Final Project: Simulating interacting bodies

Project Overview

The project's objective is to simulate a system of interacting bodies with gravitational and electrostatic potentials and allow the verification of some of its simple physical properties.

The system is composed of N spherical objects of masses m_i, charges q_i and radii r_i in a three-dimensional space.

Any two bodies interact through the potentials

$$V_g(r_{ij}) = A_g \frac{m_i m_j}{r_{ij}^{l_g}} \tag{1}$$

$$V_e(r_{ij}) = A_e \frac{q_i q_j}{r_{ij}^{l_e}}$$

where V_g is the gravitational potential, V_e is the electrostatic potential, m_i and q_i are the i-th body mass and charge, respectively, and r_{ij} is the distance between bodies i and j. A_g, l_g, A_e & l_e are the parameters of the potentials. We are using parameters and not the known constants ($A_g = -G$, $l_g = 1$, $A_e = k$, $l_e = 1$) in order to be able to simulate more general potentials. The force acting between any two bodies is given by $\vec{F} = -\vec{\nabla} V$.

In this project we shall neglect the effect of the magnetic force (Lorentz force), i.e., we will consider only the electrostatic part of the electro-magnetic potential. This is done not only in order to avoid further complications but also due to the fact that in many naturally occurring systems the magnetic forces, exerted by the moving electric charges, are considerably weaker than the gravitational and electrostatic forces and therefore, can be neglected.

Based on the initial state of the system at a time t_0 the program will simulate the evolution of the system until some final time t_1 in time steps dt. Generally, dt should be chosen by the user so that the energy does not change much through the course of the simulation. The program will also display the system's projection on a two dimensional plane at n_{out} equally spaced times during the simulation.

The project requirements are divided into basic and extra requirements (specified below). Performing the basic requirements correctly will be rewarded with 80 points. An additional 30 points (up to a grade of 100) may be obtained by implementing the extra requirements.

Basic Requirements (80 points)

1. The system's parameters should be read from the standard input in the format given below. These parameters include the initial and final times, the time step, flags (one for controlling the display of the system at intermediate states, and another, used in the extra requirements), the coordinates of the view port, the potential parameters, the bodies' masses, charges, radii, and their initial positions and velocities.

2. The program should validate the initial data and give appropriate warning messages on invalid data parameters, such as a negative mass value, bodies on top of each other, etc. Note that the program should *not* exit upon invalid data.

3. The bodies interact through the potentials given in Eq. (1). The system is to be advanced from the initial time t_0 to the final time t_1 with discrete time steps dt. Further explanation of advancing the system in time is given below.

4. The bodies may pass one through another.

5. In case the display flag is set (i.e., its value is non zero) the system's projection on a two dimensional plane should be displayed graphically, at n_{out} equally spaced intermediate states. Note that since the actual display times might not fall exactly on computational time steps the display should be

on n_{out} *as equally spaced as possible* states. When displaying the bodies, different colors should be used for positively and negatively charged particles. The plane on which the system should be projected, when displayed, is given by the display flag's value: 1 for projection on the x-y plane, 2 for projection on the y-z plane and 3 for projection on the x-z plane. A description of the graphics functions to be used is given below.

6. When displaying the system, only bodies inside the view port should be displayed. Note that bodies may leave the displayed area. Those bodies continue to interact and might "return" to the displayed area.

7. When the simulation ends, the program should display on the same graph the total kinetic energy, E_k, the total potential energy, E_p, and the total energy, $E_{tot} = E_k + E_p$, at n_{out} equally spaced times. (Note that the graph's scales should be chosen wisely.)

8. The program should create an output file **coord.out** containing the three-dimensional coordinates of the bodies at each of the n_{out} equally spaced intermediate states. This file will be used to display a three dimensional simulation of the systems' evolution using a provided program. A description of the output file format is given below.

General Remarks for the Basic Requirements

1. The total number of bodies N is not fixed and is determined by the number of bodies in the input data file. However, you may assume that $1 \leq N \leq 50$.

2. You may assume that $n_{out} \leq 1000$.

3. In this project the system is numerically advanced from the initial time t_0 to the final time t_1. This is done by dividing the time span to equal steps of duration dt. The accuracy of the method depends upon the formulae for advancing from time t to time $t + dt$. We work with the *Verlet leap-frog* method (see **The Feynman Lectures on Physics**, first volume, chapter 9). In this method, the position and acceleration are computed at the times t_l, while the velocities are computed at times $t_l + dt/2$. That is, first we have to advance the initial velocities by the time $dt/2$,

$$\begin{aligned} \vec{a}_i(t_0) &= \sum_{j \neq i} \vec{F}_{ij}(\vec{x}_i(t_0))/m_i \\ \vec{v}_i(t_0 + dt/2) &= \vec{v}_i(t_0) + \vec{a}_i(t_0)dt/2 \end{aligned}$$

and then advance the system many times by the time dt,

$$\begin{aligned} &\vdots \\ \vec{x}_i(t_l) &= \vec{x}_i(t_l - dt) + \vec{v}_i(t_l - dt/2)dt \\ \vec{a}_i(t_l) &= \sum_{j \neq i} \vec{F}_{ij}(\vec{x}_i(t_l))/m_i \\ \vec{v}_i(t_l + dt/2) &= \vec{v}_i(t_l - dt/2) + \vec{a}_i(t_l)dt \\ &\vdots \end{aligned}$$

Note that as there are a number of particles, they should all be advanced simultaneously. In other words, make sure that any time layer is built according to the previous one.

Extra Requirements (30 Points)

1. Verify the virial theorem (cf. H. Goldstein, Classical Mechanics, chapter 3.4) which states in our case that in the center of mass system the following relation holds:

$$\langle E_k \rangle = -\frac{l_g}{2}\langle E_p^{(g)} \rangle - \frac{l_e}{2}\langle E_p^{(e)} \rangle \tag{2}$$

   where $\langle \ldots \rangle$ denotes averaging over a long enough time, i.e., the entire span of the simulation, $E_p^{(g)}$ is the gravitational potential energy, and $E_p^{(e)}$ is the electrostatic potential energy. Calculate and print the two sides of Eq. (2) and the relative difference between them in percent. You may assume that the parameters for simulations designed to validate this theorem are already given in the center of mass system.

2. Modify the program so that if the *cols* flag is set, then two bodies collide when their circumferences touch. The collisions are plastic, i.e., the two bodies become a single spherical body with the total masses, charges, volumes and momenta of the colliding bodies. The center of the newly formed body is at the center of mass of the two bodies. Note that upon collision of two bodies the total number of bodies in the system decreases. You may assume that the number density of bodies in the box is sufficiently low that there are no collisions involving three or more bodies.

3. Add a check of Kepler's 2^{nd} law (see Berkeley Physics Course, Vol. 1, Ch. 9) for the case in which the first body is the Sun and the second one a planet. The area swept by the radius vector in one time step (given by the formula $\frac{1}{2}|\vec{r} \times \vec{v}|dt$) at n_{out} equally spaced times should be written to the file **kepler.dat** and plotted as a graph at the end of the simulation. This check of Kepler's 2^{nd} law is done only for the first two bodies in the input file, assuming the first is the Sun and the second a planet. Note that your program should recognize a case where there is only a single body in the system and this check should be skipped.

Project technical details

The final version of the *C* program will be kept in the file **project.c** *on zoot*. You should submit a documented printout of the project. In addition, you will be orally examined to verify the validity of the program and your understanding of the project. It is *not* allowed to work in pairs or groups.

The Input File

The program will read the parameters from the standard input using the **scanf** command. That way, the use of different input files will be possible using redirection: `% a.out < infile`. The input file will consist of alternating header lines and data lines *in fixed order* as follows:

TIME — title
$t_0 \;\; t_1 \;\; dt \;\; n_{out} \;\; flag \;\; cols$
initial time, final time, time step, no. of intermediate steps, display flag and collisions flag
VIEWPORT — title
$x_0 \;\; x_1 \;\; y_0 \;\; y_1 \;\; z_0 \;\; z_1$ — the coordinates of the view port corners
POTENTIAL — title
$A_g \;\; l_g \;\; A_e \;\; l_e$ — the potential parameters
DATA — title
$m_1 \;\; q_1 \;\; r_1 \;\; x_1 \;\; y_1 \;\; z_1 \;\; v_{1x} \;\; v_{1y} \;\; v_{1z}$ — the mass, charge, radius, position and velocity of the bodies
$m_2 \;\; q_2 \;\; r_2 \;\; x_2 \;\; y_2 \;\; z_2 \;\; v_{2x} \;\; v_{2y} \;\; v_{2z}$

$\vdots$

$m_N \;\; q_N \;\; r_N \;\; x_N \;\; y_N \;\; z_N \;\; v_{Nx} \;\; v_{Ny} \;\; v_{Nz}$

A sample file with only two particles is given below:

```
TIME
0  30  0.001  1000  1  0
VIEWPORT
−10  10  − 10  10  − 10  10
POTENTIAL
−39.4771  1  0  1
DATA
1          0   0.3   0   0          0   0           0   0
0.00095    0   0.2   0   5.2028     0   −2.7546     0   0
```

More input examples will be available in the directory ∼compphys/inputs04. The file README in this directory will contain a short description of these files. It is recommended that you also create your own files in order to test various parts of your program.

