

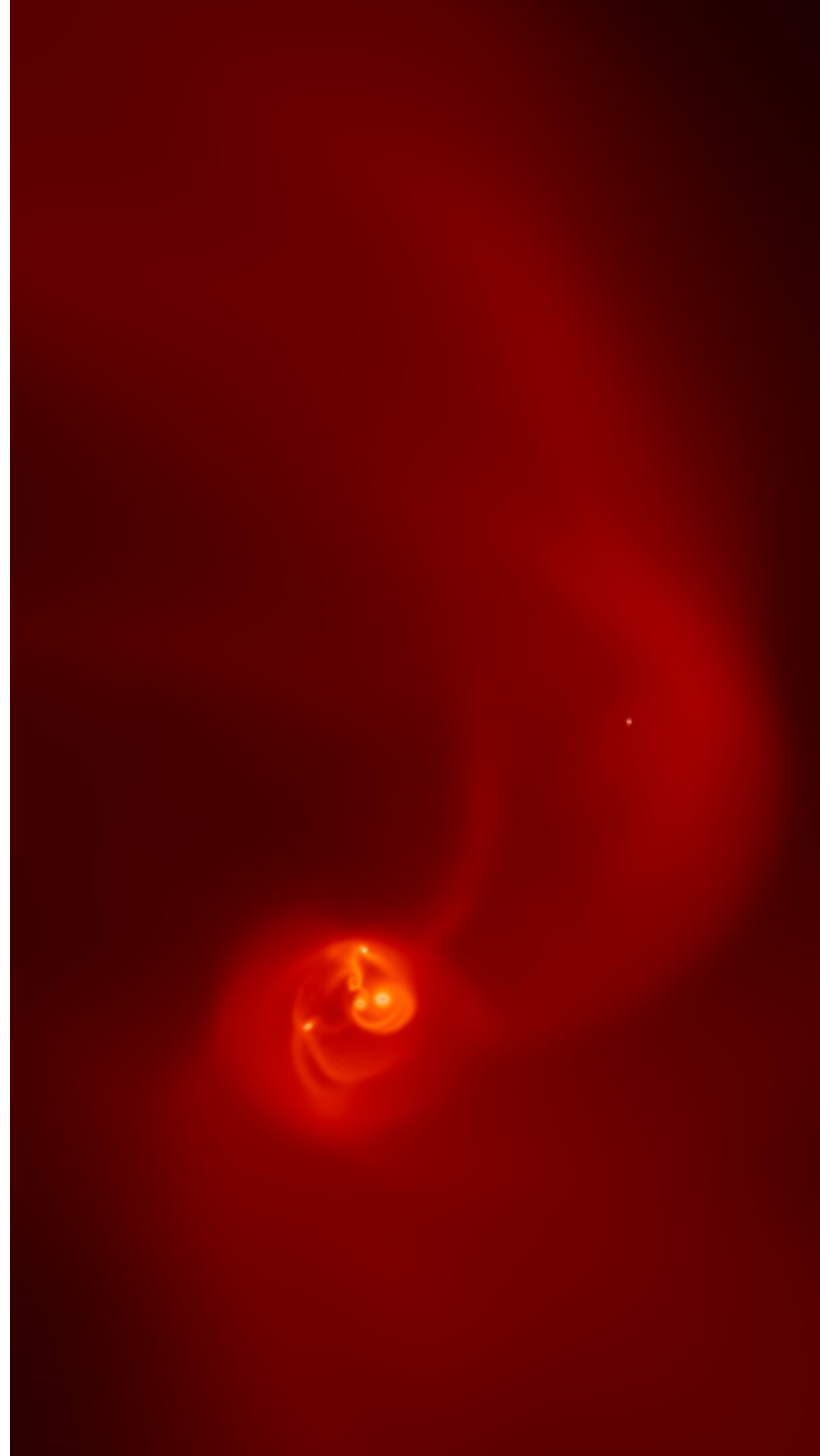
# A quick tour of GANDALF

---

David Hubber

USM, LMU, München  
Excellence Cluster Universe,  
Garching bei München

27th October 2015



Welcome to Freising and  
Thank You for coming



# What is GANDALF?

- The **G**raphical **A**strophysics code for **N**-body **D**ynamics **A**nd **L**agrangian **F**luids, aka **GANDALF**, is a new Particle Hydrodynamics code written in C++ written with Star and Planet Formation problems in mind
- As well as the main C++ code, GANDALF contains a python wrapper for easy use in normal python scripts

