A simple planetary orbit simulator, written in the C language

Why Feynman?

A little history

The name of this computer program, Feynman, comes from the Feynman's lectures on physics book. In this book Feynman suggests a very simple (yet effective) calculation procedure for the gravitational N-body problem. I took on his suggestion, it turned out to be quite good fun. I must say I kind of enjoy programming and operating systems theory in general, so yeah, why not learn some C by writing something interesting?

What is Feynman good for?

1. First of all, as a disclaimer, I am a beginner in C programming so the code is not guarranteed to have the best performance ever. Nor is it free from bugs. If you find any, let me know the issue.

2. Feynman aims at being capable of simulating the largest possible number of celestial bodies in a single home desktop/laptop. It has an option for the number of CPU cores to be used (either 1 core or 4 cores). In my sony 8GB RAM laptop I managed to simulate the coupled motion of around 40,000 celestial bodies.

3. It can be used for education since it is a very simple C program wired up with opengl and basic freeglut. You can modify it run your own opengl code as a school project or whatever.

Installation on GNU/Linux

Install depedencies

1. Make sure you have **gcc** installed on you system

2. Install **freeglut**. On Ubuntu systems type **sudo apt-get install freeglut3-dev**

Compile & run

1.Open up a terminal and navigate to the folder where your source code is. Then type:

gcc -o Feynman_077 main.c energy_calculation.c check_collisions.c drawing_and_input_controls.c update_motion.c -lpthread -lGL -lGLU -lglut -lm && ./Feynman_077

That is it! The feynman window should pop up with the planets on it!

Feynman should compile with 0 errors and 0 warnings using either GCC 7.1.1 or Clang 4.0.1.

I have also managed to compile and run Feynman on FreeBSD release 11.1 using GCC 5.4.0.

How to start Feynman

There are three text files in the Feynman-master folder **velocities.txt**, **positions.txt** and **mass.txt**. Just edit each of these files to add or remove planets. Let's for example have a look at the **velocities.txt** file:

```
copy/paste velocities  FORMAT -    vx(m/s) vy(m/s) with a space between coordinates)
0       0
47362 0
35020 0
30000 0
24077 0
13070 0
9690   0
6800   0
5430   0
```

The header of the file MUST be unchanged. The two columns indicate the component vx and vy in m/s of the velocity of each of the 9 CB's in the solar system (1 sun + 8 planets). To add or remove planets simply add or remove lines in each of the files **velocities.txt**, **positions.txt** and **mass.txt**. You can use a spreadsheet to generate 100's or 1000's of planet input data.

Simulation settings

The following simulation settings are available:

Time step (seconds) – The interval of time between each solver iteration. Large values for time step will make your simulation jump faster into the future but at the risk of incurring into errors.

Controls

Zoom in/out – Scroll your mouse wheel, if you still have a mouse that is.

Change point of view – Press the ‘o’ key on your keyboard if you still have one. You will enter the ‘fixed view’ mode, where a different CB is selected as the center of view everytime you press ‘o’. Eventualy, you will exit the fixed view mode when you have scrolled through all the CB’s if you keep pressing ‘o’.

Move the view around – If you are NOT on ‘fixed view’ mode, e.g. when you start Feynman, you can click the mouse wheel and drag around the view.

Calculation method

Note: Source code comments often refer to ‘CB’ which stands for Celestial Body.

General dynamics of the system

Each celestical body is treated as a point particle. Its trajectory in space is coupled to that of all the other CB’s in the simulation via newton’s 2^{nd} law of gravitation. The so called n-body problem. The exact details of the calculation are presented in Feynman’s book – Lectures on Physics Vol I, chapter 9, section 7 (Reference 1). This computer program attempts to follow Feynman’s implementation of the n-problem as close as possible. Any bugs you may find, you’re welcome to report them – your contribution will be added to the list.

Collisions between celestial bodies

By default, the collisions are considered to be perfectly inelastic. This means that the linear momentum is conserved for each collision, so:

$$v_{x3} = (m_1 v_{x1} + m_2 v_{x2})/(m_1 + m_2)$$

and

$$v_{y3} = (m_1 v_{y1} + m_2 v_{y2})/(m_1 + m_2)$$

This calculations can be found in the file **check_collisions.c**.

Energy calculation

If everything is working right then, assuming no inelastic collisions, conservation of mechanical energy must hold. That is, during a period of time t , the total mechanical energy of the system must equal some constant, according to the following equation (Reference 2):

$$E = K + U = \sum_i m v_i^2 - \sum_i \sum_j \frac{G M_i m_j}{r_{ij}}$$

where i and j are the indexes corresponding to the ith and jth celestial body. The variable r_{ij} corresponds to the distance matrix and indicates the distance between bodies i and j This equation is implemented in the **energy_calculation.c** source file.

Source code

Number of cores – The definition **Number_of_threads** at the beginning of the **main.c** can be change to determine how many CPU cores are used by Feynman to carry out the calculation. The values currently tested are 1 core or 4 cores. Other values may retrieve undefined behaviour. When using a large number of CB's (>5000) we recommend a value of 4 for machines with 8 or more CPU cores and a value of 1 otherwise. This program uses POSIX threads to run on multiple CPU cores, as explained in https://en.wikipedia.org/wiki/POSIX_Threads (Reference 3).

Feynman's general approach to variable scope – The way the program works is to define nearly every variable that matters to the calculation procedure as global. This means that the variables exist throughout all the program execution and get modified by function calls. Some of these variables could be set to local but the set of FREEGLUT functions don't accept extra input parameters.

As a consequence of working mainly on global variables, most function calls don't have arguments as these functions will act on the variables directly (such as **planet**, etc…). The function **void update_motion_parallel(void)** is an example of a function that does not receive any arguments.

The source files included in the 0.77 release are the following:

check_collisions.c

This code checks for collisions between CB's and assigns the corresponding resulting velocities (only inelastic colisions are supported at the moment). The code on this file is also responsible for doing memory management i.e. when 2 planets collide and generate 1 new planet, we need to reduce the number of planets from 2 (before collision) to 1 (after collision).

drawing_and_input_controls.c

Set of opengl commands to draw the Celestial Bodies (CB) on the screen. Basicly draws a 9 sided poligon in the shape of a circle whose radius is porpotional to mass of the respective CB. Also contains a set of freeglut commands to control mouse and keyboard input.

energy_calculation.c

Calculates kinetic and potential energy for each CB in the simulation.

main.c

Main part of the Feynman program.

Structures.h

Definition of several data structures, such as the **PLANET** structure. This sctructure contains the following members:

-double **x**, **y**, **vx**, **vy**, **ax**, **ay**, **m**; These are variables of position, velocity, acceleration and mass

-struct **color** of the type **struct RGB**; This is structure store the r,g,b values for each planet.

-int **planet_id**; Serves as an identifier. Each planet has a unique **planet_id** so it is easier to do memory management (deleting or creating planets) on the source file **check_collisions.h**.

update_motion.c – Update the acceleration of each indivudual CB (celestial body). Update the distance matrix, e.g. the distances between CB's. Used for singlecore and multicore **pthread** processing.

Author

This program and documentation was originally written by Nuno Ferreira. Feel free to contribute using the code hosting website (Gitlab at the time of this writing) and have your authorship of your modifications added here.

License

References

1. The Feynman Lectures on Physics, Matthew Sands, Richard Feynman, and Robert B. Leighton, 1964

2. From Calculus to Chaos: An Introduction to Dynamics, David Acheson, 1997

3. https://en.wikipedia.org/wiki/POSIX_Threads