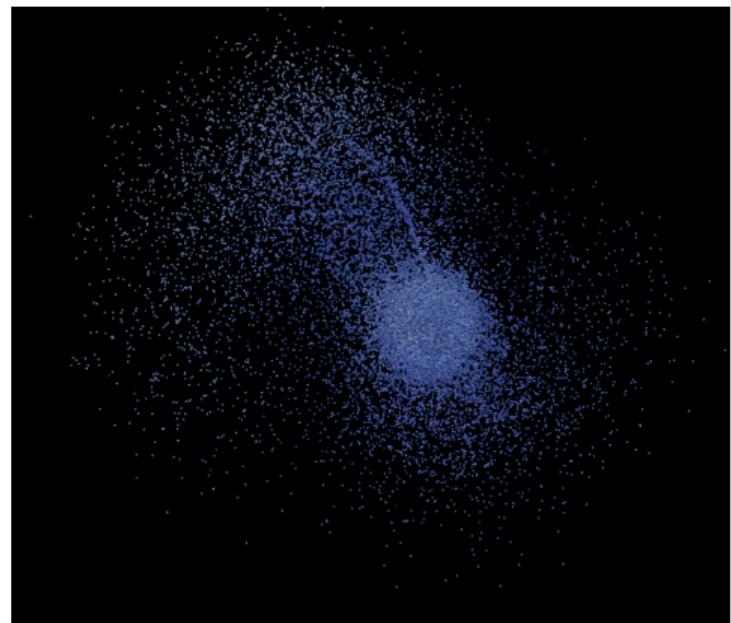
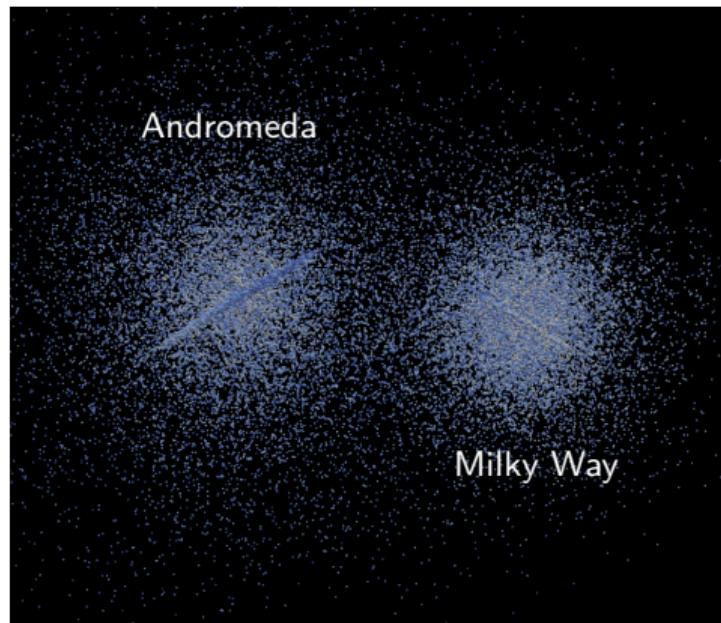


Image of the Whirlpool Galaxy (M51), located 23 million light-years away from Earth, captured by the Hubble Space Telescope in 2005.

Source: <https://hubblesite.org/contents/media/images/2005/12/1677-Image.html>

## Target - Phase 2

Simulation of the collision between the Milky Way and the Andromeda Galaxy.



# Learning Objectives

- Implementation of a non-trivial project in a small group.
- Developing a sense for different algorithmic complexities.
- Practical application of tree data structures.
- Basic knowledge of the C++ programming language
- Gaining experience with Git<sup>↗</sup>.
- Familiarization with working on remote systems using the bwCloud<sup>↗</sup>.
- Utilizing established programs in scientific computing (e.g., ParaView<sup>↗</sup> for visualization).

# Learning Objectives - Working Method

**Independent Work!**

# Learning Objectives - Working Method

## Independent Work!

*“Manchmal hat man sich schon etwas verloren gefühlt, z.B. hatte ich 0% Plan wie man die BW Cloud benutzt. Hab mir dann mit ganz viel Hilfe von Google und YouTube eine eigene VM aufgesetzt (ca. 10h bis alles läuft, da ich keine Vorerfahrung hatte). Es waren oft Kleinigkeiten, wo einem oft nur Foreneinträge geholfen haben . . . ”*

# Procedure

- Multiple phases:
  - Phase 0** Formation of groups of 3.  
(Deadline: **15.04.2025**)
  - Phase 1** Familiarization with the n-body problem and implementation using the naive algorithm, including visualization.  
(Deadline: **27.05.2025**)
  - Phase 2** Implementation of the tree-based Barnes-Hut algorithm, including evaluation and performance comparison with the naive algorithm.  
(Deadline: **15.07.2025**)
- Presentation** 5 min final presentation per group.  
(Date: **TBA**)

# Procedure

- Multiple phases:

**Phase 0** Formation of groups of 3.

(Deadline: **15.04.2025**)

**Phase 1** Familiarization with the n-body problem and implementation using the naive algorithm, including visualization.

(Deadline: **27.05.2025**)

**Phase 2** Implementation of the tree-based Barnes-Hut algorithm, including evaluation and performance comparison with the naive algorithm.

(Deadline: **15.07.2025**)

**Presentation** 5 min final presentation per group.

(Date: **TBA**)

- Acceptance by the end of each phase.  
→ Progress to the next phase only after successful acceptance of the current phase.
- Additionally, there is a small competition to determine the fastest simulation . The winners will receive a small prize.

# General - TIK GitHub

- Mandatory use of the University's provided GitHub instance [↗](#) by TiK.
- Each student can log in with their stXXXXXX credentials.
- Create **one private** repository per submission group.
- Add all group members and supervisors \* to the repository.
- Each repository **must** contain a README with all necessary information for compiling and running the code.
- C++ + CMake Repository Template [↗](#).
- For a brief Git introduction, see Git HowTo [↗](#).

