Michela Mapelli

N-body techniques for astrophysics: Lecture 4 – STARLAB

PhD School in Astrophysics,
University of Padova
November 19-30, 2018

Michela Mapelli

N-body techniques for astrophysics: Lecture 4 – STARLAB

MOST IMPORTANT INGREDIENT: stay calm, breath, don't panic, don't kill yourself, don't kill your office mates

PhD School in Astrophysics,
University of Padova
November 19-30, 2018

OUTLINE:

1* definition and structure of starlab
http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/

2* the dynamics: KIRA
http://www.sns.ias.edu/~starlab/kira/

3* the stellar evolution: SEBA

4* the outputs
http://www.sns.ias.edu/~starlab/internals/

5* compilation and installation

6* writing initial conditions
http://www.sns.ias.edu/~starlab/examples/

7* running kira

8* visualize outputs

1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/

* Software environment:
  multiple codes to generate, evolve and analyze a collisional system (such as a star cluster)
  They can be combined

* Tools to
  Monte Carlo generate initial conditions (e.g. makeplummer)
  Evolve stars and binaries via population synthesis (e.g. SeBa)
  Evolve collisional dynamics (e.g. kira)
  Analyze outputs (e.g. xstarplot)

* Collisional dynamics is evolved through HERMITE SCHEME with block time steps: example of code adopting tools described in lecture 3

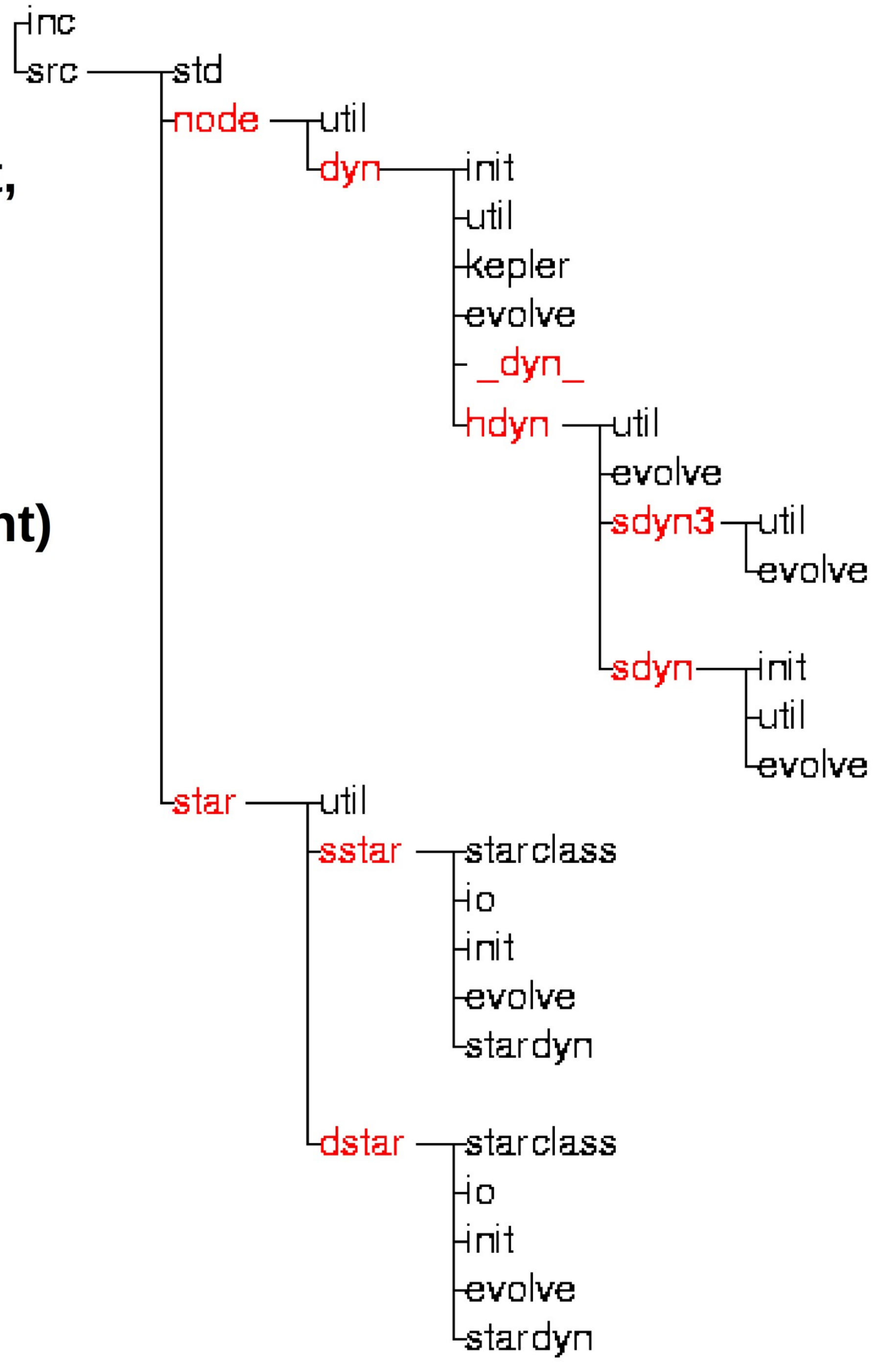
1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/

*** not a code but a software environment,**
a collection of modular software tools:
generate ICs (plummer, king),
dynamics, stellar evolution,
binary evolution,
plot tools (better not use),
analysis tools (statistics..some important)

***c++, something in fortran (DON'T USE)**
→ CLASSES!!!

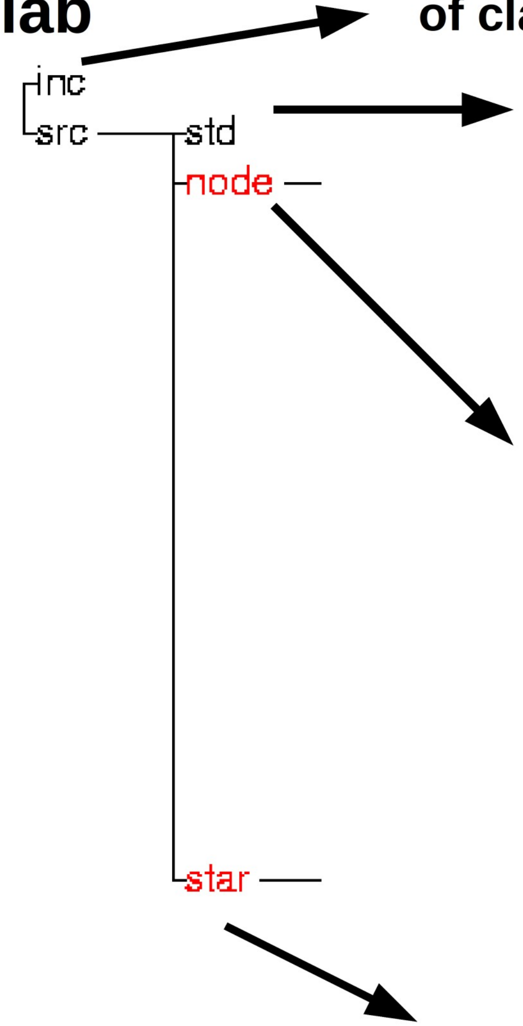
***complex, directory structure:**



1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/

***complex, directory structure:**



Include: *.h definition of classes

Common material (e.g. random generators)

Dynamics!

Stars!

