Master CompuPhys - Python Numerical Project
Scope statements

Space battle around a Black Hole

Supervisor: D. Viennot, mailto:david.viennot@utinam.cnrs.fr

Project description:
The goal of the project is to develop the physics engine for a shooting mini-game, where a spaceship shoots at a target in the vicinity of a black hole. The physics engine must respect the laws of physics as much as possible.

The mini-game will be presented in 2D (equatorial plane of the black hole), with instructions entered on the keyboard and display of the dynamics of the objects taken into account.

Development environment :
The development of the programs will be done exclusively in Python language, with if necessary the use of the numpy, scipy, matplotlib (or other) libraries.
Technical supports can be found at the following addresses:
`https://docs.python.org/`
`https://docs.scipy.org/doc/`
`https://matplotlib.org`

Description of deliverables (functional description from the user point of view):
Depending on the degree of progress of the project, it may be rendered in a pre-alpha, alpha, beta or gold version.

In alpha version, the objective is to present a mini-game whose progress is as follows:

- The game takes place on a 2D plane corresponding to the equatorial plane of the black hole.
- At the start of the game, the ship and the target are placed in opposition to the black hole (the centers of the ship, the black hole and the target are aligned, ship and target are on either side of the black hole). The player chooses the distance from the event horizon (EH) of the black hole (see physics notions) at which he wishes to start, and his initial velocity vector. The distance to the EH and the initial speed of the target are chosen randomly by the program.
- Each round of the game, a time interval has passed. The ship and the target are moved under the effect of gravity (the current position as well as the trajectories followed until then are displayed on the screen).
- At each round, the player can choose:
  - to do nothing to wait for a more favorable position;
  - throw a heavy projectile (the player can choose the angle of the projectile but not its speed which is predefined); the projectile is then managed like the ship and the target by the game. The number of heavy projectiles available in a game being limited;
  - throw a light explosive projectile (also in limited number, with a predefined speed greater than that of the heavy projectile).
- When calculating trajectories:

  - If the target encounters a heavy projectile, if two heavy projectiles meet, or if the ship encounters one of its heavy projectiles, the collision will be assumed to be elastic.
  - If an explosive projectile hits the target, it is assumed to be destroyed and the game ends in victory.
  - If an explosive projectile returns to the ship, it is assumed to be destroyed and the game ends in a Game Over.
  - If an explosive projectile meets a heavy projectile, they are assumed to be destroyed, they disappear from the game and the game continues.
  - If a projectile hits the EH, it disappears into the black hole, and the game continues.
  - If the target arrives on the EH, it disappears into the black hole, and the game ends on a victory.
  - If the ship arrives on the EH, it disappears in the black hole, and the game ends with a Game Over.
  - If a projectile exceeds the limit of the playing area, it is assumed to be ejected, it disappears from the game and the game continues.
  - If the target exceeds the limit of the playing area, it is assumed to be ejected, and the game ends in victory.
  - If the ship exceeds the limit of the playing area, it is supposed to flee, and the game ends with a Game Over.

- If the player exhausts his shot possibilities without the game stopping, the game calculates future trajectories over a long period by applying the previous rules. If none of the cases for stopping the game occur (ship and target in stable orbits without possible collision), the game is declared void.

All instructions are entered during the game in console mode, and the graph of trajectories and positions is displayed for each round, the EH and the limit of the playing zone are also represented. A pre-alpha version without the heavy projectiles and associated elastic collisions may be developed first.

The beta version may provide the following additional features:

- Better management of the launch of the heavy projectile: taking into account the recoil of the ship when launching the heavy projectile (conservation of energy and momentum).
- Better management when leaving the playing area, following two possibilities to choose at the start of the game:
  - Comparison of the speed of the object with the escape velocity at the time of exit from the playing area in order to determine if it is actually released from the attraction of the black hole. If this is not the case, a long-term calculation of the trajectory is carried out to know the instant of the object's return to the playing zone (the object will have to reappear at this point in the game; if it is the ship or the target (game time is advanced at this moment).
  - Immediate return of the object in opposite place of its exit, on the principle of periodic boundary conditions.
- Damage management during collisions that become inelastic.

