{EPITECH}

INTERSTONAR

LIGHT RAYS AND BLUE STONE

INTERSTONAR

binary name: interstonar
language: everything working on “the dump”
compilation: when necessary, via Makefile, including re, clean and fclean rules

- ✓ The totality of your source files, except all useless files (binary, temp files, obj files,…), must be included in your delivery.
- ✓ All the bonus files (including a potential specific Makefile) should be in a directory named bonus.
- ✓ Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

The stone is lost! Seeing how it affects people around it, the MIAOU team decided that the stone needed to be disposed of. As they cannot destroy it, they decided to send it in outer space on a rocket.
However, the rocket collisioned an asteroid on the solar system’s inner belt, and the stone is now lost among all the dust and regular rocks that stands there.

As a new member of the blue team, you are tasked to find the stone again. But how could you differenciate it among all these rocks? Fortunately, after several days of hard work, the answer appeared to you as quite obvious: because you remembered that the stone had special properties that made it last since its creation during the Jurassic era, you found that the easiest way to find it back would be to simply throw a regular rock on every stone of the inner belt, and check how they behave.
Before doing it in real life, you need to simulate and calibrate all your tools properly.

Project

You will have a program that works on two levels of analysis: a global planetary analysis, to check the journey of the rock, and a local asteroid analysis, that will simulate on which asteroid the rock will probably crash on.
Both modes take in parameters a TOML configuration file, as well as the position and initial velocity vector of the rock thrown in the scene.

Global Analysis

Your program will compute, given a configuration file describing a planetary scene, whether or not the rock collides a specific area. You will need to take into account the celestial bodies of the solar system to compute the trajectory of your rock.

Computations will be simplified:

- ✓ We assume a totally empty space, without any friction, dust nor dark matter/energy
- ✓ Gravitation is ruled by Newton's laws
- ✓ New system state will be computed using Euler integration method with a delta time of one hour
- ✓ The rock will have a weight of 1kg
- ✓ If the rock didn't reach the goal after 365 days, we consider that it is lost in the depth of space, and the program stops

As a reminder, Newton's law of universal gravitation states that:

$$F = G\frac{m_1 m_2}{r^2}$$

with G being the Newtonian constant of gravitation, m_1 and m_2 the masses of the two objects, and r the distance between the center of the masses.

$$G = 6.674 \times 10^{-11} m^3 kg^{-1} s^{-2}$$

The TOML configuration file will contain every information needed to describe the scene. As such, it will contain every information needed to describe the celestial bodies such as:

- ✓ Mass
- ✓ Radius
- ✓ Initial position

- ✓ Velocity vector (direction + speed)
- ✓ Name
- ✓ Goal (optional boolean setting this body as a goal for the rock)

Every value is in its SI base unit.

During the simulation, you must display the (X, Y, Z) position of the thrown rock at every iteration, until it either collides a goal, another celestial body, or if we reach the maximum number of steps.

If two or more celestial bodies collide during the simulation (other than the rock), they merge into a new body with the following properties:

- ✓ Mass is the sum of the colliding bodies
- ✓ Radius is computed by summing the volume of all the colliding bodies (as a reminder: $V = \frac{4}{3}\pi r^3$)
- ✓ New initial position is the mean of all positions
- ✓ New velocity vector will be the weighted mean of the velocity vectors by the mass of the objects
- ✓ The object will have a new name: the concatenation of all colliding bodies names sorted in ascii order ("foo" colliding "bar" will merge into "barfoo")
- ✓ It will become a goal if and only if at least one of the colliding bodies was a goal

Approximative values in the range of 1% of the expceted value are tolerated

Local Analysis

You will be given different shapes in space, and a starting trajectory. You will need to compute which shape the rock will intersect, if any, using sphere-assisted ray marching. In this scenario, the mass of the objects compensate each other, and are negligable.

O being the origin of the coordinate system, and x, y and z the axis, z being the vertical axis, the shapes that must be handled in this project are:

- ✓ Spheres
- ✓ Right circular cylinders around z axis
- ✓ Boxes parallel to the Oxy plane
- ✓ Torus parallel to the Oxy plane

The TOML configuration file will contain every information needed to describe every body of the

scene: their type, center position, and properties (cf. Examples)

We consider that the rock intersects a shape when the SDF becomes less than or equal to 0.1.
On the other hand, we consider that the rock will never intersect anything if no intersection is found after 1000 steps, or when the distance to the closest object becomes greater than 1000.