1) definition and structure of starlab

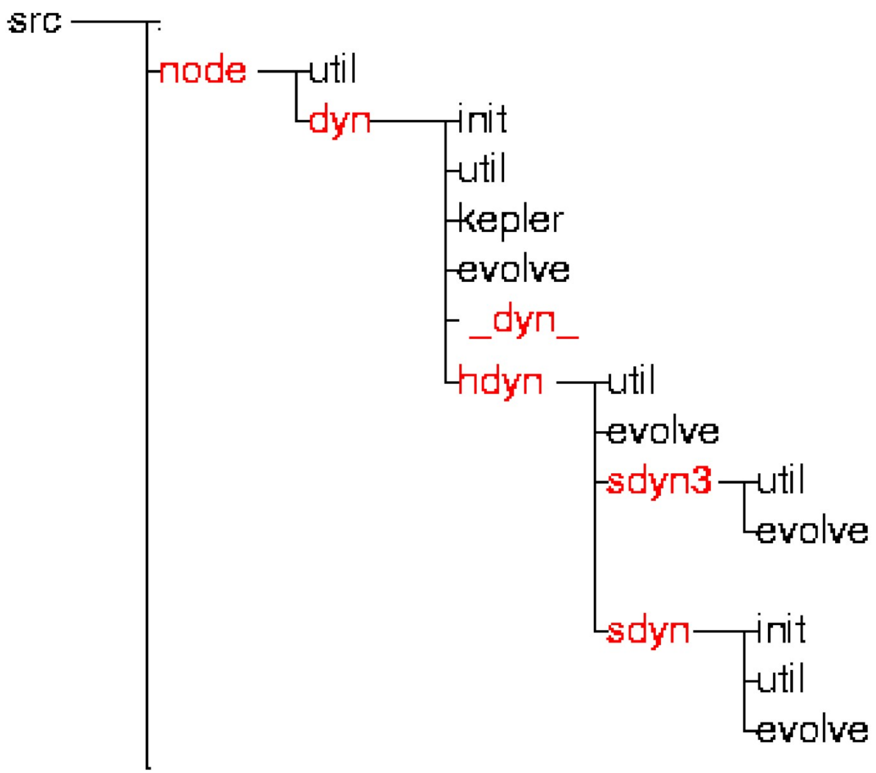
http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/

*** dynamics:**

init: contain tool for initialization

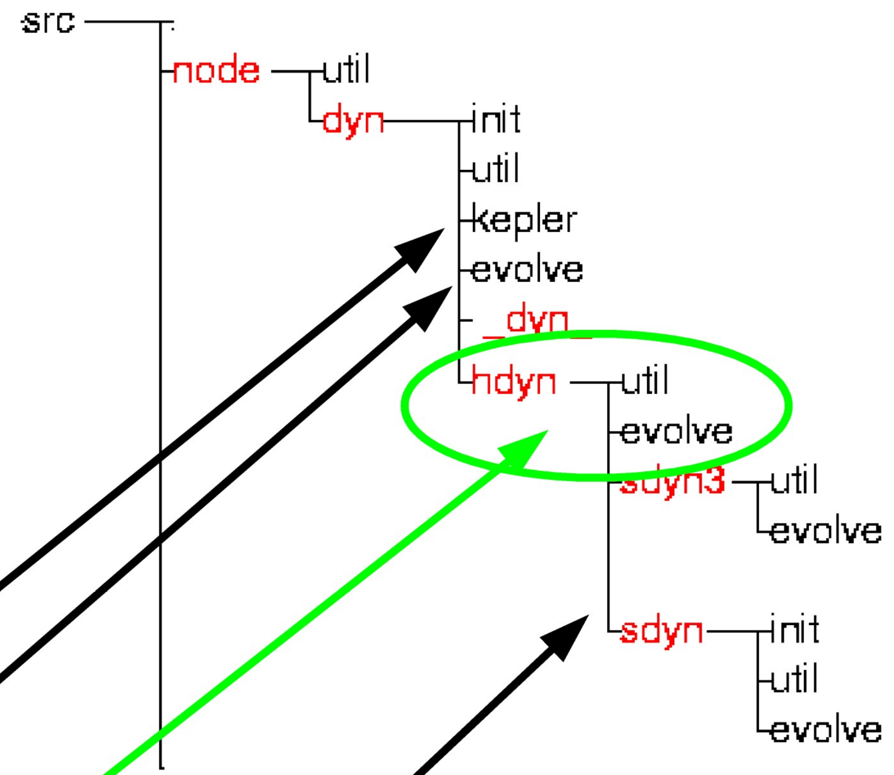
util: data analysis or plot

evolve: evolve dynamics in time



1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/



*** dynamics:**

init: contain tool for initialization
(src/node/dyn/init/makeking.C)

util: data analysis or plot

evolve: evolve dynamics in time

Kepler: only 2-body Keplerian

Only leapfrog

HDYN: high-res dynamics
KIRA INTEGRATOR
./src/node/dyn/hdyn/evolve/kira.C

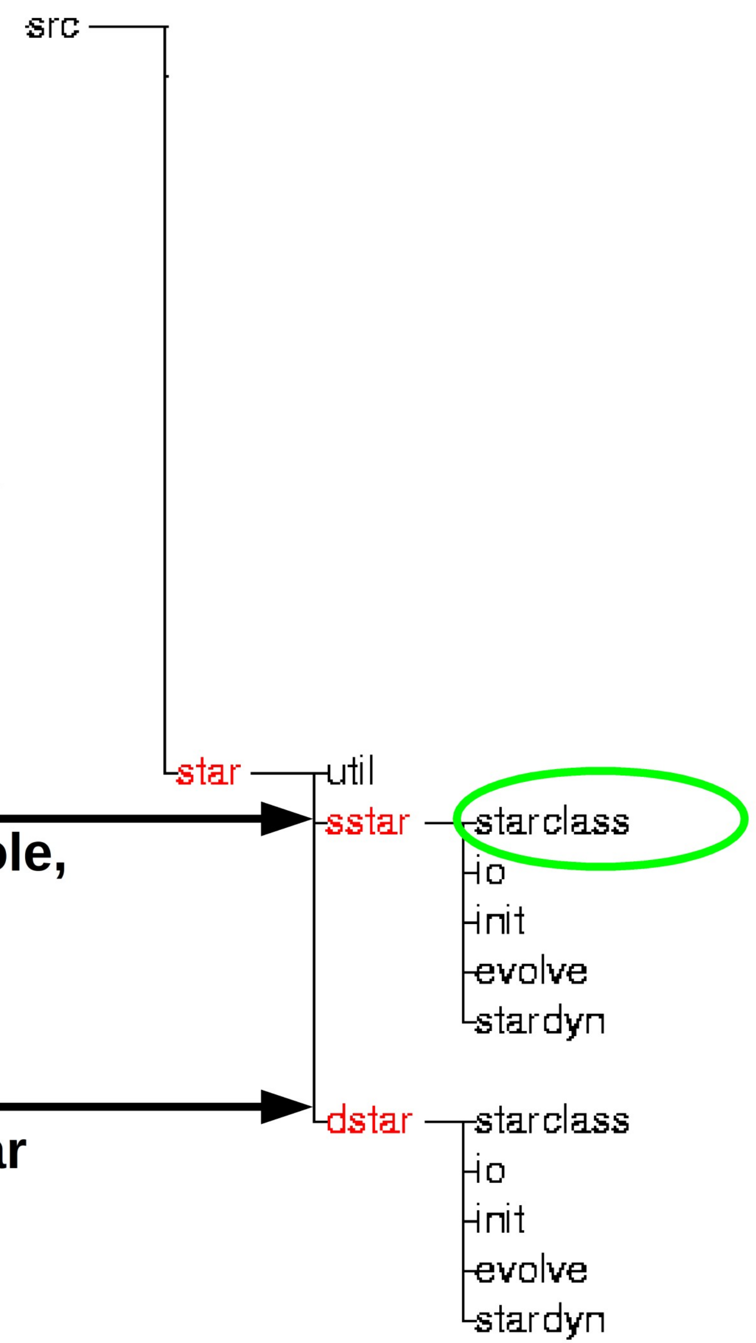
only 3-body scattering

1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/overview/
http://www.sns.ias.edu/~starlab/structure/

*** stars:**

- **init: contain tool for initialization**
- **util: data analysis or plot**
- **evolve: evolve in time star or binary**
- **io: input output of star data**
- **sstar: single stars**
  - **Class: single star,**
  - **derived class: MS star, black hole, hyper-giant, etcetc**
  - **In starclass/**
- **dstar: double star**
  - **starclass: only class double star**



1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

- CLASS: description of structure+ its functions
- an OBJECT belongs to a class if it is DEFINED as member of the class → star a;

EACH PARTICLE + root belongs to the node class (include/node.h)

```
class node {
        static node* root;  // Global address of the root node.
        long int node_flag; // Indicator of valid node (for internal
                            // bookkeeping purposes only)
        int index;          // Nodes can be numbered,
        char * name;        // or they can receive individual names.
        real mass;
        node * oldest_daughter;    // Define the node's place in
        node * elder_sister;       // the tree.
        node * younger_sister;
        story * log_story;  // Log story is a generalized scratchpad.
        story * dyn_story;  // The dyn story is a placeholder for
                            // dynamical information not recognized by
                            // a program -- this allows the information
                            // to be preserved and passed down a pipe.
        hydrobase * hbase;  // hydrobase is the class underlying all
                            // classes that handle hydrodynamics.
        starbase * sbase;   // starbase is the class underlying all
                            // classes that handle stellar evolution.
    }
```

1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

EACH PARTICLE + root belongs to the node class

If dynamics is defined, the dyn class is derived from node (include/dyn.h) HEREDITARIETY

```
class dyn : public node {
        static real system_time;
        static bool use_sstar;  // Single star evolution if true.
        vector pos;             // Position (3-D Cartesian vector).
        vector vel;             // Velocity: (d/dt) pos.
        vector acc;             // Acceleration: (d/dt) vel.
        kepler * kep;           // Pointer to a kepler orbit object.
    }
```

NB: mass belongs to node, pos, vel, acc only to dyn

1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

EACH PARTICLE + root belongs to the node class

If dynamics is defined, the dyn class is derived from node

If high-res hdyn class is derived from _dyn_ which is derived from dyn (include/hdyn.h, include/_dyn_.h)

```
class _dyn_ : public dyn {
        real time;                  // Individual particle time
        real timestep;              // and time step.
        real pot;                   // Potential.
        vector jerk;                // (d/dt) acc
        vector pred_pos;            // Predicted variables for use in the
        vector pred_vel;            // standard predictor-corrector scheme.
        real t_pred;                // Time of prediction.
        real radius;                // Effective (or actual) radius.
    }
```

1) hdyn

Tidal field

Binary evolution

Time dynamical Integration (e.g. softening)

Removal of escapers

Infos on perturbers (see kira)

```
class hdyn : public _dyn_ {
        //-----------------------------------------------------------------
        // Global variables:
        // Tidal field:
        static int  tidal_type; // none, point-mass, halo, disk
        static real alpha1;     // tidal field is conventionally taken
        static real alpha3;     // to be (-alpha1*x, 0, -alpha3*z)
        static real omega;      // system angular speed
        // Binary evolution:
        static bool use_dstar;  // binary evolution if true
        // Stellar encounters and mergers:
        static real stellar_encounter_criterion_sq;
        static real stellar_merger_criterion_sq;
        static real stellar_capture_criterion_sq;
        // Run-time integration parameters:
        static real eta;        // time step parameter
        static real eps;        // softening length
        static real d_min_sq;   // scale term governing tree adjustment
        static real lag_factor; // squared hysteresis factor
        static real mbar;       // mass scale
        static real gamma2;     // squared threshhold for unperturbed motoin
        static real gamma23;    // gamma^{-2/3}
        static real initial_step_limit; // limit on first time step
        static real step_limit;         // limit on all time steps
        // Escaper removal:
        static real scaled_stripping_radius; // stripping radius for unit mass
        //-----------------------------------------------------------------
        // Variables for unperturbed motion:
        real perturbation_squared;   // Relative perturbation squared.
        real unperturbed_timestep;   // Time step for unpert. motion.
        bool fully_unperturbed;      // True if orbit is fully
                                     // unperturbed.
        // Perturber information:
        int n_perturbers;            // Number of perturbers.
        hdyn** perturber_list;       // Pointer to perturber array.
        bool valid_perturbers;       // True if any particle is
                                     // within the perturbation
                                     // radius and the perturber
                                     // list has not overflowed.
        // Other neighbor information:
        hdyn* nn;                    // Pointer to nearest neighbor.
        real d_nn_sq;                // Distance squared to nn.
        hdyn* coll;                  // Pointer to neighbor whose
                                     // surface is closest to this node.
        real d_coll_sq;              // Distance squared to coll.
        // HARP-3 variables:
        int harp3_index;             // HARP-3 address of this particle.
        real harp3_rnb_sq;           // HARP-3 neighbor sphere radius.
    }
```

1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

Basic class for stars is starbase (include/starbase.h, for root):

```
class starbase {
        node  * the_node;                    // pointer to associated node
        story * star_story;                  // pointer to star story

        static real m_conv_star_to_dyn;      // mass conversion factor
        static real r_conv_star_to_dyn;      // length conversion factor
        static real t_conv_star_to_dyn;      // time conversion factor
        static bool use_hdyn;                // true iff binary evolution
                                             // is enabled
    /*mmapelli add on December 30 2012*/
        static real starmetal; /* default is solar metallicity*/
    /*mmapelli add on December 30 2012*/
    }
```

1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

Basic class for stars is starbase (for root)

For each particle, star class is derived from starbase (include/star/star.h)

```
class star : public starbase {
        // No private or
protected data...
        public:
        .
        .
        .
    }
```

1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

Basic class for stars is starbase (for root)

For each particle, star class is derived from starbase

If star evolution, single_star class is derived from star (include/star/single_star.h)

```
class single_star : public star {
        int identity;
        stellar_type star_type;
 // main sequence,

 // red giant, etc.
        star_type_spec spec_type[no_of_spec_type];
 // spectral type
        real current_time;
        real relative_age;
        real last_update_age;
        real next_update_age;
        real relative_mass;
        real envelope_mass;
        real core_mass;
        real radius;
        real core_radius;
        real effective_radius;
        real luminosity;
    }
```

1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

Basic class for stars is starbase (for root)

For each particle, star class is derived from starbase

If star evolution, single_star class is derived from star

Each stellar type derives from single_star
E.g. main_sequence (in include/star/main_sequence.h)

```
class main_sequence : public
single_star {

        real main_sequence_core_mass();
        real
main_sequence_core_radius();
        void adjust_donor_age(const
real mdot);
    }
```

1) definition and structure of starlab

http://www.sns.ias.edu/~starlab/structure/

Basic class for stars is starbase (for root)

For each particle, star class is derived from starbase

If star evolution, single_star class is derived from star

If binary evolution, double_star class is derived from star (include/star/double_star.h)

```
class double_star : public star {
        real semi;
        real eccentricity;
        binary_type bin_type;
        int identity;
        real binary_age;
        real minimal_timestep;
        int donor_identity;
        stellar_type donor_type;
        real donor_timescale;
        mass_transfer_type
current_mass_transfer_type;
    }
```

NB: single_star is associated with leaves, double_star with parent (kira!)

2) kira

http://www.sns.ias.edu/~starlab/kira/
based on 4th order Hermite with corrector/predictor

STEPS:

1. determines which stars need to be updated

2. checks for: reinitialization, log output, escaper removal, termination, snapshot output

3. perform low-order prediction (grape)

4. calculates acceleration/jerk and correct position/velocities (grape)

5. checks for all unperturbed motion

6. checks for collisions and mergers

7. checks tree reorganization

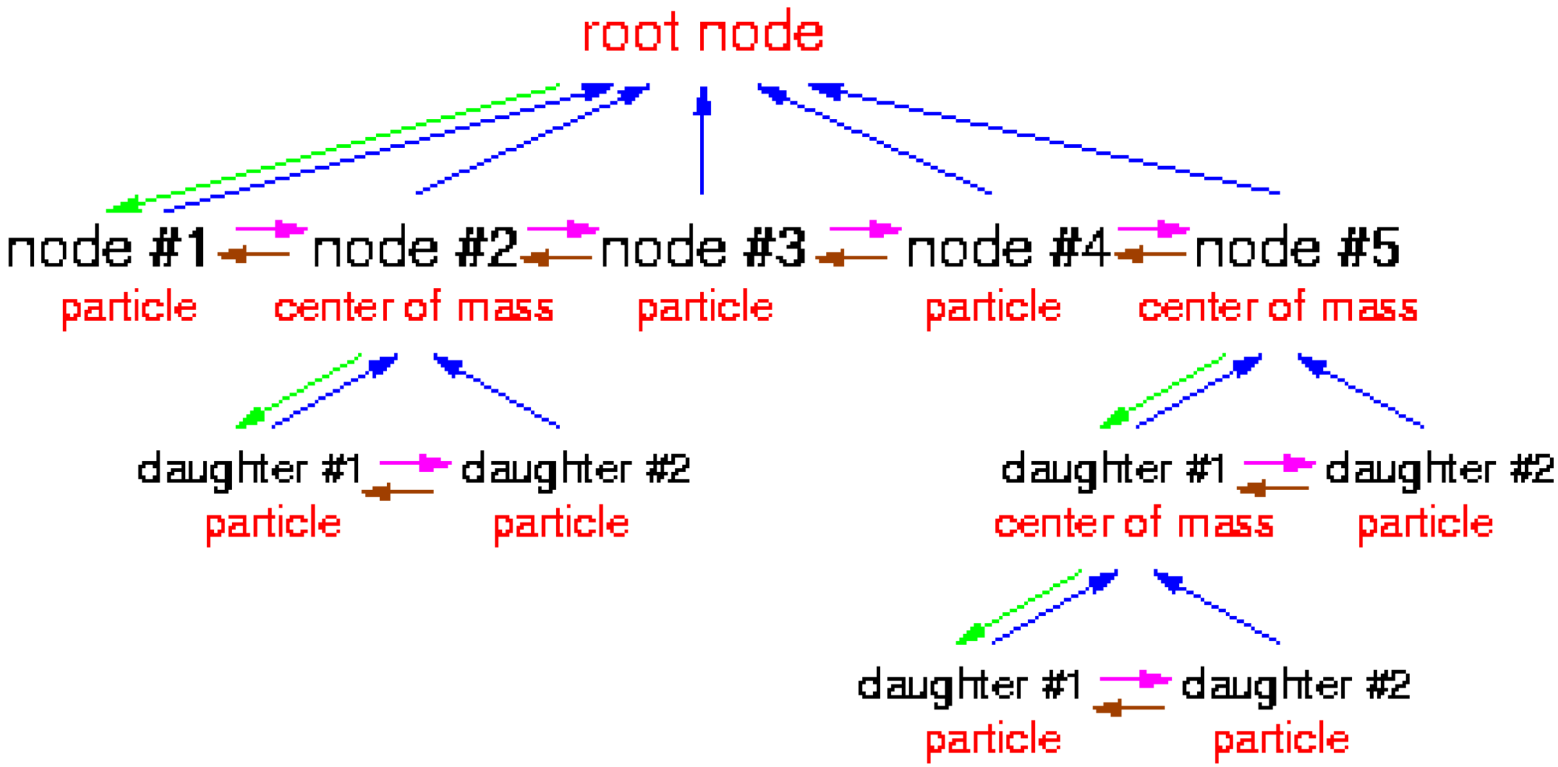
8. checks for stellar/binary evolution

2) **kira**

http://www.sns.ias.edu/~starlab/kira/

based on 4th order Hermite with corrector/predictor

TREE simpler than tree code: leaves are single stars, parents can be binaries or multiples, no more



Forces are computed using direct summation over all other particles in the system; no tree or neighbor-list constructs are used!!!

NO O(N logN)

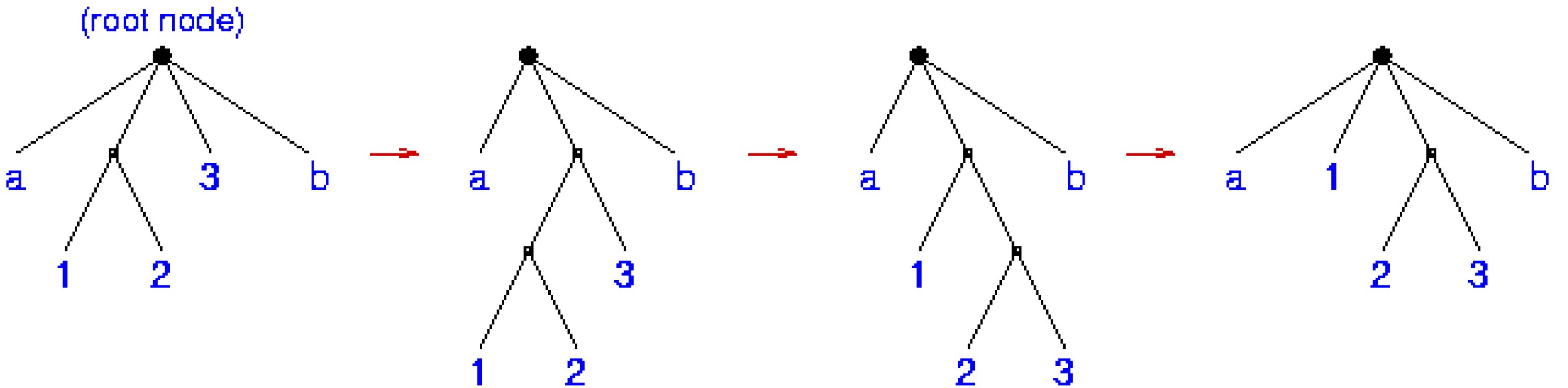
2) kira

http://www.sns.ias.edu/~starlab/kira/

based on 4th order Hermite with corrector/predictor

TREE simpler than tree code: leaves are single stars, parents can be binaries or multiples, no more

Example of a 3-body encounter



PERTURBED binaries (3-body) are splitted into components

UNPERTURBED binaries are evolved ANALYTICALLY

Critical point: how to decide perturber list!!!