# Why bother writing a new code?

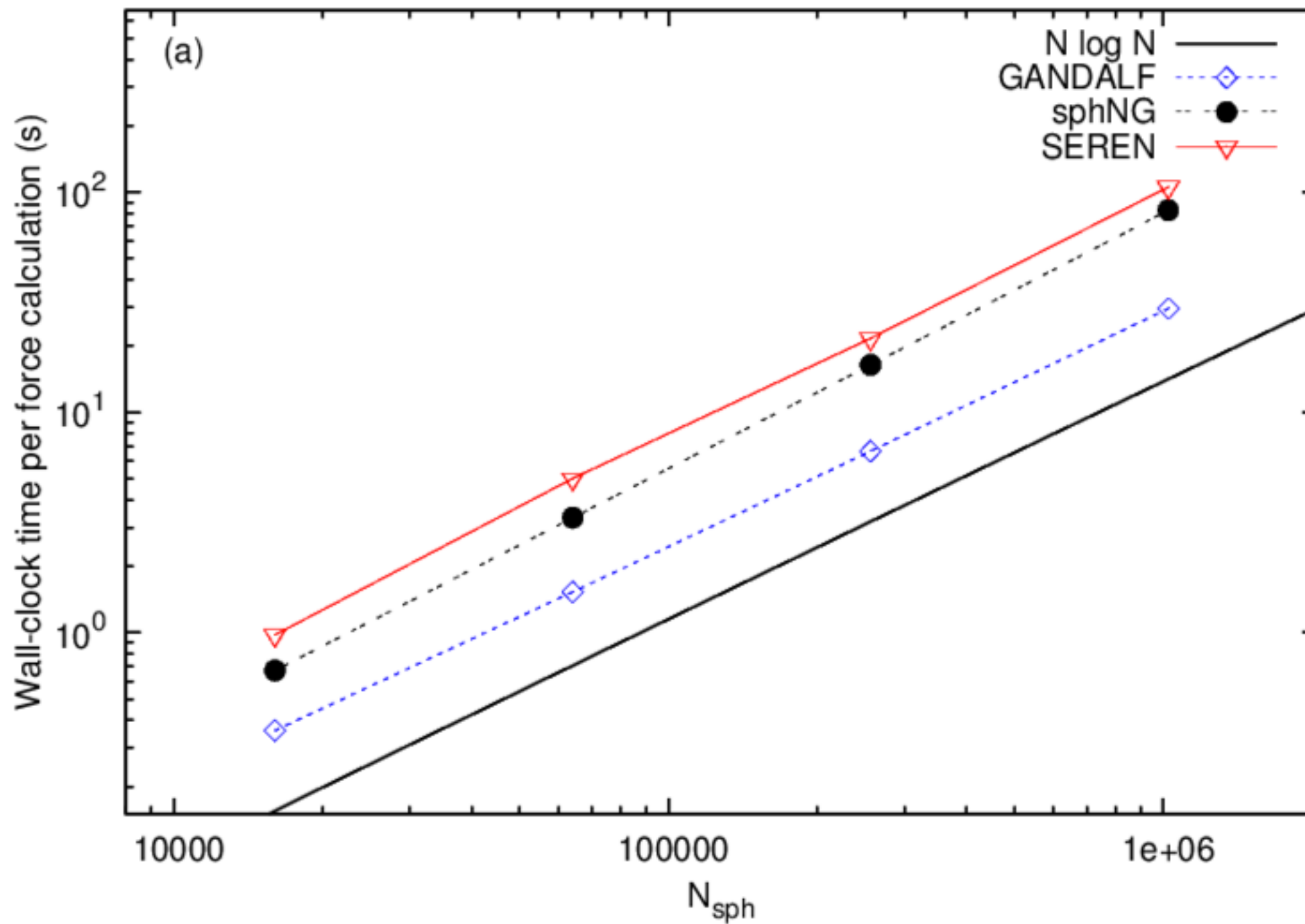
- Originally **no intention** of writing a new code
- Wanted to add a python wrapper to SEREN (Hubber et al. 2011) and add some additional physics in a more modular way
  - Python is rapidly becoming one of the most used languages in astronomy with packages such as *astropy* and *amuse*.
  - Both (particularly the python wrapper) were proving difficult in Fortran
- So we decided to start a new project in C++ as a successor code to SEREN while easily binding to python with swig