Examples

Global mode

```
Terminal
~/G-CNA-400> ./interstonar --help
USAGE
    ./interstonar [--global | --local] CONFIG_FILE Px Py Pz Vx Vy Vz

DESCRIPTION
   --global  Launch program in global scene mode. The CONFIG_FILE will describe a scene
containing massive spherical moving bodies of which at least one is a goal.

   --local  Launch program in local scene mode. The CONFIG_FILE will describe a scene
containing massless motionless shapes.

   Pi    Position coordinates of the rock

   Vi    Velocity vector of the rock

    CONFIG_FILE  TOML configuration file describing a scene.
```

Terminal

```
~/G-CNA-400> cat global_scene_example.toml
[[bodies]]
name = "Sun"
position = {x = 1.81899E+8, y = 9.83630E+8, z = -1.58778E+8}
direction = {x = -1.12474E+1, y = 7.54876E+0, z = 2.68723E-1}
mass = 1.98854e30
radius = 696_342_000
goal = true

[[bodies]]
name = "Mercury"
position = {x = -5.67576E+10, y = -2.73592E+10, z = 2.89173E+9}
direction = {x = 1.16497E+4, y = -4.14793E+4, z = -4.45952E+3}
mass = 3.30200E+23
radius = 2_439_000

[[bodies]]
name = "Venus"
position = {x = 4.28480E+10, y = 1.00073E+11, z = -1.11872E+09}
direction = {x = -3.22930E+04, y = 1.36960E+04, z = 2.05091E+03}
mass = 4.86850E+24
radius = 6_051_000
```

Terminal

```
~/G-CNA-400> cat global_scene_example2.toml
[[bodies]]
name = "Sun"
position = {x = 1.81899E+8, y = 9.83630E+8, z = -1.58778E+8}
direction = {x = -1.12474E+1, y = 7.54876E+0, z = 2.68723E-1}
mass = 1.98854e30
radius = 696_342_000

[[bodies]]
name = "Mercury"
position = {x = -5.67576E+10, y = -2.73592E+10, z = 2.89173E+9}
direction = {x = 1.16497E+4, y = -4.14793E+4, z = -4.45952E+3}
mass = 3.30200E+23
radius = 2_439_000
goal = true

[[bodies]]
name = "Venus"
position = {x = 4.28480E+10, y = 1.00073E+11, z = -1.11872E+09}
direction = {x = -3.22930E+04, y = 1.36960E+04, z = 2.05091E+03}
mass = 4.86850E+24
radius = 6_051_000
```

Terminal

```
~/G-CNA-400> ./interstonar --global global_scene_example.toml 1 2 3 4 5 6
At time t = 1: rock is (14401.000, 18002.000, 21603.000)
At time t = 2: rock is (301152889.708, 1628382987.862, -262805310.247)
Collision between rock and Sun
Mission success
```

Terminal

```
~/G-CNA-400> ./interstonar --global global_scene_example2.toml 1 2 3 4 5 6
At time t = 1: rock is (14401.000, 18002.000, 21603.000)
At time t = 2: rock is (301152889.708, 1628382987.862, -262805310.247)
Collision between rock and Sun
Mission failure
```


```
Terminal
~/G-CNA-400> ./interstonar --global global_scene_example2.toml -1.14746E+11 -1.96294E+11
-1.32908E+09 2.18369E+04 -1.01132E+04 -7.47957E+02 | tail -n 5
At time t = 8757: rock is (-5572802125.756, 237219245668.805, 4793938231.760)
At time t = 8758: rock is (-5656586227.782, 237224782907.202, 4796128760.832)
At time t = 8759: rock is (-5740369624.471, 237230289307.376, 4798318643.846)
At time t = 8760: rock is (-5824152304.948, 237235764871.540, 4800507880.576)
Mission failure
```

Local mode

```
Terminal
~/G-CNA-400> cat local_scene_example.toml
[[bodies]]
position = {x = 0, y = 0, z = 0}
type = "sphere"
radius = 1

[[bodies]]
position = {x = 0, y = 0, z = 0}
type = "cylinder"
radius = 1
height = 100 # optional. If not given, assume an infinite cylinder

[[bodies]]
position = {x = 0, y = 0, z = 0}
type = "box"
sides = {x = 10, y = 10, z = 10} # Gives the sides of the box in every direction

[[bodies]]
position = {x = 0, y = 0, z = 0}
type = "torus"
inner_radius = 3
outer_radius = 1
```

Terminal

```
~/G-CNA-400> cat infinite_cylinder.toml
[[bodies]]
position = {x = 0, y = 0, z = 12}
type = “cylinder”
radius = 1
```

