*KGS is an algorithm for computing and displaying dynamical simulations of orbital systems comprised of* ***n*** *bodies.*

* **output**:
* **data**:

matrix ***OrbS*** of orbital systems:

where

***S*** *is a matrix**of*dimensions ***n* x *∆t/δt****,* of the dynamical states of **n** bodies during the interval of time ***∆t*** and computed with theincrement of time ***δt*** in describing the paths of the orbital system with respect to a frame of reference ***REF***.

and metadata matrix

(REF = frame of reference)

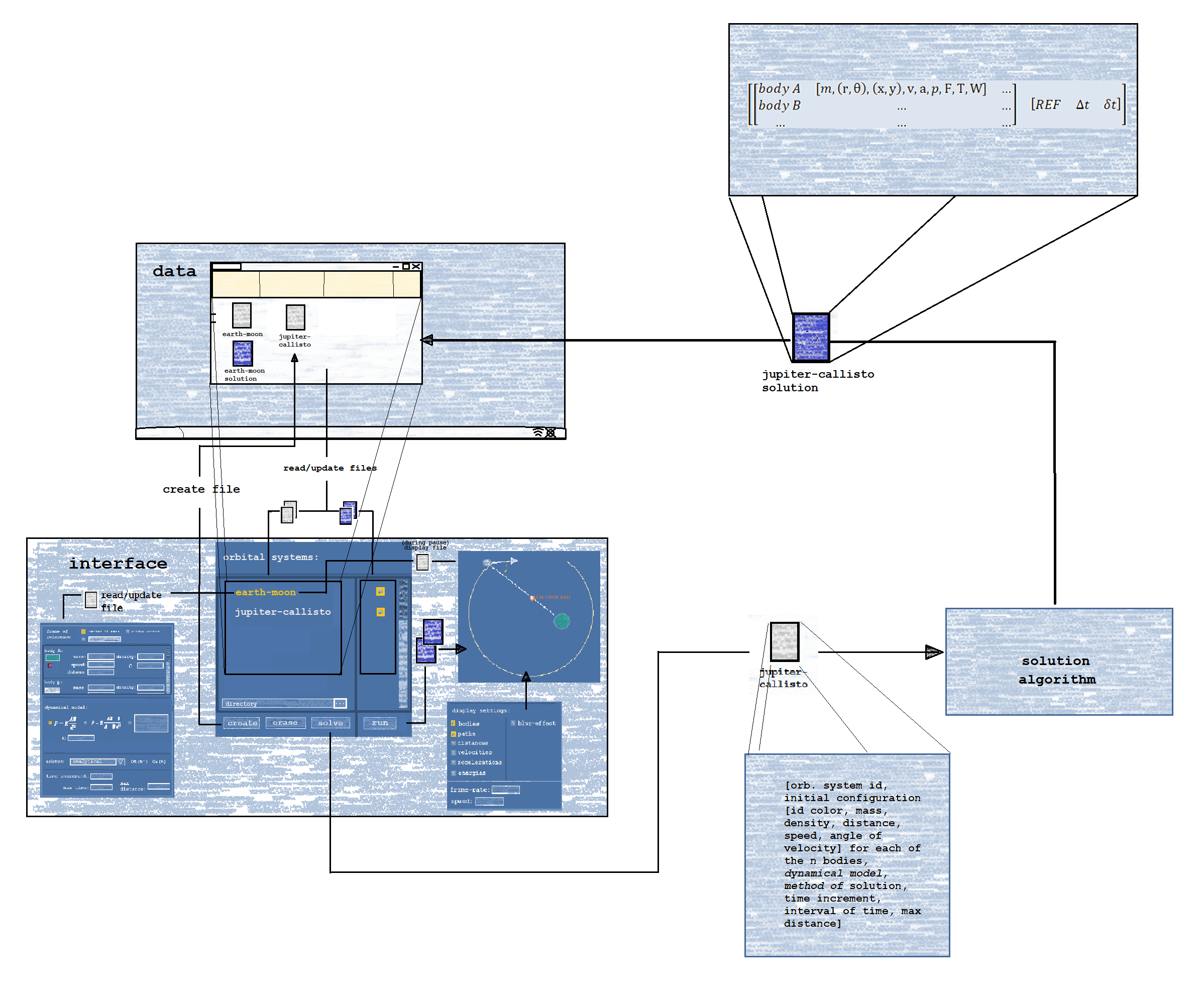
* **display**:
* Interactive interface for entering the inputs and animated output display of all dynamical data withframe-rate ***f*** and speed ***s***, with option to pause and rewind to any instant.
* Time and memory complexity of each method of solution available.
* **input**:
* **output display options:** [framerate, display speed, show paths, show distances, show velocities, show momentum, show forces, show accelerations, show energies, blur-effect]
* **orbital system parameters**:

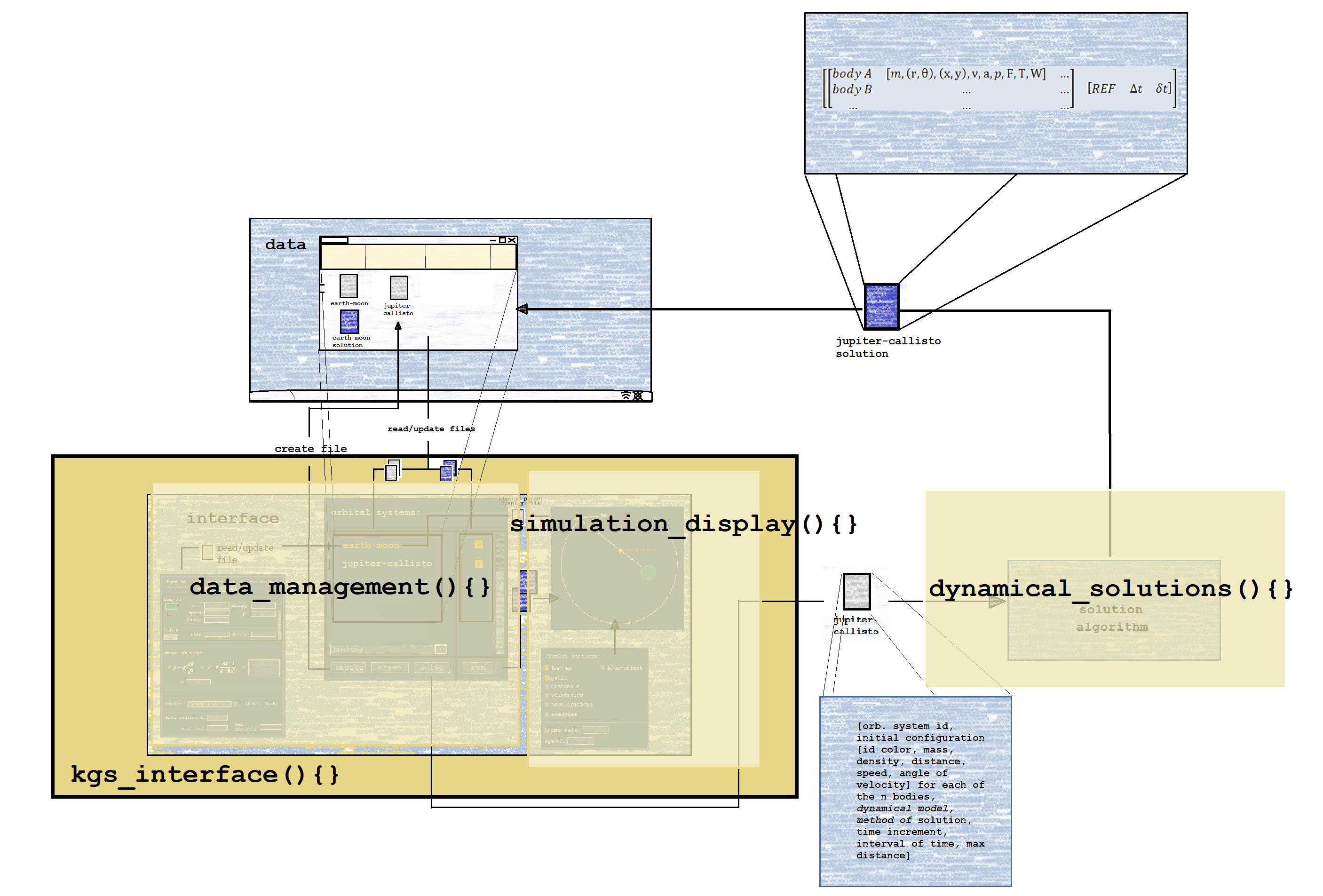
[orb. system id, initial configuration [id color, mass, density, distance, speed, angle of velocity] for each of the **n** bodies*, dynamical model, method of* solution,time increment,interval of time*,* max distance]

* **GUI commands**:

[select file, select parameter, edit parameter, edit preferences, create, erase, solve, run/pause]

* **diagrams:**

****



* theory:
* algorithm:
* algorithm analysis:
* complexity analysis:
* **code Infrastructure:**

**dynamical\_solutions(**[orb. system id, initial configuration [id color, mass, density, distance, speed, angle of velocity] for each of the **n** bodies*, dynamical model, method of* solution,time increment,interval of time*,* max distance]**) {**

**return**

**}**

**data\_management(**directory**,** orbital\_system\_file, solution\_file, dynamical\_parameter**) {**

**get\_files\_list(**directory**){return** list\_of\_files**}**

**fill\_parameters(**orbital\_system\_file**){return** list\_of\_parameter\_values**}**

**update\_file(**orbital\_system\_file, dynamical\_parameter**){}**

**erase\_file(**orbital\_system\_file, solution\_file**){}**

**}**

**simulation\_display(**orbital\_system\_file, solution\_file**) {**

**display\_orbital\_system(**orbital\_system\_file**){}**

**display\_solution(**solution\_file**){}**

**}**

**kgs\_GUI(){**

**while (not exit){**

**if** user\_input selects directory, call **data\_manager.get\_files\_list(**directory**)**

**if** user\_input selects file, call

**if** user\_input modifies parameter, call **update\_file(**orbital\_system\_file, dynamical\_parameter**)**

**if** user\_input selects “erase”, call **erase\_file(**orbital\_system\_file, dynamical\_parameter**)**

**if** user\_input selects “solve”, call **dynamical\_solutions**(orbital\_system\_file)

**if** user\_input selects “run”, call **simulation\_display**(solution\_file)

**}**

**}**

* data structure:
* system requirements:
* language: