Physics 750: Assignment 2

due Tuesday, October 10, 2017

The goal of this assignment is to simulate the dynamics of a system of gravitating particles. You should begin by downloading the `assignment2.tgz` archive from the class website. In the `assignment2` directory there are two header files, `vec2.hpp` and `particle.hpp`, that contain class definitions for a two-dimensional vector and a point particle. You are free to use these classes and to modify them if necessary.

As an example of how to proceed, I have included a program `example.cpp`, which simulates three small "planets" orbiting a larger "star" in an artificial (and unstable) solar system. All four bodies interact via the graviational force (with $G = 1$) and are confined by their initial conditions to the x-y plane. From the `assignment2` directory, invoking

```
$ make
g++ -o example_openGL example.cpp -O2 -ansi -pedantic -Wall `./glflags.bash` -DUSE_OPENGL
g++ -o example example.cpp -O2 -ansi -pedantic -Wall
```

will build both graphical (`example_openGL`) and non-graphical (`example`) versions of the program. A single source file `example.cpp` contains OpenGL code that is selectively activated with the `USE_OPENGL` macro.

Running `./example_openGL` opens a new window and animates the planetary motion. The window displays everything inside the coordinate square with vertices $(-1, -1)$ and $(1, 1)$. You can adjust the speed of the animation by recompiling with different values of `const int delay`. The nongraphical version writes the four trajectories to files in a 7-column format. (`operator<<` is overloaded in the `particle` class definition, so `cout << gas[n]` is a legal way to output information about the nth particle.) Try the following:

```
$ ./example
$ ls tr*
trajectory-A.dat trajectory-C.dat
trajectory-B.dat trajectory-D.dat
$ gnuplot -persist view1.gp
```

1. (3 points) Write programs `figure8` and `figure8_openGL` that execute the *choreographic* three-body motion described in Phys. Rev. Lett. **98**, 201102 (2007). Use `example.cpp` as a template, and add the appropriate entries to the `makefile`.

   ```
   $ cp example.cpp figure8.cpp
   $ emacs makefile &
   ```

   Instead of the example planetary system, set up three particles of unit mass with the following initial positions and velocities (again, in units where $G = 1$).

   $$\begin{aligned} \boldsymbol{r}_1 &= (0.48500218, -0.121543765) & \boldsymbol{v}_1 &= -\frac{1}{2}\boldsymbol{v}_3 \\ \boldsymbol{r}_2 &= -\boldsymbol{r}_1 & \boldsymbol{v}_2 &= -\frac{1}{2}\boldsymbol{v}_3 \\ \boldsymbol{r}_3 &= (0, 0) & \boldsymbol{v}_3 &= (-1.31862315, -1.222914959) \end{aligned}$$

   (Hint: The collection of particles is stored in a global `vector` variable named `gas`. Particles are added in the function `initialize_gas` using the `push_back` method.) Have the program dump the trajectory of one of the particles to a file. Plot the x coordinate of the particle versus t and observe that the motion is periodic. Then plot x versus $\sin(2\pi t/T)$ and extract the period T by data collapse.

2. (2 points) The implementation inherited from `example.cpp` is not as efficient as it could be. In particular, the functions `update_gas` and `recompute_acceleration` are set up in such a way that the overall computational cost scales as N^2. Rework the code so that only $N(N-1)/2$ pairwise forces are computed. [In this context, N is `gas.size()`]. You should be able to do this without introducing any expensive new storage.

3. (3 points) Now a further improvement. As it stands, each force calculation requires calls to the expensive `cmath` functions `atan2`, `cos`, and `sin`. Rework the expressions so that the same calculation is performed with at most one call to `sqrt`.