\*ac134440, ac142406

## General - bwCloud

- bwCloud  as the online VM hosting service of the state of Baden-Württemberg.
- Register: [https://www.bw-cloud.org/de/erste\\_schritte#step1](https://www.bw-cloud.org/de/erste_schritte#step1)
- Log in after registration: <https://portal.bw-cloud.org/auth/login/> (via “bwIDM via OpenID Connect” → regular st-account)
- Create an instance with the following configuration:
  - Details: Assign a name
  - Source: Ubuntu 24.04
  - Flavor: m1.tiny (1 VCPU, 1 GB RAM)
  - Key Pair: Add your own public SSH key (Instructions for creating an SSH key on Linux or Windows: <https://docs.oracle.com/en/cloud/cloud-at-customer/occ-get-started/generate-ssh-key-pair.html>)
- Log in to the instance: `ssh ubuntu@IP_address_of_the_instance`

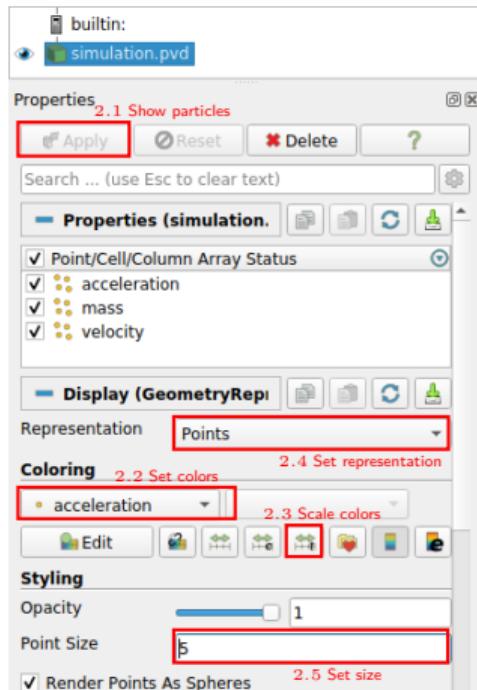
## General - bwCloud: Installation of allowed software

- Update: `sudo apt update && sudo apt upgrade`
- Allowed software for C++:  
`sudo apt install g++ cmake cmake-curses-gui`

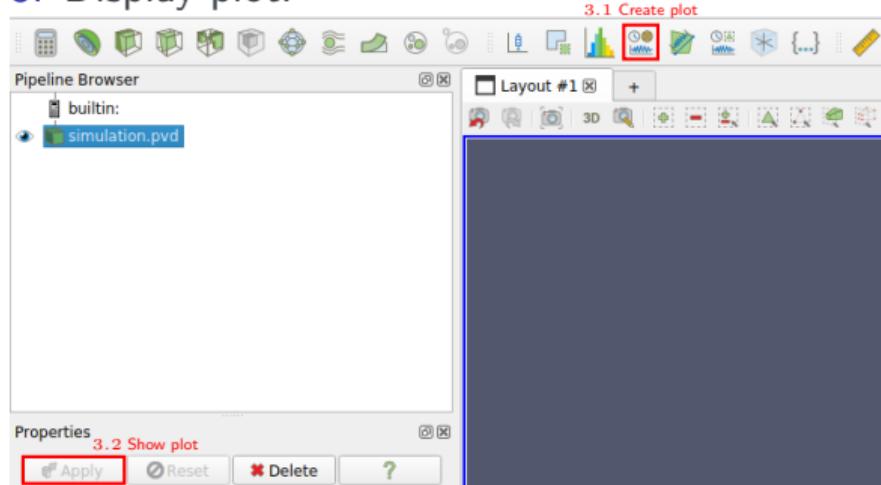
You are only allowed to use any C standard functionality **as long as** there is **no** C++ equivalent! As an example, allowed is the `<cmath>` header since there is no pure C++ alternative. Not allowed are, e.g., `printf` (use `std::cout` or `std::format` instead) or the C string functions operating on `const char *` (use `std::string` instead).

# General - ParaView

1. Open simulation file:  
File → Open → .pvda
2. Display simulation:



3. Display plot:



4. Take screenshot:  
File → Save Screenshot... (Save All Views)
5. Create animation:  
File → Save Animation... (Save All Views)

# Submission Format

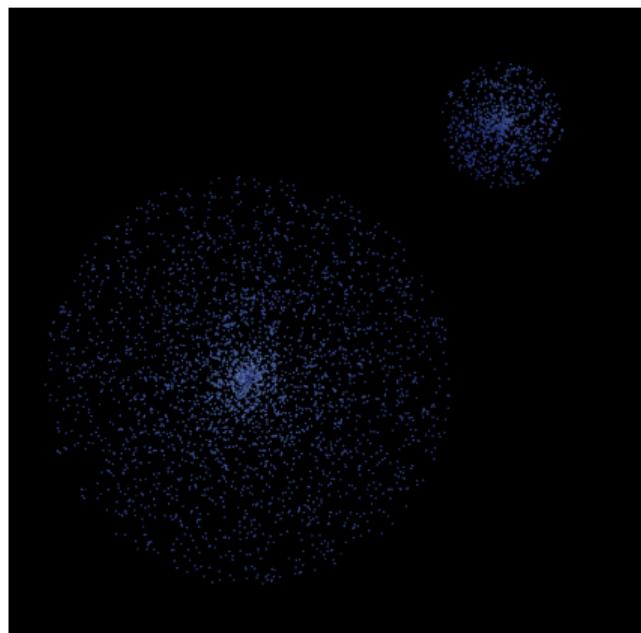
- For each phase, **exactly one** \*.zip or \*.tar.gz file must be submitted in the ILIAS course<sup>↗</sup>, containing the content corresponding to the respective phase.
- Phase number followed by the last names of all group members, e.g., Phase1\_LastName1\_LastName2\_LastName3.tar.gz.
- Each file must include the names of **all** contributors.
- Submission must be compilable and executable from the command line under Linux, i.e., **no** IDE-specific build scripts!

# Phase 0: Formation of groups

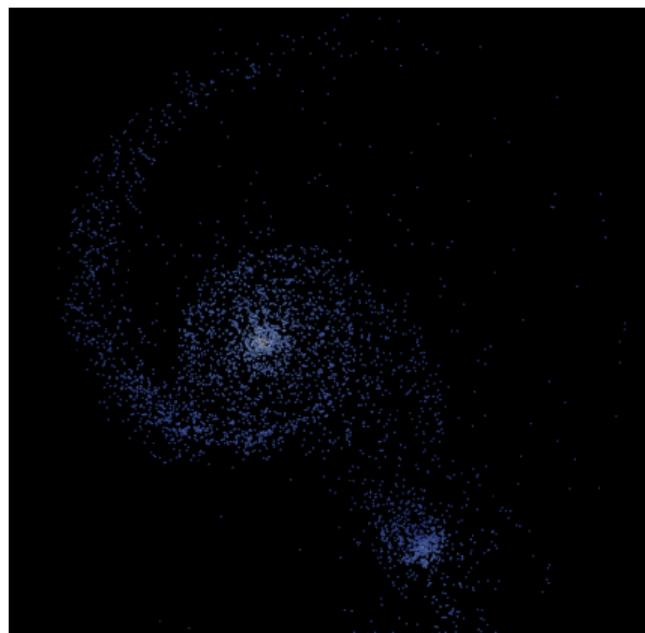
- Independent formation of groups of 3 members (e.g., through the ILIAS Forum ).
- As a first submission , upload a text file with all group members and a group name for the competition in ILIAS **and** add all group members to the submission group.
- Deadline: **15.04.2025**
- If you haven't found a group by then, you must send us an email, and will be assigned to a group.
- If you haven't found a group by the deadline and haven't contacted us, you will **not** be considered a participant in this course!

# Phase 1

- Initial naive implementation of an n-body simulation including visualization.
- Simulation of two colliding galaxies with 4000 particles.



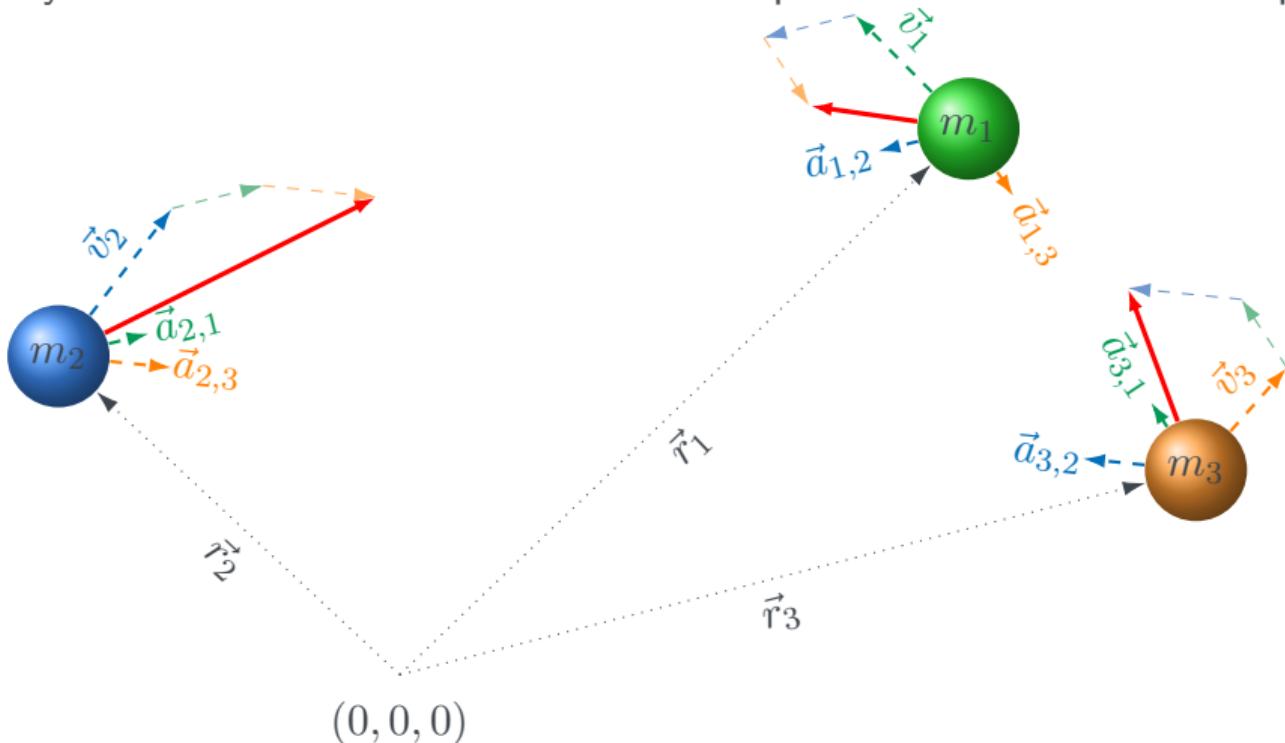
$t = 0$



$t = 6\,000\,000$

# Phase 1: Basic idea of n-body simulations

The n-body simulation aims to establish motion equations for each individual particle.



## Phase 1.1: Naive implementation of an n-body simulation

Each particle  $i$  with mass  $m_i$  and position vector  $\vec{r}_i$  experiences the force  $\vec{a}_i$  from all other particles according to Newton's law of universal gravitation:

$$\vec{a}_i = \sum_{i \neq j} Gm_j \frac{\vec{r}_j - \vec{r}_i}{(\|\vec{r}_j - \vec{r}_i\|_2^2 + \epsilon^2)^{\frac{3}{2}}}$$

## Phase 1.1: Naive implementation of an n-body simulation

Each particle  $i$  with mass  $m_i$  and position vector  $\vec{r}_i$  experiences the force  $\vec{a}_i$  from all other particles according to Newton's law of universal gravitation:

$$\vec{a}_i = \sum_{i \neq j} G m_j \frac{\vec{r}_j - \vec{r}_i}{(\|\vec{r}_j - \vec{r}_i\|_2^2 + \epsilon^2)^{\frac{3}{2}}}$$

- Gravitational constant:  $G = 6.674\,28 \times 10^{-11} \frac{\text{m}^3}{\text{kg}\cdot\text{sec}^2}$
- Units do not match the given datasets:
  - Mass in solar masses:  $M_\odot = 1.988\,435 \times 10^{30} \text{ kg}$
  - Distance in parsec:  $1 \text{ pc} = 3.085\,677\,581\,29 \times 10^{16} \text{ m}$
  - Time in years:  $1 \text{ yr} = 365.25 \cdot 86\,400 \text{ sec}$
- Note: the correct scaling of  $G$  should be calculated in the program!
- Softening factor:  $\epsilon = 0.1$  to avoid collisions between particles

## Phase 1.2: Energies in the system

- Kinetic energy:  $E_K = \sum_i \frac{1}{2} m_i \|\vec{v}_i\|_2^2$
- Potential energy:  $E_P = - \sum_{i < j} \frac{G m_i m_j}{\|\vec{r}_j - \vec{r}_i\|_2}$
- Total energy in the system:  $E_T = E_K + E_P$
- $E_T$  remains constant in a stable simulation (energy conservation)!

## Phase 1.2: Energies in the system

- Kinetic energy:  $E_K = \sum_i \frac{1}{2} m_i \|\vec{v}_i\|_2^2$
- Potential energy:  $E_P = - \sum_{i < j} \frac{G m_i m_j}{\|\vec{r}_j - \vec{r}_i\|_2}$
- Total energy in the system:  $E_T = E_K + E_P$
- $E_T$  remains constant in a stable simulation (energy conservation)!

The "Virial Equilibrium":

$$\frac{2 \cdot E_K}{|E_P|} \approx 1.0$$

as a dynamic equilibrium state within a timescale comparable to a multiple of the typical time it takes for a particle to traverse the system.

**Question:** What happens if the result is  $> 1.0$  or  $< 1.0$ ?

See: [http://www.scholarpedia.org/article/n-body\\_simulations\\_\(gravitational\)](http://www.scholarpedia.org/article/n-body_simulations_(gravitational))

## Phase 1.3: Leapfrog Method

The Leapfrog method, a second-order time integration method, combines two variants of the symplectic Euler method (SE) to move from time step  $k$  to the next time step  $k + 1$ :

$$(SE1) \begin{cases} \vec{v}^{k+\frac{1}{2}} = \vec{v}^k + \vec{a}^k \frac{\Delta t}{2} \\ \vec{r}^{k+\frac{1}{2}} = \vec{r}^k + \vec{v}^{k+\frac{1}{2}} \frac{\Delta t}{2} \end{cases}$$

$$(SE2) \begin{cases} \vec{r}^{k+1} = \vec{r}^{k+\frac{1}{2}} + \vec{v}^{k+\frac{1}{2}} \frac{\Delta t}{2} \\ \vec{v}^{k+1} = \vec{v}^{k+\frac{1}{2}} + \vec{a}^{k+1} \frac{\Delta t}{2} \end{cases}$$

See: O. Buneman: Time-Reversible Difference Procedures (1967)  
(DOI: [https://doi.org/10.1016/0021-9991\(67\)90056-3](https://doi.org/10.1016/0021-9991(67)90056-3))

## Phase 1.4: Complete Simulation

---

- 1: Adjust initial velocities:  $\vec{u} = \frac{\sum_j m_j \vec{v}_j}{\sum_j m_j}; \quad \vec{v}_i = \vec{v}_i - \vec{u}$
  - 2: Update accelerations  $\vec{a}_i$
  - 3: Visualize initial state
  - 4: **while** Simulation end not reached **do**
  - 5:     Perform Leapfrog time integration step
  - 6:     **if** Time step should be visualized **then**
  - 7:         Visualize time step
  - 8:     **end if**
  - 9:     Update time step
  - 10: **end while**
  - 11: Save final state of the system as .csv (same format as input)
-

## Phase 1.5: Reading the Simulation Data

- Initial scenarios of different sizes are provided in ILIAS [↗](#):
  - Small dataset for testing and validation: 50 particles
  - Medium-sized dataset for performance optimizations: 500 particles
  - Large dataset for final simulation: 4000 particles
- .csv file format: the first line contains header information, followed by the actual data:

```
id,mass,pos_x,pos_y,pos_z,vel_x,vel_y,vel_z
0,1e+06,0,0,0,0,0,0
1,1.6389,5.3848,-3.9855,0.10744,1.5294e-05,2.1107e-05,0
2,5.5756,6.7128,-8.9865,0.15009,1.5936e-05,-1.2401e-05,0
```

- The values are in the following units: mass in  $[M_{\odot}]$ , positions in [pc], and velocities in  $\frac{[pc]}{[yr]}$ .
- Data must be stored and processed in double precision (FP64).

## Phase 1.6: Visualization

Mandatory:

- Visualize the data using ParaView .
- Create a .vtp file at each visualization step, containing the current state of the simulation: particle mass, position, velocity, acceleration, and kinetic, potential, and total energy of the system.
- Create a single .pvf file for the **entire** simulation, referencing all the .vtp files along with their corresponding time steps.

## Phase 1.6: Visualization

Mandatory:

- Visualize the data using ParaView [↗](#).
- Create a .vtp file at each visualization step, containing the current state of the simulation: particle mass, position, velocity, acceleration, and kinetic, potential, and total energy of the system.
- Create a single .pvf file for the **entire** simulation, referencing all the .vtp files along with their corresponding time steps.

Optional (easier for debugging):

- Live visualization during the actual simulation.
- In C++, for example, visualize the ...
  - Bodies using PCL [↗](#).
  - Energies using matplotlib-cpp [↗](#).
- **Important:** Other visualization libraries can be used here, but the final submission must be compilable without them.

## Phase 1.6: Example of a .vti file with 2 data points

```
<?xml version="1.0"?>
<VTKFile type="PolyData" version="0.1" byte_order="LittleEndian" header_type="UInt64">
  <PolyData>
    <Piece NumberOfPoints="2" NumberOfVerts="2">
      <Points>
        <dataArray type="Float64" Name="position" NumberOfComponents="3" format="ascii">
          0 0 0
          20 20 0
        </dataArray></Points>
        <PointData><dataArray type="Float64" Name="mass" NumberOfComponents="1" format="ascii">
          1e+06
          100000
        </dataArray>
        <!-- TODO: velocity, acceleration -->
      </PointData>
      <Verts><dataArray type="Int64" Name="offsets">
        1 2
      </dataArray><dataArray type="Int64" Name="connectivity">
        0 1
      </dataArray>
    </Verts>
  </Piece>
  <FieldData>
    <dataArray type="Float64" Name="kinetic energy" NumberOfTuples="1" format="ascii">
      1.31546e-05
    </dataArray>
    <!-- TODO: potential energy, total energy -->
  </FieldData>
  </PolyData>
</VTKFile>
```

## Phase 1.6: Example of a .pvf file

Note: `timestep` does not correspond to the current time step number, but to the current simulation time!

Example: Time step size  $dt = 0.5$  and visualization step size  $vs = 10$

```
<?xml version="1.0"?>
<VTKFile type="Collection" version="0.1" byte_order="LittleEndian" compressor="vtkZLibDataCompressor">
  <Collection>
    <DataSet timestep="0.0" group="" part="0" file="time_series/sim.0.vtp"/>
    <DataSet timestep="5.0" group="" part="0" file="time_series/sim.1.vtp"/>
    <DataSet timestep="10.0" group="" part="0" file="time_series/sim.2.vtp"/>
    <DataSet timestep="15.0" group="" part="0" file="time_series/sim.3.vtp"/>
  </Collection>
</VTKFile>
```

## Phase 1.7: Running the Simulation

The simulation must be executable from the command line with the following parameters, which can be read in any order:

```
./simulate --file data.csv --dt 2.5 --t_end 100.0 --vs 5
```

Simulate every 2.5 years up to 100 years (40 time steps), with visualizations every 12.5 years (8 visualizations + the initial state).

```
./simulate. --vs 10 --dt 20 --t_end 4000 --file data2.csv
```

**-file** the simulation file to be used

**-dt** the time step size  $\Delta t$  in years

**-t\_end** the end time step in years

**-vs** the visualization step size

(example above: visualize every 5th or 10th time step)

## Phase 1.8: Validation of the Simulation

- In ILIAS<sup>✉</sup>, there is a 50\_ground\_truth.csv file with the simulation result of the 50 particle test case.
- Result can be reproduced with:  
`./simulate --file 50.csv --dt 100 --t_end 1000000`
- **Caution:** Due to floating-point inaccuracies, the results should not be tested for equality, but checked if they are sufficiently close (number of decimal places in the .csv file, etc.).

# Phase 1: Requirements

The following simulation must not take longer than 60 min on the bwCloud:

```
./simulate --file 4000.csv --dt 300 --t_end 6000000 --vs 200
```

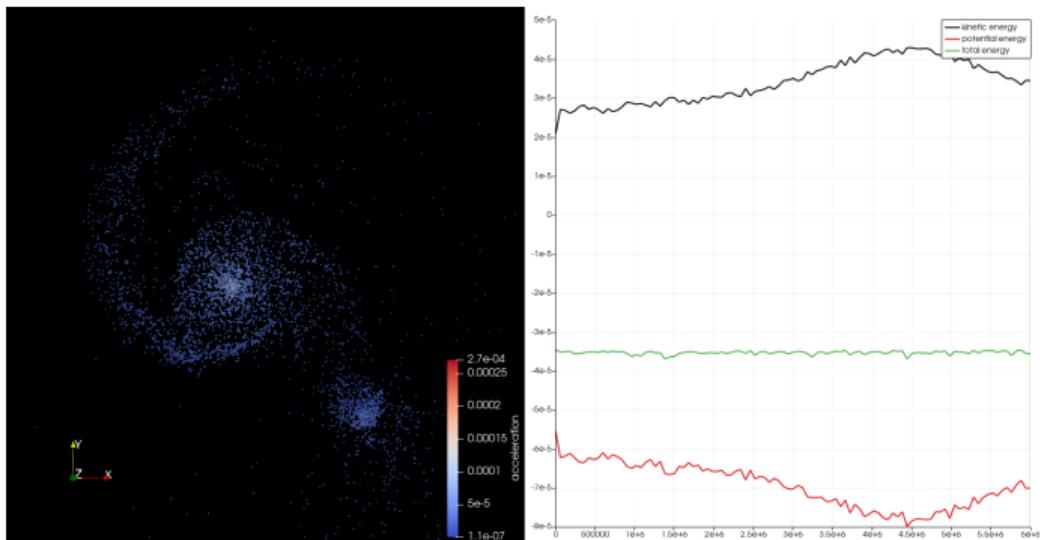


Figure: Simulation result after 20 000 time steps including energy development.

