Case Study: *N*-Body Problem

Outline

1. *N*-Body Simulation
2. Body Data Type
3. Universe Data Type

*N*-Body Simulation

Newton’s first law of motion states that a body in motion remains in motion at the same velocity unless acted on by an outside force.

Newton’s second law of motion explains how outside forces on a body affect its velocity.

The *N*-body simulation problem, originally formulated by Isaac Newton over 350 years ago, describes the motion of the *N* bodies, mutually affected by gravitational forces.

Body Data Type

A data type *Body* for moving bodies

<table>
<thead>
<tr>
<th>method</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body(r, v, mass)</td>
<td>a new body <em>b</em> with mass <em>mass</em> at position <em>r</em> moving at velocity <em>v</em></td>
</tr>
<tr>
<td>b.move(f, dt)</td>
<td>move <em>b</em> by applying force <em>f</em> for <em>dt</em> seconds</td>
</tr>
<tr>
<td>b.forceFrom(a)</td>
<td>force vector from body <em>a</em> on <em>b</em></td>
</tr>
<tr>
<td>b.draw()</td>
<td>draw <em>b</em> to standard draw</td>
</tr>
</tbody>
</table>

Note that *Body* is a *Vector* client — the values of the data type are *Vector* objects that carry the body’s position and velocity, as well as a float that carries the mass.

We use Newton’s second law (\( F = ma \)) for updating the position and velocity of a body due to a given force vector \( f \) and amount of time \( t \):

\[
a = f \cdot \text{scale}(1.0 / \text{mass})
\]

\[
v = v + a \cdot \text{scale}(dt)
\]

\[
r = r + v \cdot \text{scale}(dt)
\]
We use Newton’s law of universal gravitation (\( F = -\frac{Gm_1m_2}{r^2} \hat{r} \)) for computing the force imposed on one body by another:

\[
\begin{align*}
G &= 6.67\times10^{-11} \\
\delta &= b._r - a._r \\
\text{dist} &= \text{abs} (\delta) \\
\text{magnitude} &= \frac{G \times a._mass \times b._mass}{\text{dist} \times \text{dist}} \\
f &= \delta \cdot \text{direction}().\text{scale}(\text{magnitude})
\end{align*}
\]

The data type `Universe` models the universe:

- **Method**: `Universe(file)`
  - a new universe u built from a description in file

- **Method**: `u.increaseTime(dt)`
  - update u by simulating the universe for dt seconds

- **Method**: `u.draw()`
  - draw universe u to standard draw

The constructor reads the universe parameters and body descriptions from a file that contains the following information:

- The number of bodies
- The radius of the universe
- The position, velocity, and mass of each body

Example (parameters for a 2-body system):

```plaintext
$ more 2body.txt
2
5.0e10
0.000 4.5e10 1.0e04 0.000 1.5e30
0.0e0 -4.5e10 -1.0e04 0.000 1.5e30
```
```python
def increaseTime(self, dt):
n = len(self._bodies)
f = stdarray.create1D(n, Vector([0, 0]))
for i in range(n):
    for j in range(n):
        if i != j:
            bodyi = self._bodies[i]
            bodyj = self._bodies[j]
            f[i] = f[i] + bodyi.forceFrom(bodyj)
for i in range(n):
    self._bodies[i].move(f[i], dt)
def draw(self):
    for body in self._bodies:
        body.draw()
def main():
    filename = sys.argv[1]
    dt = float(sys.argv[2])
    universe = Universe(filename)
    while True:
        universe.increaseTime(dt)
        stddraw.clear()
        universe.draw()
        stddraw.show(10)
if __name__ == '__main__':
    main()
```

```
A 2-body system
$ more 2body.txt
2
5.0e10
0.0e00 4.5e10 1.0e04 0.0e00 1.5e30
0.0e00 -4.5e10 -1.0e04 0.0e00 1.5e30
$ python universe.py 2body.txt 20000

A 3-body system
$ more 3body.txt
3
1.25e11
0.0e00 0.0e00 0.05e04 0.0e00 5.97e24
0.0e00 4.5e10 3.0e04 0.0e00 1.989e30
0.0e00 -4.5e10 -3.0e04 0.0e00 1.989e30
$ python universe.py 3body.txt 20000

A 4-body system
$ more 4body.txt
4
5.0e10
-3.5e10 0.0e00 0.0e00 1.4e03 3.0e28
-1.0e10 0.0e00 0.0e00 1.4e04 3.0e28
1.0e10 0.0e00 0.0e00 -1.4e04 3.0e28
3.5e10 0.0e00 0.0e00 -1.4e03 3.0e28
$ python universe.py 4body.txt 20000
```