Introduction to K Gravity

by A G [Greg] Kaiser

info@kravity.com
howwork.kaiser@gmail.com

November, 2013

Introduction

Gravitational Constant $G = 6.67398 \times 10^{-11}\ m^3\ kg^{-1}\ s^{-2}$

A mass in orbit about another achieves it's maximum velocity at perihelion, the point where the bodies are closest to one another and the pull of gravity the strongest. The “gravitational slingshot” can, when the circumstances are right, cause masses to attain significant fractions of c, the speed of light. KGravity calculates the acceleration due to gravity for up to 1440 masses (an arbitrary limit chosen with respect to the speed of today's PC processors) and projects the three dimensional mathematics onto the computer screen. Kgravity-demo.zip contains a preview and quite functional version of kgravity.exe. It is downloadable: http://www.kgravity.com/gravity/kgravity-demo.zip. Try it out and let me know what you think of it.

Upgrading development systems, equipment and software will hasten improvements to KGravity by increasing productivity and assuring dependability of the production platform. The over two year old laptop that I use has dead keys and usb ports. Other hardware is old or non-existent, like the CAD software I don't possess. If you would like to help with funding to support those upgrades, as well as iOS and/or Android app development, a business manager and perhaps legal assistance setting up a corporate edifice, contact me. info@kgravity.com. Feel free to ask any questions, even if you don't care to help fund the project.

THEORETICAL WONDERS

For decades I've pondered the possibility that relativistic effects [mass and especially time dilation and distance contraction/space warping] are mathematical artifacts that result from the imperfect description of the universe. That is: they are flaws in the theoretical model, singularities/anomalies only in the mathematical sense, not in reality.

In Wikipedia there's a quote from Nikola Tesla:
"“I hold that space cannot be curved, for the simple reason that it can have no properties.” The New York Herald Tribune. 11 September 1932'

Not too long ago I was watching several hundred masses whirl and whip about a large central mass. I enjoy watching my artificial reproduction of gravitational systems. They fascinate me and I'm proud of the program I've written to produce them.

That day I began to think about what's happening relativistically with masses approaching perihelion, both mathematically in a computer simulation and, if the theory of relativity holds, in a real gravitational system.

I've seen velocities greater than 0.5c [c is the speed of light] in lesser masses flung off by stars. With very large masses and collision radii uncorrected for the increased volume I've seen velocities greater than c. [c =~ 3.0e5 km/s] I'll explore what happens with realistic collision radii though there are such things as neutron stars that have very small diameters and greater masses than ordinary suns. But for now I'm thinking about the application of relativity to the approach to perihelion. I hope to have more time to play later on.

For now, imagine a mass that achieves a significant fraction of c before perihelion. Will its mass increase as relativity tells us? *{Is this true: In the Sun's frame of reference it's impossible to tell whether the mass is approaching the Sun or the Sun is approaching the mass. Therefore do both masses increase?}* Can a mass achieve c without colliding and merging with the star? What about two neutron stars approaching one another at sub light velocities? If c is attained and a mass increases to infinity, will the universe collapse into a single mass? Is it certain that Vg, the speed of gravity is, c? Has experiment shown that Vg = c or is that just to be inferred from accepted theory?

A little Google search turned up considerable discussion and controversy on the subject, starting with LaPlace in 1803. He was convinced that Vg is much greater, at least a million times, than c. Since gravity propagation waves have never been detected, only indirect measurement can be applied.

The Speed of Gravity: Einstein Was Right!
(Released July 2003)
by Salvatore Vittorio
. . . They found that gravity does move at the same speed as light.

That conclusion, based on observations of the deflection of light from a quasar by Jupiter on September 8, 2002, has been challenged by Samuel of Berkeley Lab, who says, “There’s a reasonable chance that such measurements might one day be used to define the speed of gravity, but they just aren’t doable with our current technology.”

KGravity will never be used to answer all of those questions. But it will be fitted with better “gravity engines” than it currently employs. It will have a relativity option before I'm sufficiently satisfied with it to release it to the public domain and the the open source community as a wxWidgets project. It's already sufficient to study the approach of neutron stars or other masses to perihelion, non relativistically. That's not as good as I will make it but much can be learned by the use of this program without the anticipated improvements.

For instance: I once constructed a system of three neutron stars. I don't remember how massive I made them but a little trial and error applied to the use of the application should be able to duplicate the result I achieved. I know they were several times greater than the mass of the average star, which is about 10^{30} kg. These masses were positioned a few parsecs from one another with small or no initial velocities. After a few million years, they were accelerating towards each other and soon two whipped about a common center and sped off at super light velocity.