# Phase 1: Submission - 27.05.2025

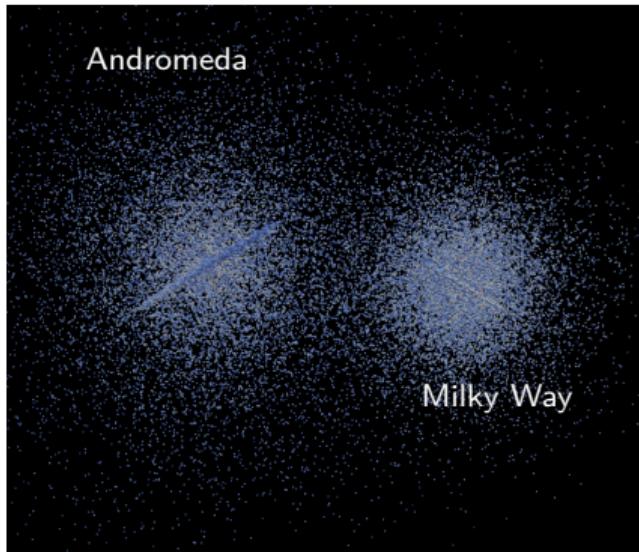
- Submit via ILIAS<sup>↗</sup>.
- Animation of the 3D simulation (20-30 s; e.g., ffmpeg<sup>↗</sup>) and image of the final time step created in ParaView, including an energy plot (simulation from the previous slide).
- A .csv file with the final state of the simulation.
- Do **not** submit the .pv and .vtp files!
- Include a file submission.sh with the following content (replace placeholders):

```
#!/bin/sh
git clone ${REPOSITORY_URL} ${GROUP_NAME}
cd ${GROUP_NAME} || exit
git checkout ${COMMIT_HASH}
```

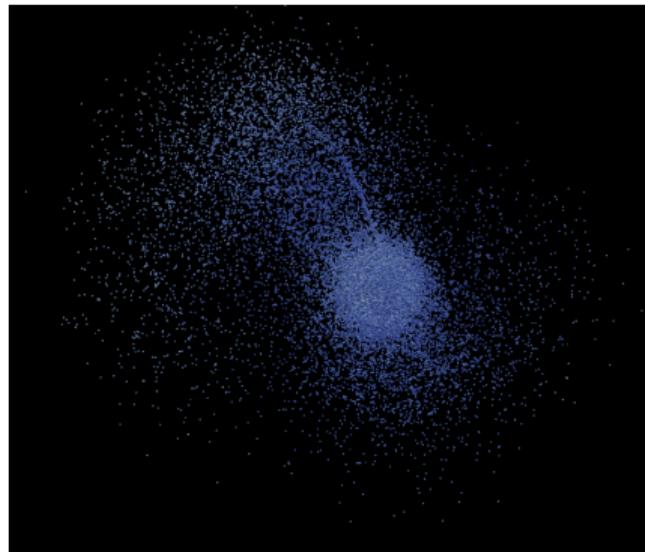
The \${REPOSITORY\_URL} should be the ssh link and **not** the https link! The commit must contain the code to be evaluated, as well as a README describing how to compile and run the code.

## Phase 2:

- Acceleration of the simulation using the tree-based Barnes-Hut algorithm.
- Simulation of the collision between the Milky Way and the Andromeda Galaxy with 81 920 particles.



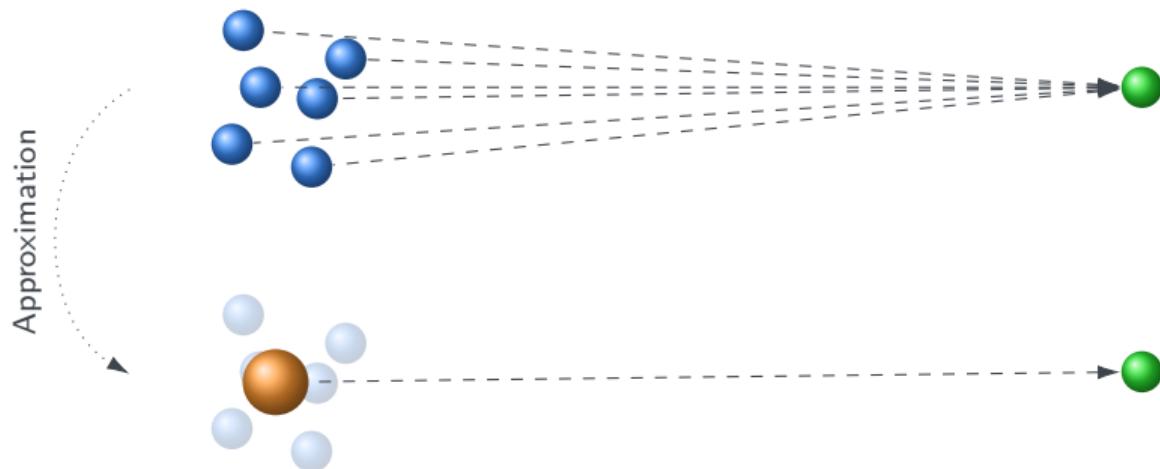
$t = 0$



$t = 120$

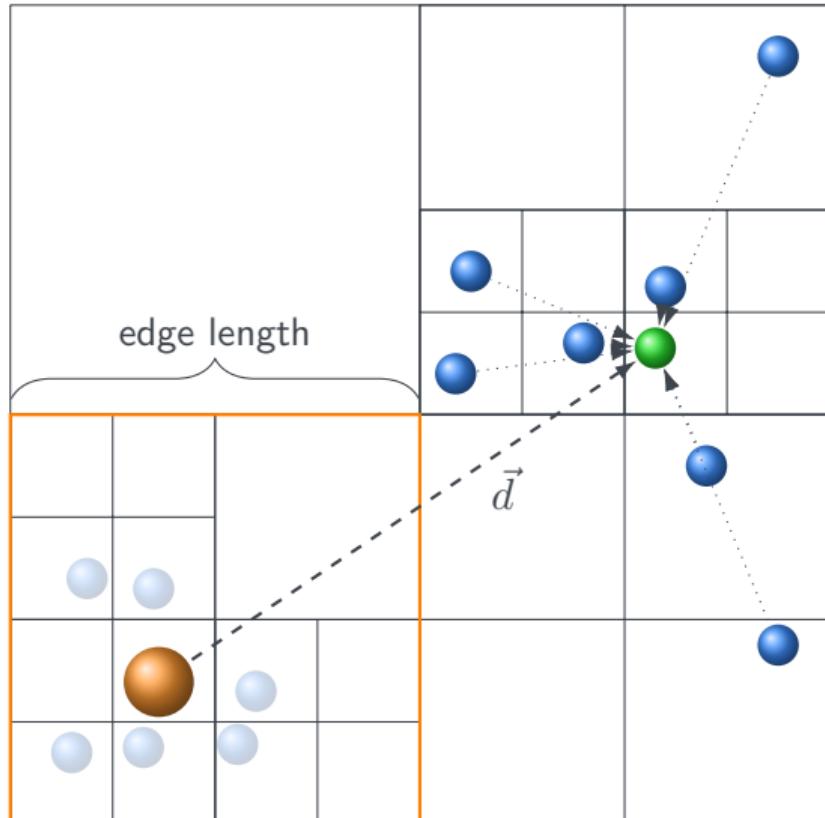
## Phase 2.1: Barnes-Hut

Summary of merging particles into pseudo-particles if they are far enough apart to reduce the number of force calculations.



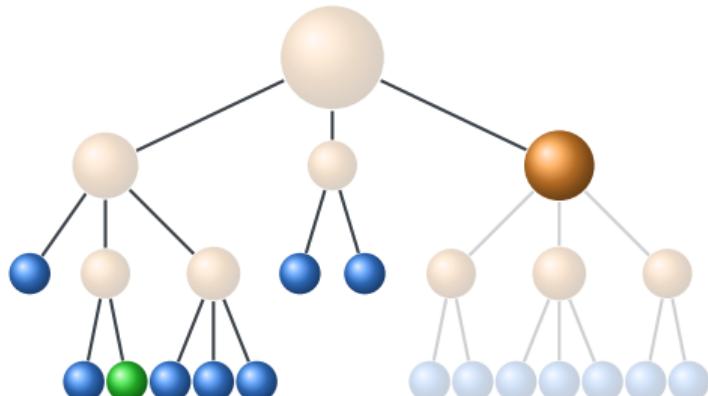
See: Josh Barnes and Piet Hut: A hierarchical  $O(N \log N)$  force-calculation algorithm (1986)  
(DOI: <https://doi.org/10.1038/324446a0>)

# Theory: Barnes-Hut - Example: Quadtree



Use pseudo-particles to calculate the forces if:

$$\frac{\text{edge length}}{\|\vec{d}\|_2} < \theta$$



naive: 14 force calculations

Barnes-Hut: 8 force calculations

## Phase 2.3: Barnes-Hut - Nodes

Each tree node of the octree (extension of the quadtree for 3-dimensional space) must store the following values:

- Edge length of the octree octant
- Total mass of all particles in the octant:  $m_{oct} = \sum_i m_i$
- Center of mass of all particles  $i$  in the octant:  $\vec{c}_{oct} = \frac{\sum_i m_i \vec{r}_i}{\sum_i m_i}$
- All children of the octant

Each octree leaf can contain at most one particle!

## Phase 2.4: Barnes-Hut - Updating Accelerations $\vec{a}_i$

---

```
1: procedure UPDATEACCELERATION(Particle, Node)
2:   Calculate direction vector  $\vec{d}$  between Particle and Node
3:   if Pseudo-particle can be used  $\vee$  Node is a leaf node then
4:     Update acceleration of  $\vec{a}_{\text{particle}}$ 
5:   else
6:     for each Child $_i$  of Node do
7:       UPDATEACCELERATION(Particle, child $_i$ )
8:     end for
9:   end if
10:  end procedure
```

---

### Question:

Can the energy calculation also be accelerated using the Barnes-Hut tree?

## Phase 2.5: Running the Simulation

The simulation must be executable from the command line as follows:

```
./simulate --file data.csv --dt 2.0 --t_end 100.0 --vs 5  
→ --algorithm_type 1 --theta 1.5
```

- file** the simulation file to be used
- dt** the time step size  $\Delta t$  in years
- t\_end** the end time step in years
- vs** the visualization step size
- algorithm\_type** the algorithm to be used
  - (0: naive brute-force algorithm; 1: Barnes-Hut algorithm)
- theta** the threshold in the Barnes-Hut algorithm

## Phase 2.6: Performance Analysis

- A scaling plot (double logarithmic axis scaling;  $x$ -axis: number of particles,  $y$ -axis: runtime) with varying number of particles for the naive algorithm and Barnes-Hut.
- Landau complexity estimation ( $\mathcal{O}$  notation) of your own code based on the scaling plot.
- A plot that compares different  $\theta$  values in the Barnes-Hut algorithm ( $x$ -axis) with their runtime ( $y_1$ -axis) and error (sum of Euclidean distances compared to the naive approach) ( $y_2$ -axis).
- Explanations with **justifications** for both plots!
- **Note:** Come up with suitable scenarios for both plots.