# Other motivations

- GANDALF (as well as SEREN) was originally written as a star and planet formation code
- Most particle hydrodynamical codes in astronomy (e.g. GADGET, GIZMO, VINE, GASOLINE, PHANTOM) are **NOT developed with star and planet formation in mind**
  - In particular, the parallelisation of most of these codes is developed with more large-scale uniform density fields like in Cosmology.
  - Star Formation density fields are more concentrated in a few small regions; **simple parallelisation schemes don't work so well**
- Such codes also don't contain important physics algorithms that are required for both, such as : Sink/star particles, accretion feedback, feedback from massive stars



# Other motivations



# Old and new algorithms

- GANDALF already contains a variety of tried-and-tested algorithms that should be part of any decent star formation code
  - Smoothed Particle Hydrodynamics
    - Conservative SPH
    - Various equations of state implemented
    - Can be run in 1D, 2D or 3D (Cartesian)
  - Stars (both N-body and sink accretion)
    - Fully conservative algorithm for integrating motion of gas and stars together with control over errors
  - Trees (for efficient gravity and neighbour searching)
    - Implemented KD-tree and Octal tree
    - Periodic Ewald gravity in 1, 2 and 3 dimensions
  - Simple thermal physics or radiative cooling algorithms

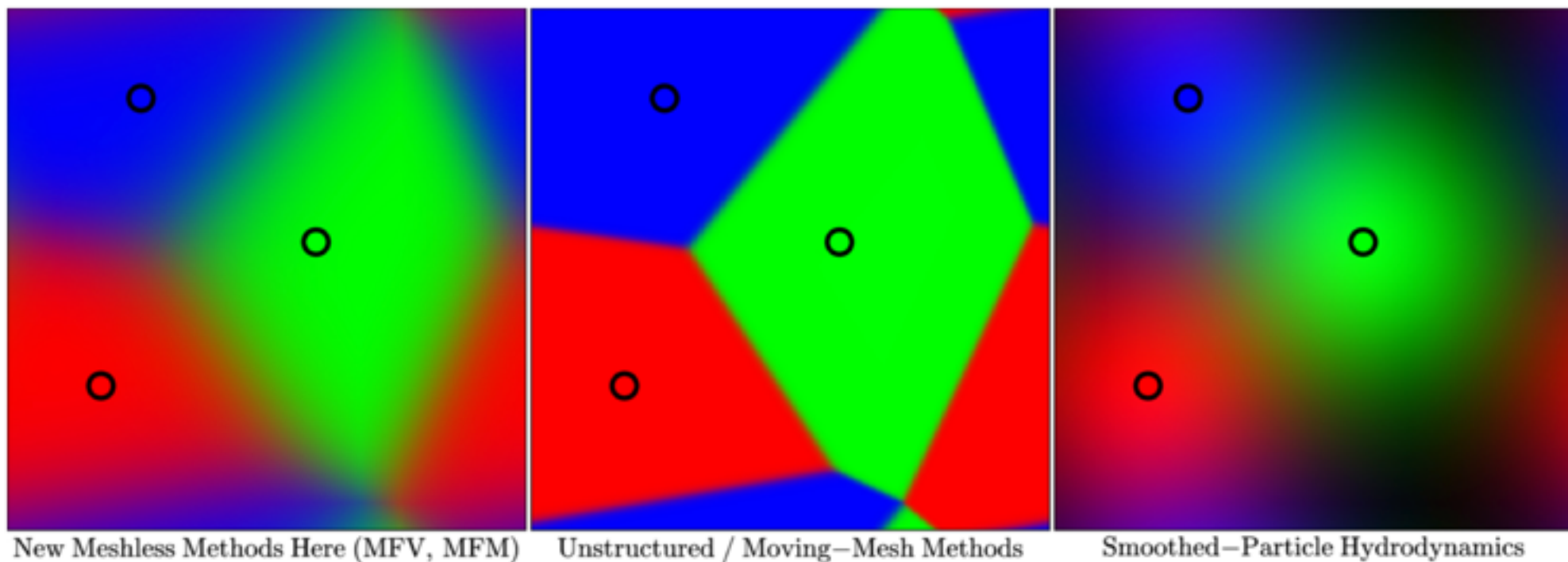
# Old and new algorithms

- Several new developments we have included/planning to include
  - Radiation feedback from stars (accretion luminosity + feedback)
    - TreeRay - Reverse ray-tracing to compute radiation field from all sources
  - Dust physics
    - Talk on Thursday (Richard)
  - Ideal MHD
  - New Hydrodynamical algorithms (see next slide)
    - Talk on Wednesday (Judith)