Terminal

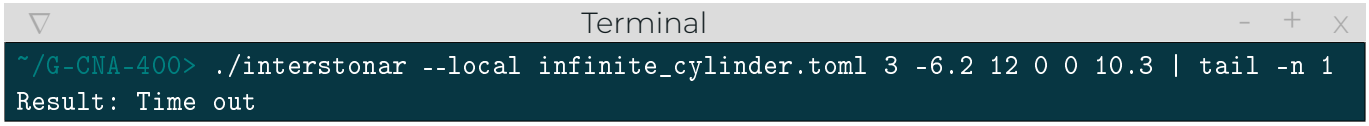
```
~/G-CNA-400> ./interstonar --local local_scene_example.toml 10 0 35 -1 0 -2
Rock thrown at the point (10.00, 0.00, 35.00) and parallel to the vector (-1.00, 0.00,
-2.00)
Sphere of radius 1.00 at position (0.00, 0.00, 0.00)
Cylinder of radius 1.00 and height 100.00 at position (0.00, 0.00, 0.00)
Box of dimensions (10.00, 10.00, 10.00) at position (0.00, 0.00, 0.00)
Torus of inner radius 3.00 and outer radius 1.00 at position (0.00, 0.00, 0.00)

Step 1: (5.98, 0.00, 26.95)
Step 2: (3.75, 0.00, 22.50)
Step 3: (2.52, 0.00, 20.04)
Step 4: (1.84, 0.00, 18.68)
Step 5: (1.46, 0.00, 17.93)
Step 6: (1.26, 0.00, 17.51)
Step 7: (1.14, 0.00, 17.28)
Step 8: (1.08, 0.00, 17.16)
Step 9: (1.04, 0.00, 17.09)
Result: Intersection
```

```
Terminal
~/G-CNA-400> ./interstonar --local local_scene_example.toml 10 0 35 -1 -1 -20
Rock thrown at the point (10.00, 0.00, 35.00) and parallel to the vector (-1.00, -1.00,
-20.00)
Sphere of radius 1.00 at position (0.00, 0.00, 0.00)
Cylinder of radius 1.00 and height 100.00 at position (0.00, 0.00, 0.00)
Box of dimensions (10.00, 10.00, 10.00) at position (0.00, 0.00, 0.00)
Torus of inner radius 3.00 and outer radius 1.00 at position (0.00, 0.00, 0.00)

Step 1: (9.55, -0.45, 26.02)
Step 2: (9.12, -0.88, 17.48)
Step 3: (8.72, -1.28, 9.34)
Step 4: (8.43, -1.57, 3.64)
Step 5: (8.26, -1.74, 0.22)
Step 6: (8.10, -1.90, -3.04)
Step 7: (7.94, -2.06, -6.13)
Step 8: (7.79, -2.21, -9.27)
Step 9: (7.53, -2.47, -14.36)
Step 10: (7.19, -2.81, -21.27)
Step 11: (6.85, -3.15, -27.97)
Step 12: (6.53, -3.47, -34.49)
Step 13: (6.21, -3.79, -40.87)
Step 14: (5.89, -4.11, -47.13)
Step 15: (5.59, -4.41, -53.30)
Step 16: (5.24, -4.76, -60.23)
Step 17: (4.65, -5.35, -72.10)
Step 18: (3.50, -6.50, -94.96)
Step 19: (1.24, -8.76, -140.26)
Step 20: (-3.28, -13.28, -230.64)
Step 21: (-12.31, -22.31, -411.27)
Step 22: (-30.37, -40.37, -772.46)
Step 23: (-66.49, -76.49, -1494.81)
Step 24: (-138.72, -148.72, -2939.50)
Result: Out of scene
```

```
Terminal
~/G-CNA-400> ./interstonar --local infinite_cylinder.toml 3 -6.2 12 0 0 10.3 | head -n 2
Rock thrown at the point (3.00, -6.20, 12.00) and parallel to the vector (0.00, 0.00,
10.30)
Cylinder of radius 1.00 and infinite height at position (0.00, 0.00, 12.00)
```


Terminal
~/G-CNA-400> ./interstonar --local infinite_cylinder.toml 3 -6.2 12 0 0 10.3 | tail -n 1
Result: Time out

Suggested Bonuses

- ✓ Use Runge-Kutta rk4 method instead of Euler's integration method (maths, yay!)
- ✓ Use Einstein's general relativity instead of Newton's law (physics, yay!)
- ✓ MORE SDF for MORE shapes
- ✓ Graphical rendering with shadersu sing ray marching
- ✓ Various optimizations (benchmark required)
- ✓ Full black hole simulation

v 1.2