## Phase 2.6: Performance Analysis - Dataset Generation

Datasets with 2 to 200 000 particles can be generated using a `wget` or `curl` call (exemplary with 200 particles):

```
wget --user ${USERNAME} --ask-password  
↪ "https://ipvs.informatik.uni-stuttgart.de/SC/cgi-bin/n-body/generate.php?num_particles=200"  
curl --user ${USERNAME}  
↪ "https://ipvs.informatik.uni-stuttgart.de/SC/cgi-bin/n-body/generate.php?num_particles=200"  
↪ -o data.csv
```

Your username corresponds to the group name from Phase 0, and you will receive the randomly generated password from us after passing Phase 1.

## Phase 2: Requirements

The following simulations must not take longer than 60 min on the bwCloud:

```
./simulate --file 20000.csv --dt 300 --t_end 6000000 --vs 200 --algorithm_type 1 --theta 1.2  
./simulate --file dubinski.csv --dt 0.03 --t_end 120.0 --vs 20 --algorithm_type 1 --theta 1.5
```

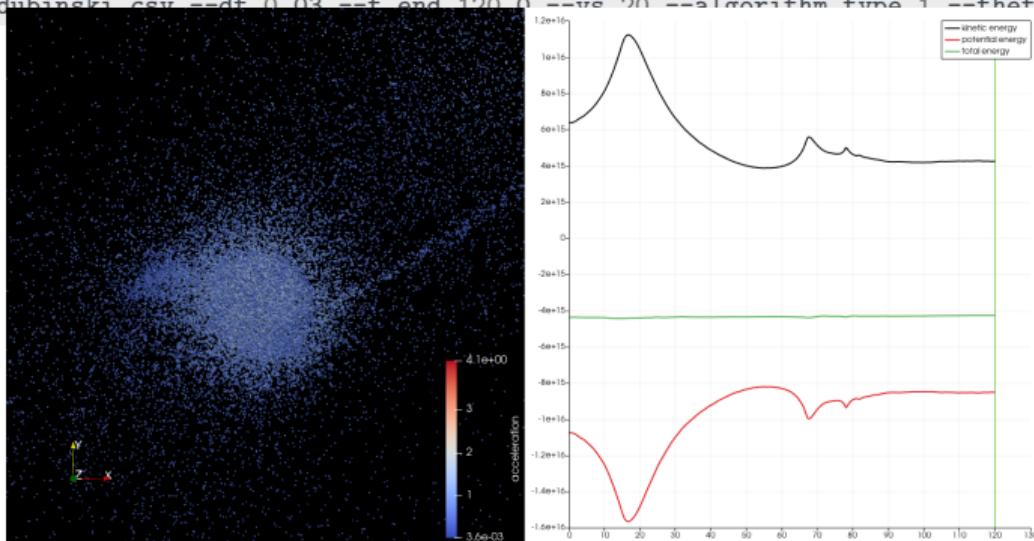


Figure: Simulation result of the Dubinski dataset after 4000 time steps including energy development.

## Phase 2: Submission - 15.07.2025

- Submission via ILIAS .
- Animations of the 3D simulations (each 20-30 s; e.g., ffmpeg ) and images of the final time steps created in ParaView, including the respective energy plot (simulations from previous slide).
- .csv files with the final states of the simulations.
- Do **not** submit the .pvda and .vtu files!
- A file submission.sh with the same content as in Phase 1 with the **adjusted** commit hash.
- Double logarithmic scaling plot **including justification** of the Landau complexities.
- Plot of the runtime behavior and accuracy of the Barnes-Hut algorithm with different  $\theta$  values **including justification** of the behavior.

# Conclusion

- Presentation max. 5 min per group.
- One slide with challenges you encountered.
- One slide each with the plots required in Phase 2.
- Slides must also be uploaded to ILIAS<sup>↗</sup>.
- Explanation of the plots, for example, based on your own code.
- Each group member must be able to answer questions about their own code!

# Competition - Who is the Fastest?

- Runtimes for the submissions of both phases.
- Enter your current runtimes (in seconds) in a table:  
Table: <https://ipvs.informatik.uni-stuttgart.de/cloud/s/4HbjLCQmfbenpWq>  
Password: wt2DKgJQWL
- Runtimes do not have to be entered only at the end but can be updated.
- The runtimes will be verified by us at the end.
- Real simulation without tricks (e.g., outputting pre-calculated time steps is **not** allowed)!
- The fastest group will receive a small prize.

# General Recommendations - 1

- Your simulation should have a progress indicator (e.g., console output every 10% of the simulation).
- Use an IDE (Visual Studio Code, CLion, Eclipse, etc.) and its features such as profilers, debuggers, auto-formatting, linters, etc.
- Use an automated build system like CMake.
- Document your code (for us **and** for yourself!) using tools like Doxygen .
- Make the installation, usage of your program, and verification of the requirements as easy as possible for the user.
- Test your program on the target system, the bwCloud (it must be runnable on Linux (keyword: paths: / vs \)!).
- Questions can be asked directly in the ILIAS forum .
- For **Windows users**, to avoid headaches due to different line endings on Linux and Windows, use : `git config --global core.autocrlf true`

## General Recommendations - 2

- Objects allocated with `new` are **not** automatically released (better: use `std::unique_ptr/std::shared_ptr` if necessary)!
- Be sure to set the necessary optimization flags (and to document them for us)!
- The runtime of your own code can be determined most easily using `time`:
- ~~time Barnes-Hut algorithm, update version that the target block hole isn't located at the intersection point of all eight quadrants.~~
- MIT insider tip: “The Missing Semester of Your CS Education” ↗

# Important Dates

08.04.2025 Kick-off

15.04.2025 Group formation deadline

20.05.2025 Kick-off Phase 2

27.05.2025 Phase 1 deadline

15.07.2025 Phase 2 deadline

TBA Final presentations

# Important Dates

08.04.2025 Kick-off

15.04.2025 Group formation deadline

20.05.2025 Kick-off Phase 2

27.05.2025 Phase 1 deadline

15.07.2025 Phase 2 deadline

TBA Final presentations

→ Don't forget to register in C@MPUS<sup>↗</sup>!