# Old and new algorithms

- Two recent methods could solve some of the various problems (affecting both grid and SPH) and become more important in future numerical investigations
  - **Voronoi moving-mesh** (e.g. AREPO, TESS)
  - **Meshless Finite-Volume** (e.g. APWHD, GIZMO)



Hopkins (2015)

# Coding philosophy of GANDALF

- We wished to design a hydrodynamicscode that could easily utilise a chosen algorithm from a selection present
- This functionality is best implemented by using the **polymorphic** qualities used in **object oriented programming**
- This is most easily described as having a family of algorithms that all share the same interface and so can be used by the same piece of code. There just needs to be a way of selecting which one to use
- For example, the thermal physics is controlled by a 'EOS' class
- There exist various different particular implementations, which are inherited child classes

## Class EOS

```
virtual float Pressure(SphParticle)  
virtual float SoundSpeed(SphParticle)  
virtual float Temperature(SphParticle)
```

## Class AdiabaticEOS

```
float Pressure(SphParticle)  
float SoundSpeed(SphParticle)  
float Temperature(SphParticle)
```

## Class BarotropicEOS

```
float Pressure(SphParticle)  
float SoundSpeed(SphParticle)  
float Temperature(SphParticle)
```

## Class IsothermalEOS

```
float Pressure(SphParticle)  
float SoundSpeed(SphParticle)  
float Temperature(SphParticle)
```

# Coding philosophy of GANDALF

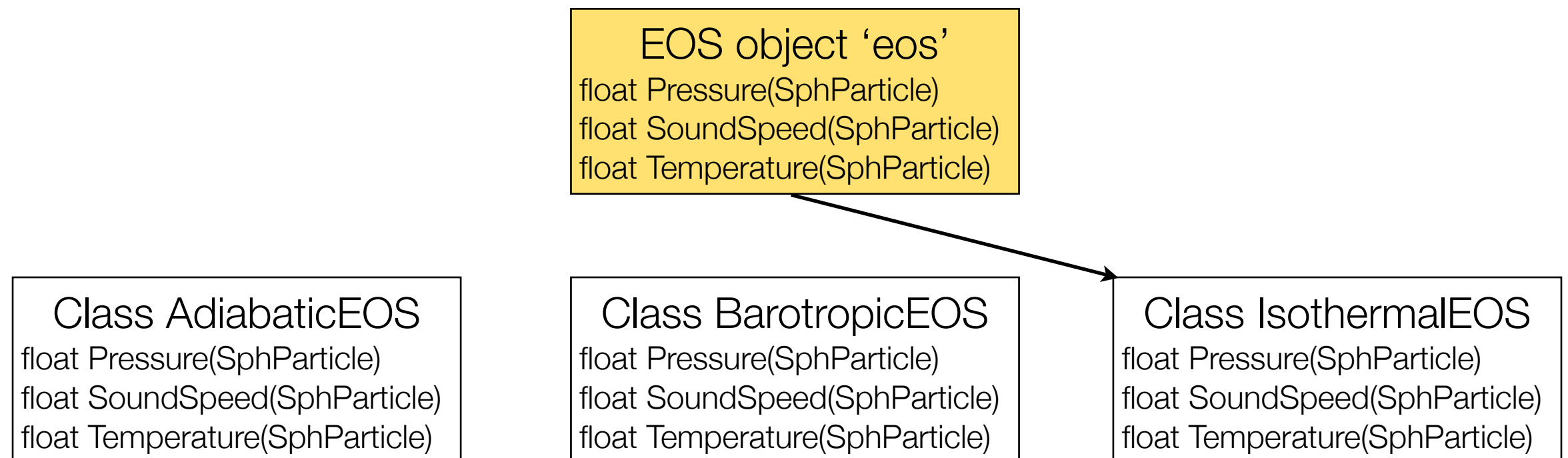
- The desired implementation to use is chosen in the parameters file

```
gas_eos = isothermal
```