The Output File

The program will write the bodies' three-dimensional coordinates at each of the n_{out} equally spaced intermediate times to the file **coord.out**. The format of the coordinates output file must be as follows: The first line of the file should contain the number of bodies in the system and the view-port corners in the following format:
$N \;\; x_0 \;\; x_1 \;\; y_0 \;\; y_1 \;\; z_0 \;\; z_1$
For each intermediate state n_i the coordinates of all the bodies should be printed. For each body m_j, at intermediate state n_i, a line should be printed in the format:
$n_i \;\; m_j \;\; q_j \;\; r_j \;\; x_j \;\; y_j \;\; z_j$
The coordinates should be printed in a chronological order. For each intermediate state, all the coordinates of the various bodies should be printed consecutively, maintaining a logical order in the body number. For example:

```
2     −10       10   −10   10           −10          10
0     1         0    0.3   0            0            0
0     0.00095   0    0.2   0            5.2028       0
1     1         0    0.3   0            0            0
1     0.00095   0    0.2   −0.082635    5.202144     0
⋮
999   1         0    0.3   −0.079281    0.009779     0
999   0.00095   0    0.2   0.898226     −5.090630    0
```

A matlab program which reads this output file and generates a three-dimensional simulation of the system's evolution will be provided. This three-dimensional simulation will be executed in a local matlab shell (under Windows). To do that, you should transfer the output files from your account (on ZOOT or COMFY) to the local PC, using FTP. Instructions on using FTP are given below.

The External Functions

In order to display the system graphically, several functions will be made available. To use them, the program should be compiled with the command **gccg** instead of **gcc**, and the line

```
#include  "/a/home/cc/physics/compphys/graphics.h"
```

should be added at the beginning of the program. The functions are:

- DefineGraph(double x0, double y0, double x1, double y1, char title[ ])

  The parameters $x0, y0, x1, y1$ are the boundaries of the visible rectangle and the title will be shown on the same window.

- EndGraph() End the graphics
- ClearScreen() Clear the screen
- Beep() Make a beeping sound
- DrawPnt(double x, double y) Draw a point at (x, y)
- DrawLine(double x1, double y1, double x2, double y2)
  Draw a line from point $(x1, y1)$ to point $(x2, y2)$
- DrawCirc(double x, double y, double r) Draw a circle of radius r around the point (x, y)
- SetColor(char color_name[ ])

  The string *color_name* can be one of the following:
  "default","white","black","red","green","blue","cyan", "magenta","yellow","orange".

Using the File Transfer Protocol (FTP)

Files can be transferred from the UNIX environment to the local Windows computer using the following procedure:

1. Start up Internet Explorer.
2. Type in the address bar `ftp://`*username*`@zoot.tau.ac.il`, where *username* stands for your UNIX user name (the name you use for logging in) and press the Enter key.
3. A dialog box should appear. If the box labeled "Remember my password" is marked, unmark it.
4. Type your password in the password edit box and click Ok.
5. Your home directory should appear inside Internet Explorer. Right-click the appropriate file, select "Copy to folder..." in the context menu and then select the directory on the Windows computer where the file is to be placed and click Ok.
6. Close Internet Explorer.

Work Instructions

- Read this project description thoroughly and make sure you understand all the project's requirements before starting to work on it.
- The program is quite involved. Try to break it into relatively simple tasks and handle each task separately. It is easier to find errors in a small portion of the code, dealing with a limited task.
- Thus, intelligent use of functions considerably facilitates the work.
- Use double precision variables whenever non-integer values are used (i.e., type double).
- Use meaningful variable names to enhance the legibility of the program.
- Document the program using comments. For example, it is recommended to explain briefly at the beginning of each function its purpose and the meaning of its main variables.
- Use global variables thoughtfully and sparingly, if at all.
- A well designed data structure for the bodies will make your life much easier.
- Testing your program under various parameters and initial conditions is essential in order to obtain a valid, working project.

Submitting a design paper

In preparation for writing the project each student will submit a general description of the project, outlining the principal functions and data structures which he/she will use in the project. An example of such a design paper, for a different project, will be provided.

Good luck!