What will happen in that scenario when relativity is applied to the mass? As two approach one another and achieve a significant fraction of c, their masses will begin to increase, which will increase the rate of acceleration and then their masses will increase more which will . . .

I'll ignore time dilation and distance contraction since unlike mass, time and space are not things. Time is derived from the motions of matter and energy. Space is emptiness. Both are only concepts that help us to put things in order.

Putting theoretical heresy aside, the enjoyment I've gotten from coding and playing with KGravity-Win32 and expect to get in the future is tangible and I know it's measure. That's what matters to me.

The earliest precursor of the KGravity windows program that performed N-body gravity calculations was named Greg's Galaxy. I started one Spring several years ago and by that August the basic Gravity Poll calculations were accurate enough to keep the Earth, Moon and a couple of planets in orbits about the Sun for centuries of KGravity time. One day passes for each computational cycle of a “planetary” system. For 336 masses that occurs about 10 times per second on my 1.8G dual core 64 bit AMD processor running Windows 7 with no other applications competing for CPU time. KGravity has been one of several programs employing the Win32 API that occupied my real time in the past 12 years.

The original inspiration was a desire to watch “sling shot acceleration” of smaller bodies as they neared perihelion. I'm still fascinated when I see them flung off at rates approaching c. KGravity reports the speed of the masses in m/s, km/s or as a fraction of c, starting at 0.01c which is 3000 km/s. I don't believe I'm the only one who can derive pleasure from watching Newton's universal gravitational formula rendered in graphics for up to 1440 masses. (at the time of this writing in October, 2013) The perspective projection, which projects the z-axis into the screen, was built into the system in the early years of the new millennium.

KGravity has grown considerably in quality and functionality since its modest beginnings. Now it's time to move from the realm of my personal amusement and demonstrate its fun and usefulness to all. I expect to take it to a further level by the Fall of 2014. The Kickstarter funding option, if successful, will expedite and facilitate the realization of the expanded KGravity project.

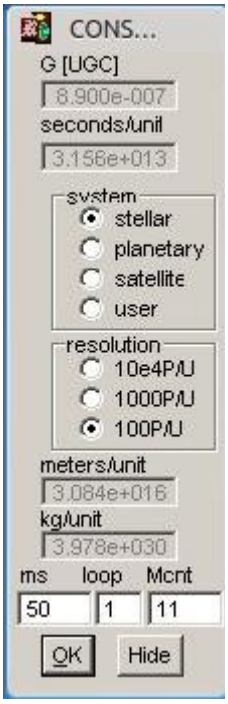
The first moves toward the goal have already begun. In October, 2013 I implemented user loadable background bitmap files. And I made many more improvements in 2013, with still more listed in a “to do” section in my 150 pages of notes and documentation to date. Most exciting, I will perfect tilt and rotate, allow system construction by scenario text files and save states to a binary file. The full list is also included at the end of this document. When these ideas and any new inspirations are dealt with (I'm open to your suggestions) the Win32 application will have come as far as I plan to take it in it's present form. Next comes a wxWidgets version for Windows which will easily be ported to GNU/Linux. After the GNU application proper is gifted to the public domain as free and open source software, I expect to write a companion book that will be developed from the notes and documentation.

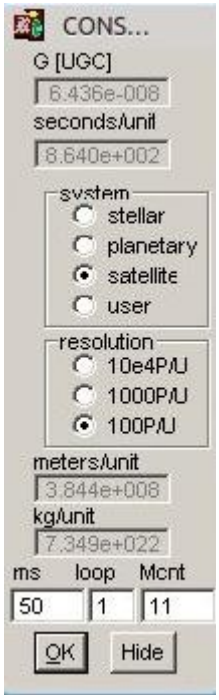
The Kickstarter presentation video and primitive Kgravity-Win32 demo exe are down-loadable at kgravity.com. http://www.kgravity.com/gravity/kgravity-demo.zip When KGravity is sufficiently funded I will be able to get better equipment and possibly even personnel to offload some of the tasks.

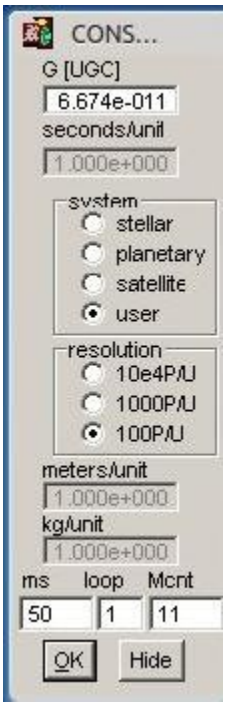
THE TECHNOLOGY

KGravity-Win32 uses several conversions of G to units suitable for the types of system simulated. Presently there are four systems represented by a radio button array in the constants dialog. These systems are: Stellar, Planetary, Satellite, and User. Each has its own su - system units.

The default value of G is calculated (see the constants G calculations below) for each of the Stellar, Planetary and Satellite systems and are plugged into the following constants dialogs:







Stellar: Planetary: Satellite: User:

As you can see, G and su are fixed for each of the first three categories of systems. The user system fixes the su as MKS: meter, kilogram, second. G is standard MKS but editable.

G=6.67398e-11; The other editable fields are the minimum milliseconds each update and draw cycle will consume; the number of update loops between screen update/draw cycles; the number of masses in the system. The latter is presently limited to 1440.

The number of steps in the "customized! Euler Method" computation of

$$dv/dt = G * M / r^2$$

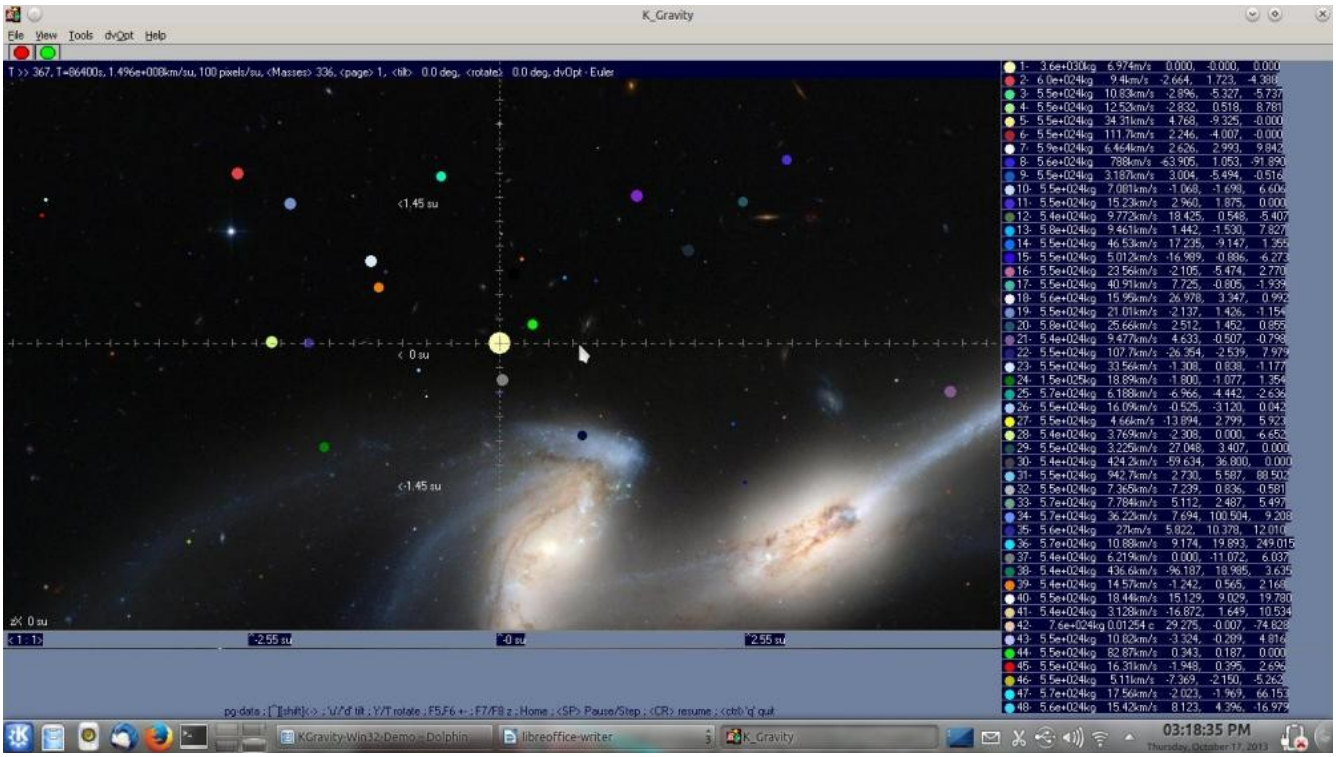
is determined by the time su. For example, in the Planetary System the unit of time is one day, which is 86,400s. Each of the up to 1440* masses updates itself by calling MassList::GravityPoll, in which every real mass in the system looks at every other mass and calculates/accumulates its own dv/dt from the deltas contributed by the other. Their actual position data member is not updated until each mass's GravityPoll function has run. Then the results of all masses are written to a memory display context that's copied onto the screen.

! Customized: Using the Euler Method (a numerical method of solving differential equations) a satellite's orbit is

broken into the number of iterations of the gravity calculation by setting the loop counter accordingly. Since kgravity does the calculation for many masses simultaneously it would not make sense to base the number of calculations on one particular orbital circumference. So the conversion of G to the System Units in use determines the number of iterations of the gravity calculation. One cycle occurs for each [system] unit of time that passes. The screen update marks the completion of cycles. The message at the upper left of the viewport includes the period [T] count.

* 1440 are slower than patience allows on a 64bit AMD C50, running either Windows 7 or Kubuntu 12.04-64 and Wine. (Some features fail in Wine.) I seldom make more than 336 mass systems, which result in 7-48 ID pages of mass, velocity and position data on the right side of the viewport. Later, a menu selectable limit may be implemented. Not all features work well in Windows emulators. KGravity has worked well in Windows XP and Windows 7. Newer versions of Windows will be tested when funds are available for new equipment.

The first three system categories correspond to the first three of the four mass classes derived from class Mass: Star, Planet, Satellite and Debris masses can be constructed and added to a system. A fifth Mass class, Message, provides information at the top of the viewport but has 0 mass and is filtered out of the GravityPoll by its typeid. I consider it to be a pseudo mass. Though it is a valid Mass derived class, aside from the Draw function, it uses little of the data, properties and methods of the ordinary masses.

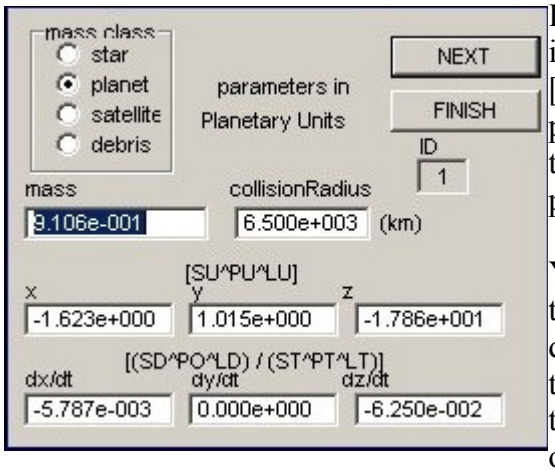


Here's a screenshot of the opening system of 336 masses: one central star, planets, satellite sized bodies and debris. The data presentation on the right consists of 7 pages of 48 masses each that are accessed with the page up and down keys. The status line at the bottom defaults to a line of the key commands available The info line under the tool bar on the left is updated each time the 336 mass system iterates: about 5 times per second when run in Kubuntu/Wine [Windows Emulator] on an AMD Dual Core 64 bit platform with 2G of ram. The same build of KGravity-Win32 on the same machine running

Windows 7 is about twice as fast. The Windows Emulator necessarily slows things down. I expect the [eventual] Linux WxWidgets 64 bit version to be near the target of 50ms [20 Hz] cycles. Currently I limit masses to 1440.

The planetary category is the default and is used in the opener. The system unit of distance is the average Earth orbital radius: 1.49597870700e8 km. The unit of time is one day: 86,400s. The unit of mass is 1 Earth mass: 5.972196e24 kg. G is calculated using those units.

To build your own system hit Ctrl+n. Enter the options you want including the number of masses the system will have in the constants dialog. When satisfied press OK. You may press HIDE to cancel. OK will bring out the mass builder dialog. If one selects 0 masses, I have KGravity create the sun, 9 planets and a moon orbiting the third planet. All the masses in this system and their positions, velocities and orbital tilts correspond to our Solar System. They are entered as average distances and velocities and their initial positions are arbitrary and selected for ease of calculation. They correspond to no real alignment of Sun and Planets that I've referenced. [a simple test of the accuracy of KGravity calculations: calculate the average tangential velocity of our planet in km/s. Build the "0" system and check the data for the third planet. Watch it go and estimate how much it varies. Can you spot perihelion and aphelion by those observations?]



In the mass builder dialog, you may press Finish immediately. The system will be built with a stationary star [ID #1] at the center and the remainder of masses will be planets, satellites and debris with random masses suitable to their individual classes. They'll be assigned random positions and velocities.

You may also select the class and enter your own values for the mass, position or velocity for as many as you care to do. Press next to do another. You may press finish at any time. The remainder will be built with random values and the dialog will not be recalled. If you complete the number of masses specified and press next the system will be built and the mass dialog will not be recalled.

The screen will not be updated with the new system and calculations will not begin until you click the green "go" button or press enter. You can also press the space bar to start a single cycle. If the system is active, the space bar will freeze it and subsequently will single step it. Enter or go or file menu resume to continue.

Relevant Formulas, Constants:and Conversion Factors

$$G = 6.67384(80) \times 10^{-11}\ \mathrm{m^3\ kg^{-1}\ s^{-2}} = 6.67384(80) \times 10^{-11}\ \mathrm{N\,(m/kg)^2}$$

//

// gravitational constant G = 6.67398 × 10^{-11} m^3 kg^{-1} s^{-2}
// some useful constants and equations:
// g = 9.81 m / s^2 at surface of Earth
// Earth_mass = 5.97e24 kg
// N == Newton
// N = 1 kg*m/s^2
// 86,400s / 24hrs
// Newton's Law of Universal Gravitation:
//
// F = G*m_1*m_2 / r^2 ; F = ma
// [G = 6.67398 × 10^{-11} m^3 kg^{-1} s^{-2}]
// circular orbital radius specified by orbital period, T
// ||v|| = 2*pi*r/T
// ||a|| = 2*pi*||v||/T
// ||a|| = G*M/r^2
// r = $(G*M*T^2/(2*pi)^2)^{(1/3)}$
// T = 86400 seconds [24 hrs] for Geostationary orbit
// r = 4.225 * 10^4 km
// ||v|| = 2*pi*r/T
// = 1.1061 * 10^4 km /hr = 6.8690 * 10^3 mi/hr
//
// Escape Velocity
//
// $Ve = \sqrt{(\frac{2*GM}{r})}$
//
// ----------
/**********//

$$G \approx 6.674 \times 10^{-11}\ \mathrm{N\,(m/kg)^2}.$$

In cgs, G can be written as:

$$G \approx 6.674 \times 10^{-8}\ \mathrm{cm^3 g^{-1} s^{-2}}.$$

G can also be given as:

$$G \approx 0.8650\ \mathrm{cm^3 g^{-1} hr^{-2}}.$$

The default value of G is calculated for each of the Stellar, Planetary and Satellite systems using the following:

SD = 1 parsec = 3.258 l.y. = 3.085678e16 m: intersteller unit distance [su]
SM = Sol mass x 2 = 3.94e30 kg: star mass
ST = 10,000 years: a small astro time

PD=1.49597870700e11 m/PD: Average Earth Orbital Radius,
PM =5.972196e24kg: // Earth Mass
PT=86,400s: Earth day

LD = 3.7800e8 m: average distance to Luna, meters
LM = 7.34579e22 kg: mass of the Moon, kg
LT = 864s = 0.01 days

Under the help menu “help with G”

100 pixels = 1 su // current screen
LtYr=9.471080e15 meters; year=3.15581e7 s
current system units[su] == Planetary
in which G=8.88712e-10

GS specified in Star cluster units
GS = 8.91338e-7, ST=10e4 years
SD = 1 parsec = 3.085678e16 m = 3.258 LtYr
SM = 2 X Sol's mass = 3.94E30 kg

GP in Planetary units
GP = 8.88712e-10, PT=1 day=8.64e4 s
PD = 1.49597870700e11 m
PM = 1 Earth mass = 5.972196e24kg

GL in Satellite (Lunar) units
GL = 6.77589e-8, LT= 0.01 day=8.64e2 s
LD = 3.7800e8 m meters = average lunar distance
LM = 1 Lunar mass = 7.34579e22 kg

key commands:
arrow key viewport tranlations, +-LroffSet [horizontal], +-UdoffSet [vertical]
ctrl+arrow LroffSet,UDoffSet +-1
shift+arrow LRoffSet,UDoffSet X 10
CR starts timer if off
ctrl+'n' New System
F1 and Shift F1 trigger message boxes [stops timer]
F5 and F6 zoom in and zoom out
F7 and F8 translation on z axis
Space Bar timer off and single step

Page up and Page down data display pages.
Home key zeroes viewport offsets and zooms

TO DO LIST

System reset reloads current system from original data
No new data members required. It should be a simple addition to File or View Menu.

On second thought, it will be better to create a new data struct to store originals. A struct with only primary data types, ints, floats, char, and enum, can be easily saved to a binary file.

Reset and file save and retrieve will all be done together. That way reset will include absorbed by collision masses that would be lost if not saved in array or file. File better.

One:
clean up koords.h; done 4/6/13

mass.h

Two:
//class Mass protected data MClass thisClass; //typedef enum MCLASS {str, pln, lun, deb} Mclass;
struct and typedef MassData with long id and enum thisClass; initial position[u,v,w,true], velocity[u,v,w,true], mass, collision radius.

MassData massData[numMasses];

in mass.h class definitions and mass.cpp constructors, replace orgMass, KPoint original etc. with new data structure.

Three:
masslist.cpp
save masslist at startup and on demand.
Reset or retrieve file: destroy MassList. NewGravity masses built from massData[(id-1)%numMasses]; loaded from file. Or: ID 0 first and contains numMasses and filesize and other info: massData[id]

Canned Scenarios
more like Prime-System and make them menu selectable.

10/23/13
Build new system is now a popup under file. It contains “New.” Prime-System and Neutron-Stars are stubbed out with “future home of” messages.

Configuration management and version control:
A version will be released.
The release will be built in a designated project directory [dpj].
It will have its own k_winobj in the project.
The dpj will have its own img, Icons, kinclude and klib subdirectories.

Numeric Methods for Differential Equations – Improving Gravity Calculations
Heun's method, improved Euler's or other numeric differential equation implementation in:

void MassList::GravityPoll(Mass *caller)
Alternate GravityPoll functions are loaded into an array whose index is menu selectable.

GravityPoll[] {Euler's, Heun's, relativistic, . . .}
Euler's and Heun's are implemented. Runge-Kutta and relativistic are stubbed out.

Relativistic Gravity
see long double and dilation factor study under Z- miscellany - low priority

Re-enable zoom and enhance perspective rendering.
These are adequate but too far from ideal for satisfaction. Work on it!

Tilt and rotate. [started 2/27/13]
compensate rotate and tilt for SortByZ is messy. Rotate and tilt axes with distance labels compensated and for z translation while I'm at it is all very messy. Eventually I'll deal with it.
Address jerkiness on the periphery

Individual Mass Data Display
Click on ID # in data display to bring up a more complete description of the Mass, including escape velocity required at the moment. The data draw algorithm can access masses by ID, obviously. Do I need another data structure to do this?

The same array of masses should be designed so it may also serve for position 0 of a Huen's Method GravityPoll(. The loop parameter in the set constants dialog could be used to choose differential equation approximation to be used. Either GravityPoll [Euler] or GravityPollHuen

A global pointer to the array might be more versatile than a MassList public Mass[] data member. Maybe not! If ML data it would allow for multiple MassLists if I discover a need or use for such. It would probably also be easier to stream a ML system to or from a file.

Each mass has MassList* clerk in its data stash but there is no global MassList*. Don't presently see a need for one.

Map viewport on 8x8 pixel grid. Right – click in zone results in most forward z found by MassList traversal to be loaded into masser dialog for viewing data.
Requires additional dialog functionality and mousemove intricacy.

User Selectable Units for Translation Display
Perhaps make units displayed on gravity canvas user selectable at a later date.

Accelerate Masses
Apply force to a mass to accelerate it. See Mass::SetMomentum.

Save state. Input data from text file.

Standard Astronomical Coordinates
and conversions among them and to KCoords for input and state.

MORE PLAYING WITH KGRAVITY - NEUTRON STARS

“A typical neutron star has a mass between about 1.4 and 3.2 solar masses (see Chandrasekhar Limit), with a corresponding radius of about 12 km.” http://en.wikipedia.org/wiki/Neutron_star

Recent play: “Planetary” system of 96 masses; 1 star 10**35 kg (0,0,0) v=0; 3 more stars similar masses and random positions and velocities; remainder of bodies random with normal masses. Immediately after start several objects achieved super-light velocities. No real black holes are known to be so massive as the four “stars” created for this scenario. The largest known, to my understanding, is about 15 solar masses. When I have more time to play with KGravity, I'll see how heavy I must make stars to achieve faster than light velocities in the program's virtual cosmos.