2) kira

http://www.sns.ias.edu/~starlab/kira/
based on 4th order Hermite with corrector/predictor

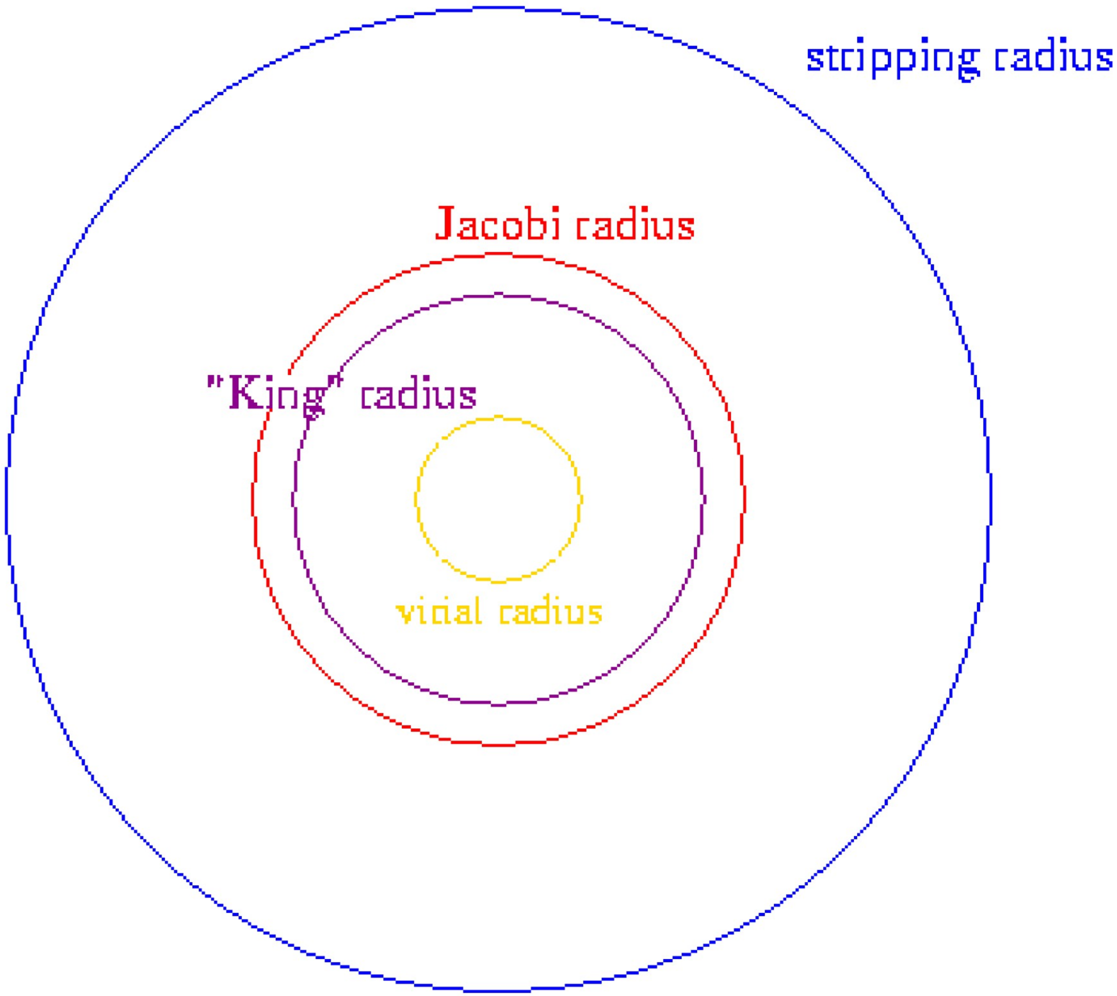
ESCAPER REMOVAL (not in current simulations)

*** Virial radius**

*** King radius:**
cutoff of King model

*** Jacoby radius:**
tidal radius

***stripping radius:**
radius for escaper removal
(eg 2 Jacobi)



3) the stellar evolution: SEBA
http://www.sns.ias.edu/~starlab/seba/
Portegies Zwart & Verbunt 1996

* Particles in collosional dynamics usually coincide with STARS:

  - have stellar mass (~ msun, distributed according to IMF)

  - have stellar radius

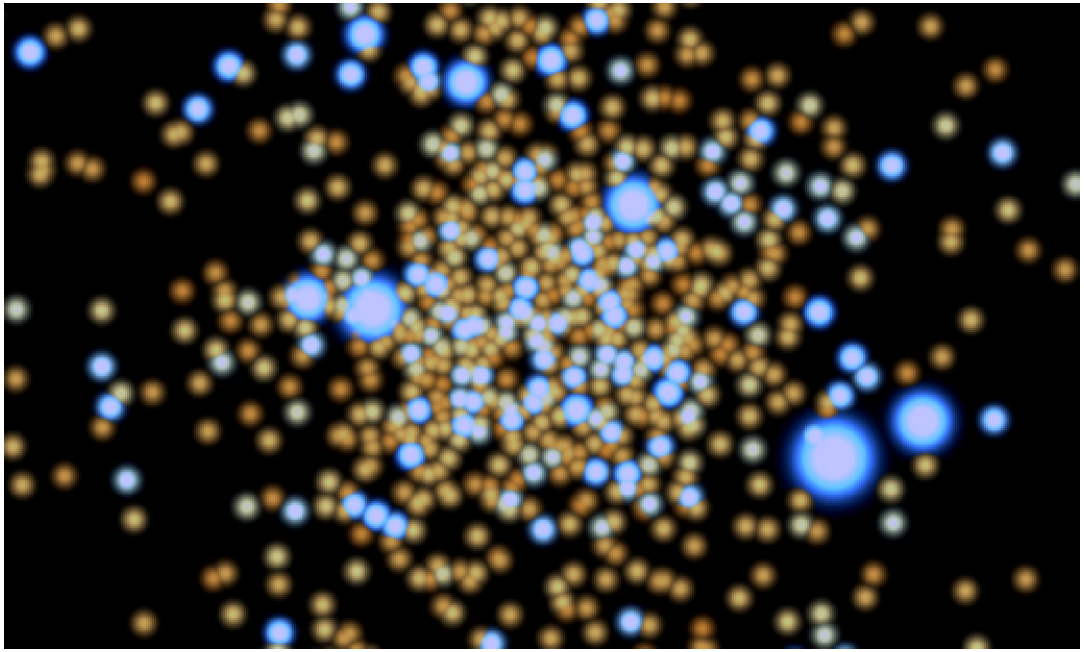
* If code includes prescriptions for stellar evolution and binary evolution (population-synthesis code), each particle has:

  - luminosity,

  - temperature,

  - stellar type,

  WHICH EVOLVE IN TIME!



* Stars in binaries can transfer mass, collide, merge, feel tidal evolution

* Stars can undergo supernova explosions and become compact objects (black holes, neutron stars)

3) the stellar evolution: SEBA
http://www.sns.ias.edu/~starlab/seba/
Portegies Zwart & Verbunt 1996

proto star (0) Non hydrogen burning stars on the Hyashi track

planet (1) Various types, such as gas giants, etc.; also includes moons.

brown dwarf (2) Star with mass below the hydrogen-burning limit.

main sequence (3) Core hydrogen burning star.

Hypergiant (4) Massive (m>25Msun) post main sequence star with enormous mass-loss rate in a stage of evolution prior to becoming a Wolf-Rayet star.

Hertzsprung gap (5) Rapid evolution from the Terminal-age main sequence to the point when the hydrogen-depleted core exceeds the Schonberg-Chandrasekhar limit.

sub giant (6) Hydrogen shell burning star.

horizontal branch (7) Helium core burning star.

supergiant (8) Double shell burning star.

helium star (9-11) Helium core of a stripped giant, the result of mass transfer in a binary. Subdivided into carbon core (9), helium dwarf (10) and helium giant (11).

white dwarf (12-14) Subdivided into carbon dwarf (12) , helium dwarf (13) and oxygen dwarf (13).

Thorne-Zytkow (15) Shell burning hydrogen envelope with neutron star core.

neutron star (16-18) Subdivided into X-ray pulsar (16), radio pulsar (17) and inert neutron (18) star (m<2Msun).

black hole (19) Star with radius smaller than the event horizon. The result of evolution of massive (m>25Msun) star or collapsed neutron star.

disintegrated (20) Result of Carbon detonation to Type Ia supernova.

4) the outputs

http://www.sns.ias.edu/~starlab/internals/

comes naturally from the class structure

PARTICLE: each single node

LOG: log story of the node

DYN story of the node

Hydro story of the node

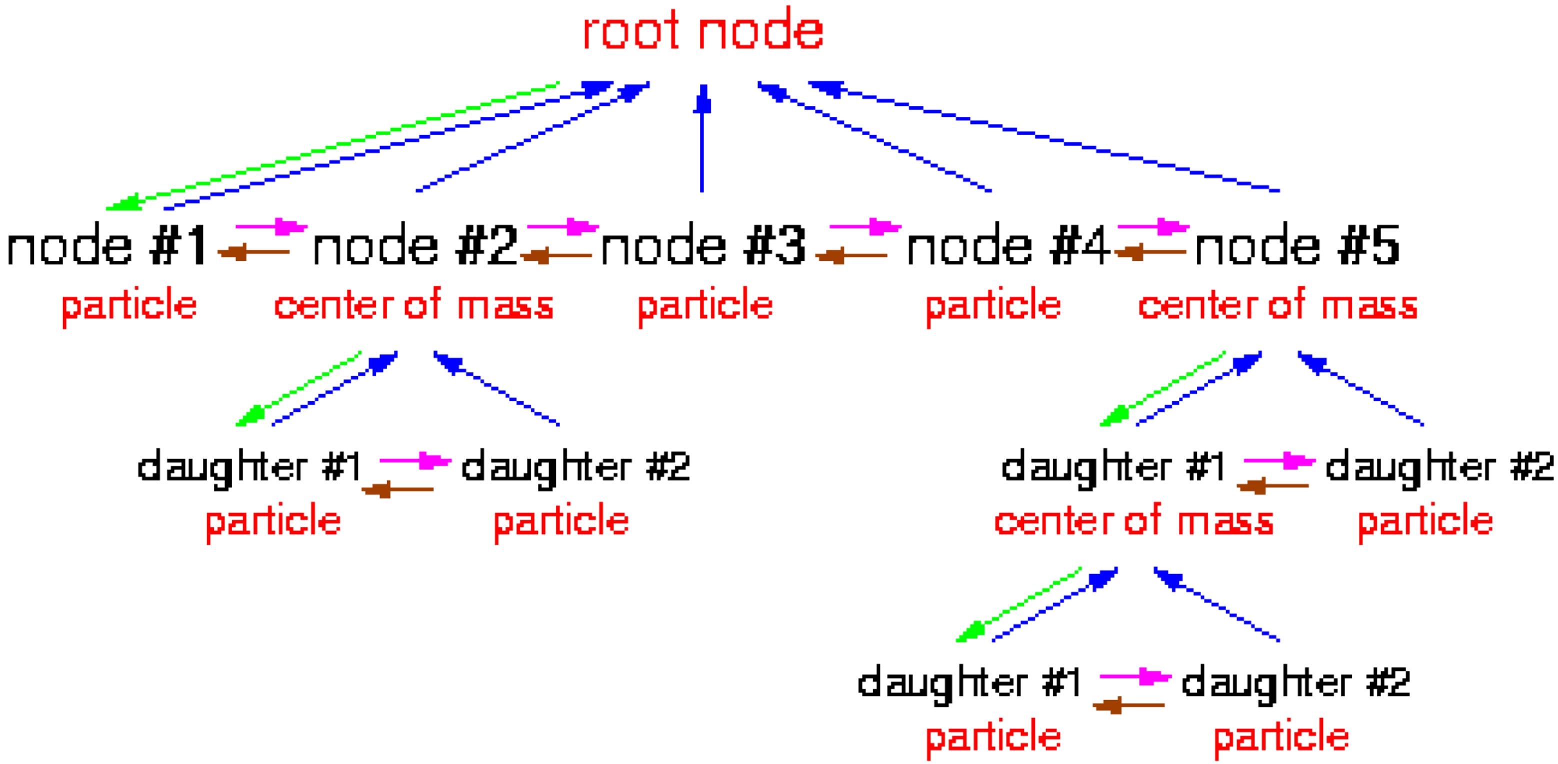
STAR class story

```
(Particle
  i = 4
  N = 1
(Log
Close encounter with black hole #7 at time 10 Myr
)Log
(Dynamics
  m  =  0.5
  r  =  -0.1  0.2   0.5
  v  =  0.3  -0.4  -0.3
)Dynamics
(Hydro
)Hydro
(Star
  Type   =  main_sequence
  T_cur  =  0
  M_rel  =  1
  M_env  =  0.99
  M_core =  0.01
  T_eff  =  6000
  L_eff  =  1
)Star
)Particle
```

4) **the outputs**

http://www.sns.ias.edu/~starlab/internals/

comes naturally from the class structure



4) the outputs

http://www.sns.ias.edu/~starlab/internals/

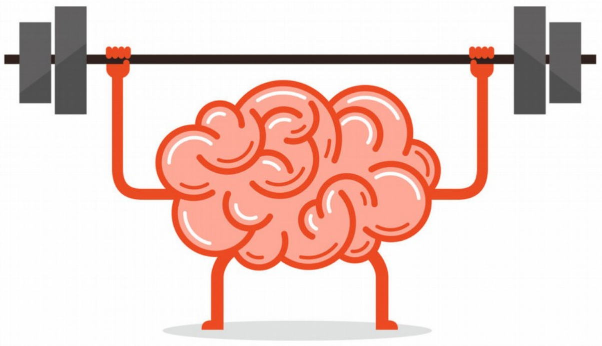
comes naturally from the class structure

PARTICLE: can be single (N = 1), or a binary (N=2) with 2 daughter particles, or a root (name=root), or more complicated dependence

```
(Particle        <--------.
  N = 1                   |
(Log                      |
)Log                      |
(Dynamics                 |
  m  =  1                 |
)Dynamics                 |
)Particle        <--------'
(Particle        <--------.
  N = 1                   |
(Log                      |
)Log                      |
(Dynamics                 |
  m  =  1                 |
)Dynamics                 |
)Particle        <--------'
(Particle        <--------.
  N = 1                   |
(Log                      |
)Log                      |
(Dynamics                 |
  m  =  1                 |
)Dynamics                 |
)Particle        <--------'
```

```
(Particle        <----------------.
      N = 2                       |
    (Log                          |
    )Log                          |
    (Dynamics                     |
      m  =  1                     |
    )Dynamics                     |
    (Particle        <--------.   |
      N = 1                   |   |
    (Log                      |   |
    )Log                      |   |
    (Dynamics                 |   |
      m  =  0.5               |   |
    )Dynamics                 |   |
    )Particle        <--------'   |
    (Particle        <--------.   |
      N = 1                   |   |
    (Log                      |   |
    )Log                      |   |
    (Dynamics                 |   |
      m  =  0.5               |   |
    )Dynamics                 |   |
    )Particle        <--------'   |
    )Particle        <----------------'
```

Exercise # 11:



With my help (next slides)
*** compile starlab**
*** generate initial conditions for a simple King model**
*** run the initial conditions with kira**

5) compilation & installation

```
tar xvfz starlab.tgz
cd starlab/
make clean
./configure --without-f77
make
make install
```

Executables are in usr/bin/

NB: you might have the following error

```
/tmp/ccLw1BTp.o: In function `main':
sqrt.c:(.text+0x3b): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
make[1]: *** [sqrt] Error 1
```

→

```
cd sbin/
mv sqrt.c sqrt.cpp
```

6) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory): makeking.sh

```
./makeking -n 1000 -w 5 -i -u \
     | ./makemass -f 8  -l 0.1 -u 150 \
     | ./makesecondary -f 0.1 -q -l 0.1 \
     | ./add_star -R 1 \
     | ./scale -R 1 -M 1\
     | ./makebinary -f 2 -o 1 -l 1 -u 1000000.0 \
       > king_n1000_frac01_W5.txt
```

* makeking: generates a king profile with
    -n number of centres of mass
    -w dimensionless central potential
    -i number the particles sequentially
    -u leave final N-body system unscaled
  src/node/dyn/init/makeking.C

Useful alternative: makeplummer (**src/node/dyn/init/makeplummer.C**)

6) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory): makeking.sh

```
./makeking -n 1000 -w 5 -i -u \
     | ./makemass -f 8  -l 0.1 -u 150 \
     | ./makesecondary -f 0.1 -q -l 0.1 \
     | ./add_star -R 1 \
     | ./scale -R 1 -M 1\
     | ./makebinary -f 2 -o 1 -l 1 -u 1000000.0 \
       > king_n1000_frac01_W5.txt
```

* makemass: generates mass of primary & single stars from IMF
  -f 1-8: kind of IMF (1 Power-law, 2 Miller & Scalo, 3 Scalo, 4 old Kroupa, 5 DeMarchi, 6 old Kroupa+ 1991, 7 two power law, 8 Kroupa 2001)
  -l minimum star mass (units of Msun)
  -u maximum star mass (units of Msun)

src/node/util/makemass.C

6) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory): makeking.sh

```
./makeking -n 1000 -w 5 -i -u \
      | ./makemass -f 8  -l 0.1 -u 150 \
      | ./makesecondary -f 0.1 -q -l 0.1 \
      | ./add_star -R 1 \
      | ./scale -R 1 -M 1\
      | ./makebinary -f 2 -o 1 -l 1 -u 1000000.0 \
        > king_n1000_frac01_W5.txt
```

* makesecondary: generates mass of secondary from flat distribution
  - -f binary fraction
  - -q if present, secondary mass ratio is chosen uniformly on [lower_limit, upper_limit]
  - -l lower limit secondary mass (if -q in fraction of primary mass)
  - -u upper limit secondary mass (if -q in fraction of primary mass) If not specified =1

src/node/util/makesecondary.C

6) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory): makeking.sh

```
./makeking -n 1000 -w 5 -i -u \
     | ./makemass -f 8  -l 0.1 -u 150 \
     | ./makesecondary -f 0.1 -q -l 0.1 \
     | ./add_star -R 1 \
     | ./scale -R 1 -M 1\
     | ./makebinary -f 2 -o 1 -l 1 -u 1000000.0 \
       > king_n1000_frac01_W5.txt
```

* add_star: generates physical properties of stars (radius)
  - -M mscale - mass scale for stars. If not set uses Mtot → better!
  - -R lscale - dynamical size scaling (in parsecs)
    Error if you do not put anything. May be virial radius of cluster or other scale. Suggestion: put 1 (1 parsec=44370956 sun radii), otherwise you lose control on units.

src/star/sstar/init/add_star.C

6) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory): makeking.sh

```
./makeking -n 1000 -w 5 -i -u \
     | ./makemass -f 8  -l 0.1 -u 150 \
     | ./makesecondary -f 0.1 -q -l 0.1 \
     | ./add_star -R 1 \
     | ./scale -R 1 -M 1\
     | ./makebinary -f 2 -o 1 -l 1 -u 1000000.0 \
       > king_n1000_frac01_W5.txt
```

* add_star: produces in output

```
(Star
  mass_scale     =  0.00137394860469027469
  size_scale     =  2.2550000000000001e-08
  time_scale     =  1.81681396487543956
)Star
```

1/Mtot in Msun
1/Rsun in pc NB! BUG!!!
1/tscale in Myr
in Zsun

6) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory): makeking.sh

```
./makeking -n 1000 -w 5 -i -u \
      | ./makemass -f 8  -l 0.1 -u 150 \
      | ./makesecondary -f 0.1 -q -l 0.1 \
      | ./add_star -R 1 \
      | ./scale -R 1 -M 1\
      | ./makebinary -f 2 -o 1 -l 1 -u 1000000.0 \
        > king_n1000_frac01_W5.txt
```

* kira + add_star: produces in stderr

```
scale factors taken from input snapshot
    [m]: 727.829 M_sun
    [R]: 1 pc
    [T]: 0.550414 Myr
```

Mscale =Mtot/Msun
lscale in pc
tscale in Myr

6) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory): makeking.sh

```
./makeking -n 1000 -w 5 -i -u \
    | ./makemass -f 8  -l 0.1 -u 150 \
    | ./makesecondary -f 0.1 -q -l 0.1 \
    | ./add_star -R 1 \
    | ./scale -R 1 -M 1\
    | ./makebinary -f 2 -o 1 -l 1 -u 1000000.0 \
      > king_n1000_frac01_W5.txt
```

* scale: generates physical scales for final SC

-R specify virial radius

in parsecs, if add_star -R 1 →

(1) ./add_star -R 1 | ./scale -R 5 means rvir=5 in units of 1 pc → rvir=5 pc!!

in units of add_star, if add_star -R != 1 →

(2) ./add_star -R 5 | ./scale -R 1 means rvir=1 in units of 5 pc → rvir=5 pc!!

Almost equivalent, (1) easier, (2) gives more physical meaning to timescale

-M specify star cluster mass

in units of Mtot, if add_star has no -M option →

-M 1 means that mass units in the output file are /Mtot

IMPORTANT THAT SCALE BE AFTER ADD_STAR IF STAR EVOL

src/node/dyn/util/scale.C

6) writing initial conditions

http://www.sns.ias.edu/~starlab/examples/

Use a sh script (easier and keep memory): makeking.sh

```
./makeking -n 1000 -w 5 -i -u \
    | ./makemass -f 8  -l 0.1 -u 150 \
    | ./makesecondary -f 0.1 -q -l 0.1 \
    | ./add_star -R 1 \
    | ./scale -R 1 -M 1\
    | ./makebinary -f 2 -o 1 -l 1 -u 1000000.0 \
      > king_n1000_frac01_W5.txt
```

* makebinary: generates orbital properties of primordial binaries
  - -f function select option
    - 1: angular momentum per unit reduced mass (L^2 = am[1-e^2]), solar units
    - 2: semi-major axis or peri/apo, solar units
    - 3: energy
  - -o specify interpretation of limits - With -f 2 -o 1: semi-major axis,
  - -l lower limit on selected binary parameter (sma in Rsun)
  - -u upper limit on selected binary parameter (sma in Rsun)

src/node/dyn/init/makebinary.C

6) writing initial conditions
http://www.sns.ias.edu/~starlab/examples/

Last note on units

-units of stdoutput:

In (Dynamics ..)Dynamics	units scaled to Mscale, lscale, tscale (note *)
In (Star ..)Star	units scaled to Msun, Rsun=6.95e10 cm, Myr
-units in stderr:	units scaled to Msun, Rsun=6.95e10 cm, Myr

Note * = lscale is that in stderr ([R]: .. pc)

or 2.255e-8/(value in stdout)

where 2.255e-8=Rsun in pc

7) running kira

http://www.sns.ias.edu/~starlab/kira/

Use a sh script (easier and keep memory): run_kira.sh

```
./kira -t 5 -d 1 -D 1 -b 1 -n 10 -e 0.000 -B -s 31107 \
< king_n1000_frac01_W5.txt \
> out_king_n1000_frac01_W5.txt \
2> err_king_n1000_frac01_W5.txt
```

-t number of timesteps

-d log output interval

-D snapshot interval

-b specify frequency of full binary output

-n minimum number of particles (below n terminate simulation,
i.e. if >(N-n) stars escape and are removed, terminate the simulation)

-e softening

-B with binary evolution (and also star, otherwise -S)

-s random seed (default internal clock)

8) visualize outputs

http://www.sns.ias.edu/~starlab/kira/

Simple : use xstarplot

```
./xstarplot < out_king_n1000_frac01_W5.txt
```

8) visualize outputs
http://www.sns.ias.edu/~starlab/kira/

Simple : use xstarplot

```
./xstarplot < out_king_n1000_frac01_W5.txt
```

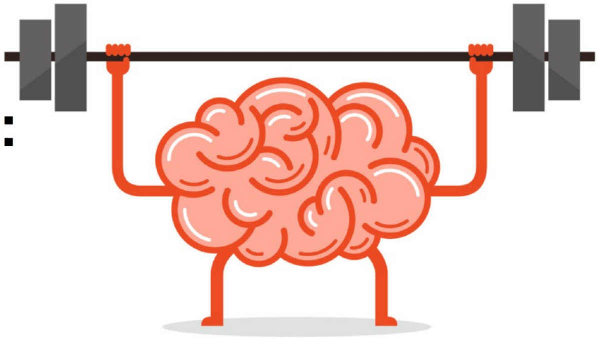
More quantitative: use chop_snapshots.py and read_output.py

* chop_snapshots.py: starlab writes a single output file which contains different output times, but usually we want to analyze stellar properties at a given time.
  chop_snapshots.py divides the output of starlab in single SNAPSHOTS, i.e. outputs with a single time

* read_output.py: reads the outputs of chop_snapshots.py and transforms them into more readable ascii format (columns)

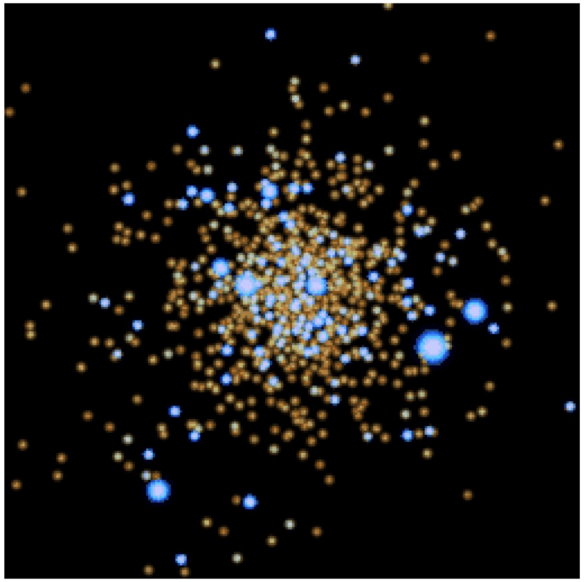
  c.1:Time[Myr] c.2:Mass[Msun] c.3:x[pc] c.4:y[pc] c.5:z[pc]
  c.6:vx[km/s] c.7:vy[km/s] c.8:vz[km/s] c.9:Lum[Lsun] c.10:Temp[K]
  c.11:ID c.12:StarType c.13:single/binary

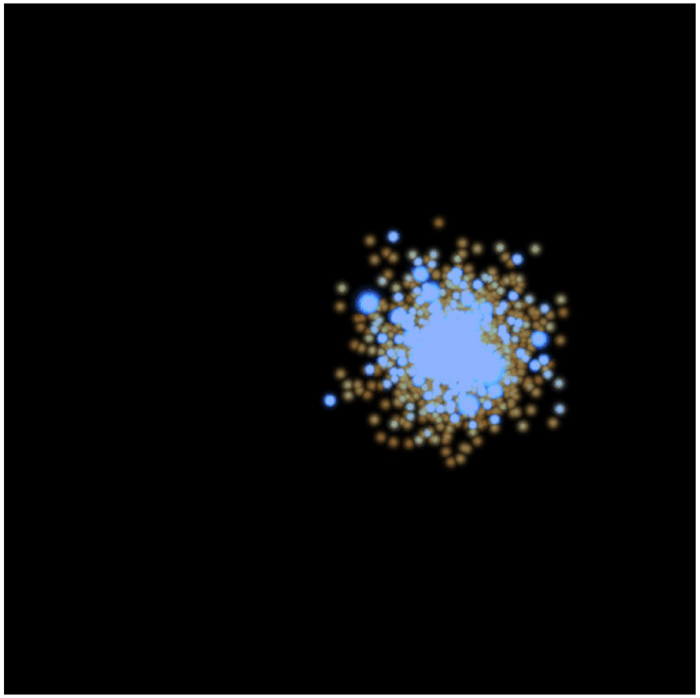
Exercise # 12:



RUN some EXAMPLES of starlab on your laptop:

1- isolated star cluster, 2000 stars, NO BINARIES
createIC_template.sh: initial conditions
run_template.sh: run with kira
./xstarplot < stdout_N2000_W5_Z01.txt
on the fly movie





2- star cluster in Plummer tidal field, 700 stars, no bin.
IC_tidal_field_template.sh: initial conditions
run_tidal_field_template.sh: run with kira
./xstarplot -l 20 < stdout_td_N700_W5_Z01.txt
on the fly movie

3 – use chop_snapshots.py and read_output.py to produce movies of the clusters (or similar plots)

9) CREDITS for STARLAB:

* Thank the authors in the acknowledgments (Portegies Zwart, McMillan, Makino, Hut,...)
* Cite Portegies Zwart+ 2001MNRAS.321..199
  Portegies Zwart & Verbunt 1996A&A...309..179P
* If use GPU, thank the authors of Sapporo: Gaburov, Harfst, Portegies Zwart
  and cite Gaburov+ 2009NewA...14..630G
* If use my metallicity-dep. Version
  cite Mapelli+ 2013MNRAS.429.2298M

10) Online material:

http://www.science.uva.nl/sites/modesta/wiki/index.php/Starlab_tools

and of course

http://www.sns.ias.edu/~starlab/index.html

Download my version and templates