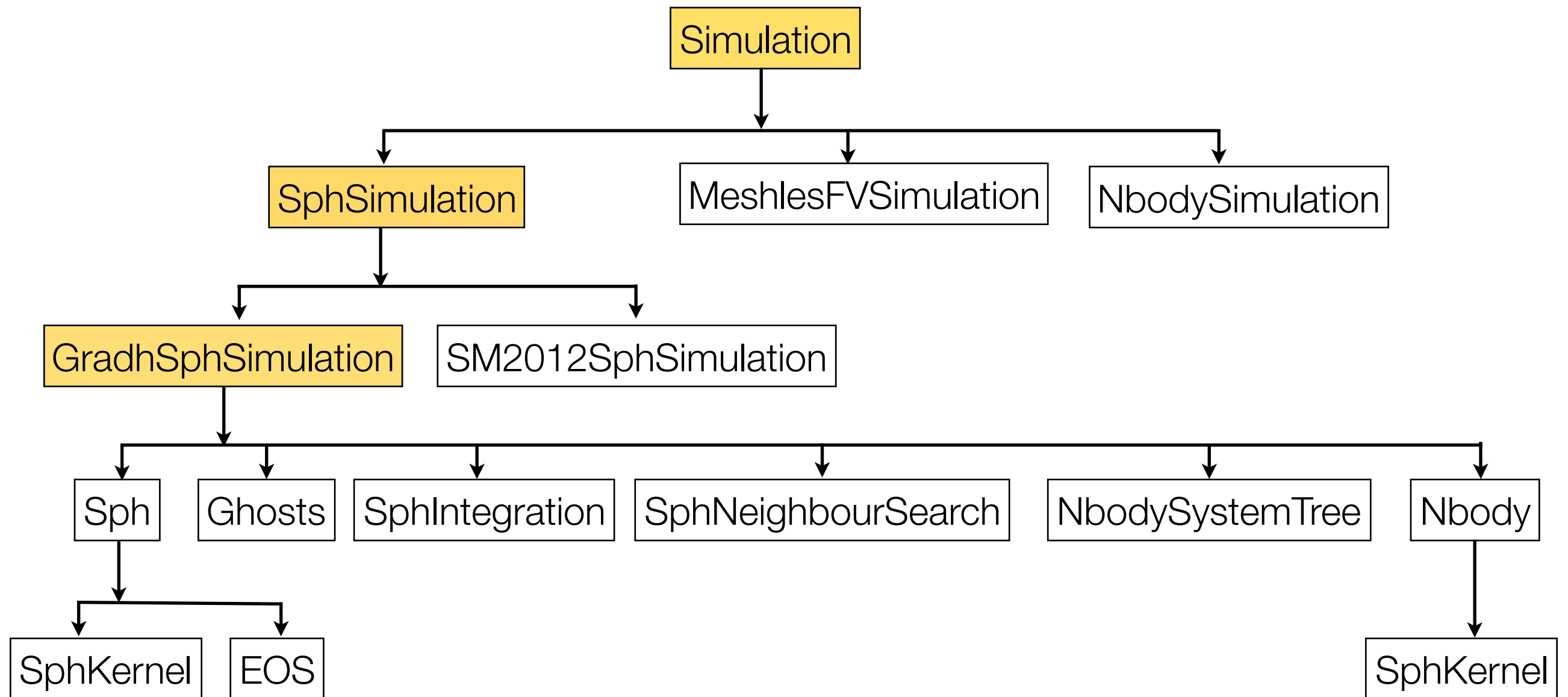
- The required codee object is then initialised in the code

```
if (gas_eos == "isothermal")  
    eos = new Isothermal();
```

- The eos pointer used throughout the code then 'points' to the correct implementation



# Coding philosophy of GANDALF



# Important classes in GANDALF

- Simulation
- Hydrodynamics
- NeighbourSearch
- Radiation
- EnergyEquation
- Nbody
- Sinks

# GANDALF mode 1 : Command-line

- Run GANDALF as a typical command-line executable with a parameter file

```
bin/gandalf adsod.dat > out.txt &
```

```
#-----  
# Adiabatic Sod shock tube test  
# Creates an adiabatic Sod shocktube test  
#-----  
  
#-----  
# Initial conditions variables  
#-----  
Simulation run id string           : run_id = ADSOD1  
Select SPH simulation              : sim = sph  
Select shocktube initial conditions : ic = shocktube  
1D shocktube test                  : ndim = 1  
x-velocity of LHS fluid            : vfluid1[0] = 0.0  
x-velocity of RHS fluid            : vfluid2[0] = 0.0  
Pressure of LHS fluid              : press1 = 1.0  
Pressure of RHS fluid              : press2 = 0.2  
Density of LHS fluid               : rhofluid1 = 1.0  
Density of RHS fluid               : rhofluid2 = 0.5  
No. of particles in LHS fluid      : Nlattice1[0] = 200  
No. of particles in RHS fluid      : Nlattice2[0] = 100  
Dimensionless units                : dimensionless = 1
```

- Most useful for running large jobs on large super-computers for maximum performance



# GANDALF mode 2 : Python script

- Write and run a short python script to perform all tasks

```
python adsodtest.py
```

```
#=====
# example01.py
# Basic example to run a simulation from a parameters file.
#=====
from gandalf.analysis.facade import *

# Create simulation object from parameters file
sim = newsim("adsod.dat")

# Perform all set-up routines and then run simulation to completion
setupsim()
run()
```

# GANDALF mode 2 : Python script

- Everything can be done in python (i.e. ICs, run simulation, analysis and plotting)

```
#=====
# example08.py
# Create initial conditions for SPH simulation inside the python script, and
# then run the simulation to completion while plotting results.
#=====
from gandalf.analysis.facade import *
import numpy as np
import time

# Set basic parameters for generating initial conditions
Nsph = 200
vfluid = 4.0
xmin = -1.5
xmax = 1.5

# Set uniform line of Nsph particles between the limits of xmin and xmax
# in local numpy arrays
deltax = (xmax - xmin) / Nsph
x = np.linspace(xmin + 0.5*deltax, xmax - 0.5*deltax, num=Nsph)
m = np.ones(Nsph)*(xmax - xmin)/Nsph

# Set velocities of shock-tube so v = vfluid for x < 0 and -vfluid for x > 0
vx = np.ones(Nsph)*vfluid
vx[x > 0.0] = -vfluid

# Create new 1D simulation object and set parameters
sim = newsim(ndim=1)
sim.SetParam('sim','sph')
sim.SetParam('ic','python')

sim.SetParam('gas_eos','isothermal')
sim.SetParam('Nsph',Nsph)
sim.SetParam('tend',0.2)
sim.SetParam('dt_snap',0.05)
sim.SetParam('dimensionless',1)
sim.SetParam('vfluid1[0]',vfluid)
sim.SetParam('vfluid2[0]',-vfluid)
sim.SetParam('boxmin[0]',xmin)
sim.SetParam('boxmax[0]',xmax)
sim.SetParam('run_id','SHOCKTUBE1')

# Call setup routines and import particle data
sim.PreSetupForPython()
sim.ImportArray(x,'x')
sim.ImportArray(vx,'vx')
sim.ImportArray(m,'m')
sim.SetupSimulation()

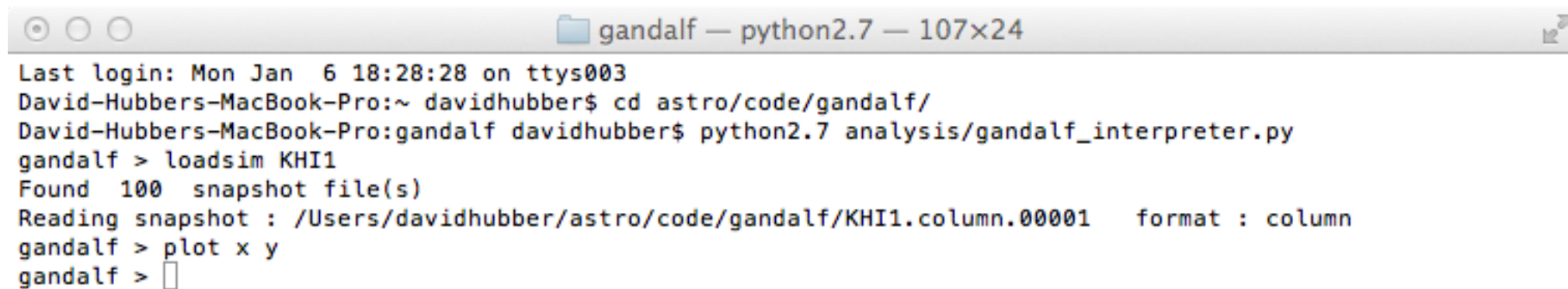
# Plot the density of all particles near the shock
plot("x","rho")
plotanalytical("x","rho",ic="shocktube")
limit("x",-0.17,0.17,window="all")
limit("rho",0,21.0,window="all")

# Run simulation and save plot to file
run()
savefig("shocktube.png")
block()
```