4. (5 points) The assignment directory contains a file `planets.dat` that provides sufficient information to determine the initial conditions of the planetary system. For each planet, the semi-major axis, period, eccentricity, relative rotation, time offset, and mass are listed. The values are given in units of the semi-major axis of Neptune, the solar mass, and the terrestrial day. Each planetary position $\boldsymbol{r} = (x, y) = (x' \cos\omega - y' \sin\omega, x' \sin\omega + y' \cos\omega)$ is rotated away from the coordinates $x' = a(\cos\mathcal{E} - e)$ and $y' = a\sqrt{1-e^2}\sin\mathcal{E}$, which are themselves written in terms of the semi-major axis a and eccentric anomaly $\mathcal{E}$. The radius from the sun is $r = a(1 - e\cos\mathcal{E})$. The time dependence enters through the progession of the mean anomaly $\mathcal{M} = \mathcal{E} - e\sin\mathcal{E} = 2\pi(t + t_{\text{offset}})/T$. Here, t_{offset} is the offset time and T the period. Convince yourself that this is the correct expression for the velocity:

$$\boldsymbol{v} = (v_x, v_y) = (v'_x \cos\omega - v'_y \sin\omega, v'_x \sin\omega + v'_y \cos\omega),$$
$$\boldsymbol{v}' = (v'_x, v'_y) = \frac{2\pi a/T}{1 - e\cos\mathcal{E}}(-\sin\mathcal{E}, \sqrt{1-e^2}\cos\mathcal{E}).$$

Write a program that converts this information to a tabular form consistent with the friend function `operator>>` defined in `particle.hpp`, and write the results to a new file `planets-snapshot.dat`. Running your program at $t = 0$ should give the following.

```
$ head -n2 planets-snapshot.dat
Earth      3.04e-06   0.0  -0.00553554  0.0322417  -0.000573523  -9.87179e-05
Mars       3.227e-07  0.0   0.0456764   0.0102734  -8.32558e-05   0.000493383
$ tail -n2 planets-snapshot.dat
Neptune    5.181e-05  0.0  -0.520517   -0.863652    8.87598e-05  -5.3273e-05
Sun        1.0        0.0   0.0         0.0         0.0           0.0
```

5. (4 points) Start working on a new file `solar.cpp` derived from `figure8.cpp`. Add entries to the `makefile` so that nongraphical and graphical programs `solar` and `solar_openGL` can be compiled.

```
$ cp figure8.cpp solar.cpp
$ emacs makefile &
```

Have the program read in all the planetary positions from `planets-snapshot.dat` You'll now need to introduce a nontrivial gravitational constant that is consistent with the current system of units:

```
const double day = 86400; // [s]
const double aNep = 4496.6E9; // [m]
const double massSun = 1.9891E30; // [kg]
// G = 6.67384E-11 [m^3/kg.s^2]
const double G = ... ;
```

If you've done this correctly, you should be able to watch Earth complete a closed orbit in one year.

6. (5 points) Alter the code so that two copies of `gas` are maintained, one propagating forward in single time steps `dt` and another in triple steps of size `dt/3.0`. Set the value of `dt` small enough that the planetary positions and velocities agree to one part in 10^6 between the two copies over the course of a year.

7. (4 points) The values provided in `planets.dat` correspond to January 1, 1970. In the late 1970s, NASA took advantage of a "Grand Tour" alignment of the outer planets. Launch a satellite on August 20, 1977, the launch date of the Voyager 2 probe, with some plausible initial speed v_0 from a position $\boldsymbol{r}_{\text{earth}} + \rho(\cos\theta, \sin\theta)$ on Earth's surface. Here ρ is the radius of the Earth expressed in the appropriate dimensionless units. Vary v_0 and θ and see if you can manage to get the satellite to pass by Jupiter, Saturn, Uranus, and Neptune. Document your systematic efforts to do so.

8. (4 points) Pioneer 10 launched on March 3, 1972 and was the first spacecraft to reach the Jovian system. Following a 17 minute burn from its launch vehicle, Pioneer 10 left the vicinity of earth at 51 682 km/h. Using similar initial conditions, simulate a mission to Jupiter.