The gold version may have the following characteristics:

- Graphical interface to manage player instructions.
- Replacement of static graphs of positions and trajectories with animations.

Warhing! Please note, since Python graphical interfaces present portability problems from one OS to another, please provide, in addition to the version with interface (gold version), a version operating in console mode (beta version).

Description of expectations and constraints (functional and structural description from a developer point of view):

- A certain number of fixed parameters must be adjusted: masses of objects, speed of projectiles, size of the playing area in relation to the EH, duration elapsed between each round. Tests will have to be carried out to ensure optimal choices from the point of view of the interest of the game.
- Care must be taken to find a suitable way for the user to enter the initial distance to the black hole of the ship, the initial velocity vector, and the shooting angles. Typical values may be proposed.
- It will be necessary to define the "hitboxes" (collision masks) of the different elements, that is to say the area around an element where the collision occurs with the other elements (if we reduce the elements to points, no collision will occur).

Regarding trajectory calculations:

- In the pre-alpha version, we will use the approximate analytical expressions of the trajectories (see physical notions).
- In alpha version, we will integrate the post-Newtonian equations of motion (see physical notions).

Structurally, the code must be in the form of a main program managing the game, and python functions carrying out the different events (trajectory calculation, collision test, change of speeds upon collisions, exit from game area, graphic display, etc.).

Physical notions necessary for the project:
We denote by G the universal gravitation constant, c the speed of light, and M the mass of the black hole.

Escape velocity
The escape velocity is the minimum speed that an object must have to escape from the gravitational attraction of the black hole. The application of the principle of conservation of energy teaches us that for an object located at the distance r from the black hole it is

$$v_{esc}(r) = \sqrt{\frac{2GM}{r}} \tag{1}$$

The black holes
Black holes are bodies so dense that the escape velocity in their vicinity exceeds the speed of light. The laws of relativity prohibiting this speed from being exceeded, objects which are located under a limit called the Event Horizon (EH) cannot free themselves from the attraction of the black hole

and are therefore permanently trapped under the EH. This concerns material objects, such as light and even information. The area under the EH is "black" in the sense that nothing can come out of it.

The EH is a sphere of radius $R_S = \frac{2GM}{c^2}$ (we can verify that a body which is at a distance $r < R_S$ of a black hole does have an escape velocity greater than c). By definition, to be a black hole, a body must have sufficient density so that its surface is itself below R_S (known as the Schwarzschild radius).

Black holes also have a second boundary, a sphere of radius $1.5R_S$, called the photon sphere (PS). Under the PS, the relativistic driving inertial force due to rotation around the black hole becomes centripetal instead of centrifugal. In addition, photons can have compact trajectories. On the PS itself, the photons are in a circular orbit.

The natural units for the problem are therefore:

- the distances are expressed in relation to the Schwarzschild radius (for example we will speak of a length of $2.3R_S$);
- speeds are expressed as a fraction of the speed of light (for example we will speak of a speed of $0.3c$);

So in the program, by definition we will have $R_S = 1$ and $c = 1$ which constitute the basis of the reduced units for the calculations.

Integration of dynamics:
To calculate the trajectories, we will integrate the fundamental principle of Newton's dynamics in a polar frame centered on the black hole:

$$\vec{a} = \frac{\sum \vec{F}}{m} \tag{2}$$

We recall that the speed and acceleration in the polar frame are given by

$$\vec{v} = \dot{r}\vec{e}_r + r\dot{\theta}\vec{e}_\theta \tag{3}$$
$$\vec{a} = (\ddot{r} - r\dot{\theta}^2)\vec{e}_r + (2\dot{r}\dot{\theta} + r\ddot{\theta})\vec{e}_\theta \tag{4}$$

In the post-Newtonian approximation (application of Newton's laws to which relativistic corrections are added), the external forces are as follows:

- the gravitational force from Newton's law of universal attraction:

$$\frac{\vec{F}_G}{m} = -\frac{GM}{r^2}\vec{e}_r \tag{5}$$

- the first order relativistic correction:

$$\frac{\vec{F}_R}{m} = -\frac{3}{2}R_S\dot{\theta}^2\vec{e}_r \tag{6}$$

(To simplify the problem, we do not consider the clock shift effects induced by speed and gravity, we neglect the gravitational interactions between the ship, the target and the projectiles).
We thus obtain the radial equation:

$$\ddot{r} = -\frac{GM}{r^2} + (r - \frac{3}{2}R_S)\dot{\theta}^2 \tag{7}$$

It can be shown that there exist approximate analytical solutions to the post-Newtonian equations. These solutions provide the following polar equations for the orbits:

$$r_{\pm}(\theta) = \frac{\frac{q}{2} \pm \sqrt{\frac{q^2}{4} - \frac{3}{2}qR_S(1 + e\cos\theta)}}{1 + e\cos\theta} \tag{8}$$

with q and e parameters are defined by the initial conditions (length parameter and eccentricity). It will be noted that if the relativistic effects are negligible, i.e. $R_S \ll q$ (for example for the Earth R_S is of the order of a few centimeters while q is greater than the terrestrial radius), we find the polar equation of a conical curve (ellipse ($e < 1$), parabola ($e = 1$) or hyperbola ($e > 1$)). The angular velocity is defined by the conservation of the angular momentum:

$$r^2\dot{\theta} = \frac{L_0}{m} \tag{9}$$

The value of $\ell_0 = \frac{L_0}{m}$ is also defined by the initial conditions.
The solution $r_+(\theta)$ is to be applied if $r > 3R_S$, on the other hand we apply the solution $r_-(\theta)$ if $r < 3R_S$.
These approximate solutions neglect the phenomenon of precession of the pericenter which can in reality be very important.

Collisions:
Elastic collisions obey the laws of conservation of kinetic energy and momentum. For inelastic collisions, it is the total mechanical energy that is conserved (including the lack of energy used to create damage on the colliding bodies).

Algorithms to use:
We integrate the equations of motion over a duration T (time elapsed between two rounds of the game). The time interval $[t_i, t_i + T]$ (where t_i is the initial instant – date of the previous round –) is divided in N instants (with N large), $t_i < t_{i+1} < ... < t_{i+N} = t_i + T$, with $t_{k+1} - t_k = \frac{T}{N} = \Delta t$ the time discretisation step.

Calculation of $\theta(t)$:
We compute $\theta_k = \theta(t_k)$ by integration of eq. 9 using a first-order scheme given by the following pseudo-code:

For k from i to $i + N - 1$ do
  $\theta_{k+1} \leftarrow \theta_k + \frac{\ell_0}{r_k^2}\Delta t$
End for

The value of θ_i having been calculated in the previous round or being the initial condition $\theta_0 = 0$ in the first round, ℓ_0 is the constant calculated from the initial conditions. The value of r_k is either obtained from the approximate analytical formula equation 8 as $r_{\pm}(\theta_k)$ (alpha version), or obtained from the algorithm below (beta version).

Calculation of $r(t)$:
We compute $r_k = r(t_k)$ by integration of eq. 7 by the so-called Leapfrog algorithm which is given by the following pseudo-code:

For k from i to $i + N - 1$ do
  $r_{k+1} \leftarrow r_k + v_{k+1/2}\Delta t$
  $v_{k+3/2} \leftarrow v_{k+1/2} + a(r_{k+1})\Delta t$
End for

with $v_{k+1/2} = \dot{r}(t_k + \Delta t/2)$ the speed in the frame of reference accompanying the rotation of the

object around the black hole, the values of r_i and $v_{i+1/2}$ having been calculated in the previous round, or being defined by the initial condition r_0 and by $v_{1/2} = v_{r0} + a(r_0)\Delta t$ (v_{r0} being the radial component of the initial speed). The acceleration in the rotating frame of reference is given by $a(r) = -\frac{GM}{r^2} + (r - \frac{3}{2}R_S)\frac{\ell_0^2}{r^4}$.

Physical study:
Use the physics engine to study the orbits around the black hole and compare with the Keplerian orbits (without the relativistic correction). For example, we can study closed orbits and ejection orbits in the neighbourhood of the EH.