# GANDALF mode 3 : Interactive python

- Run python library as interactive shell

```
python analysis/gandalf_interpreter.py
```

A screenshot of a macOS terminal window titled "gandalf — python2.7 — 107x24". The terminal shows the following text:

```
Last login: Mon Jan 6 18:28:28 on ttys003
David-Hubbers-MacBook-Pro:~ davidhubber$ cd astro/code/gandalf/
David-Hubbers-MacBook-Pro:gandalf davidhubber$ python2.7 analysis/gandalf_interpreter.py
gandalf > loadsim KHI1
Found 100 snapshot file(s)
Reading snapshot : /Users/davidhubber/astro/code/gandalf/KHI1.column.00001  format : column
gandalf > plot x y
gandalf > 
```

# Git and Github

- We have used the 'git' version control software in developing and distributing the code online
- There'll be a short tutorial on using git later

The screenshot shows the GitHub interface for the repository 'gandalfcode / gandalf'. At the top, there's a navigation bar with 'Pull requests', 'Issues', and 'Gist' tabs. Below this, the repository name 'gandalfcode / gandalf' is displayed, along with 'Unwatch' (3), 'Star' (5), and 'Fork' (1) buttons. The main heading is 'GANDALF (Graphical Astrophysics code for N-body Dynamics And Lagrangian Fluids) — Edit'. Below the heading, a bar shows '1,080 commits', '6 branches', '1 release', and '2 contributors'. A green 'Branch: master' dropdown is visible. The commit history table lists the following entries:

Commit	Message	Time
giovanni-rosotti	Render does not crash anymore if called with quantities that it doesn...	Latest commit 596f4cc 2 days ago
analysis	Overhauled custom time function so that it behaves in the same way as...	2 days ago
bin	Added empty file to the bin directory to track it	3 years ago
docs	Added syntax highlighting in the user guide; the examples are now linke...	4 days ago
examples	Modified parameter files in the example folder so that they can run	4 days ago
papertests	Added text file describing the scaling tests we want to do	5 months ago

On the right side, there's a sidebar with links to 'Code', 'Issues' (20), 'Pull requests' (0), 'Wiki', 'Pulse', 'Graphs', and 'Settings'.

# Git and Github

- Different development versions of the code are held on different git **branches**
  - **master** branch (main version of the code)
  - **development\_mpi**
  - **multispecies**
- New branches may appear for developing new features before being merged into the master branch and deleted
- You'll be encouraged to create your own branches locally when adding your own new features to the code
- Makes it much easier to not mix up the main working version of the code with your own changes

# Contributions to the code

- If you'd like to 'donate' any features to the publicly available version of the code on github, then a few important things :
- GANDALF has been released on the Gnu Public Licence version 2 (GPL v2)
  - This means the code is available for anyone to take, modify and distribute themselves
  - However, any project using GPL v2 code must ALSO be GPL v2
- If you'd like to contribute, then please e-mail either the code e-mail address ([gandalfcode@gmail.com](mailto:gandalfcode@gmail.com)), myself ([david.hubber@gmail.com](mailto:david.hubber@gmail.com)) or Giovanni ([rosotti@ast.cam.ac.uk](mailto:rosotti@ast.cam.ac.uk))
- Also, we hope you would please cite the GANDALF paper (when it is finally actually published!)



# Structure of this meeting

- Contributed talks, talks about physics and algorithms are mainly concentrated in the morning sessions
- Tutorials and practicals are mainly concentrated in the afternoon sessions
- Tea/coffee breaks are 30 - 40 minutes long, giving plenty of time to
  - finish off any practicals from the previous session ready for the next one
  - looking at the posters
  - ask any additional questions
  - chatting amongst yourselves

# Practical sessions

- The tutorials/practicals will usually consist of a short talk followed by a series of small practical 'tasks' which you can do on your laptop with the assistance of the LOC members
- Please ask for help from any of the LOC members at any time
- And **please feel free to ask questions during the talks**

