N-BODY ASSIGNMENT

- n-body simulation
- problem decomposition
- the physics
- bugs
- universes

https://introcs.cs.princeton.edu
N-BODY ASSIGNMENT

- n-body simulation
- problem decomposition
- the physics
- bugs
- universes
N-body simulation

**Problem.** Simulate the motion of $n$ bodies, subject to Newton’s laws.
Physics and math

Newton’s law of gravity. \[ F = \frac{G m_1 m_2}{r^2} \]

Newton’s second law of motion. \[ F = m a \]

“Leapfrog” method. For numerical integration of differential equations.

don’t worry (this is not a math or physics course)
Applications. Cosmology, semiconductors, fluid dynamics, ....
Programming goals

Goal 1. Use standard input, standard output, and standard drawing for I/O.
Goal 2. Use parallel arrays.

Goal 3. Decompose a large program into small, manageable parts.
Before you begin

Read. Section 1.4 and 1.5 of textbook and assignment specification.

Make sure you understand these two programs from lecture and precept:

- BouncingBallDeluxe.java \(\rightarrow\) animation
- Students.java \(\rightarrow\) standard input and arrays

Check that stdlib.jar is available to Java.

- Should already be configured if you used our auto-installer.
- Remember to use javac-introcs and java-introcs at command line.

Download project folder nbody.zip:

- Contains universe files, such as planets.txt.
- Contains accompanying image and sound files, such as earth.gif.
Have you read Sections 1.4 and 1.5 of book and assignment specification?

A. Yes.

B. No.
N-BODY ASSIGNMENT

- n-body simulation
- problem decomposition
- the physics
- bugs
- universes
Incremental development

Develop program incrementally, decomposing into 6 simpler steps:

1. Parse command-line arguments.
2. Read universe from standard input.
3. Initialize standard drawing.
4. Play music on standard audio.
5. Simulate the universe.
   A. Calculate net forces.
   B. Update velocities and positions.
   C. Draw universe to standard drawing.
6. Print universe to standard output.

**Advice.** Although code will appear in the program in the order 1–6, we recommend incrementally developing in the order 1, 2, 6, 3, 4, 5B, 5C, 5A.

**Q.** Why?

A. Facilitates testing and debugging.
public class NBody {
    public static void main(String[] args) {

        // Step 1. Parse command-line arguments.

        // Step 2. Read universe from standard input.

        // Step 3. Initialize standard drawing.


        // Step 5. Simulate the universe.
            // Step 5A. Calculate net forces.
            // Step 5B. Update velocities and positions.
            // Step 5C. Draw universe to standard drawing.

        // Step 6. Print universe to standard output.
    }
}
Command-line arguments

Step 1. Parse command-line arguments.
- Read stopping time $T$ and time increment $\Delta t$ from command line.
- Print values of each variable (as debugging aid).

Note. Easy, but you should still test it!

```
~/Desktop/nbody> java NBody 10 1
tau = 10.0
dt = 1.0

~/Desktop/nbody> java NBody 157788000.0 25000.0
tau = 1.57788E8
dt = 25000.0
```
### Standard input

**Step 2.** Read universe from standard input.

```plaintext
~/Desktop/nbody> more planets.txt

5          number of bodies n
2.50e+11   radius of universe

<table>
<thead>
<tr>
<th>Mass</th>
<th>Initial X-Position</th>
<th>Initial Y-Position</th>
<th>Initial X-Velocity</th>
<th>Initial Y-Velocity</th>
<th>Image Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4960e+11</td>
<td>0.0000e+00</td>
<td>0.0000e+00</td>
<td>2.9800e+04</td>
<td>5.9740e+24</td>
<td>earth.gif</td>
</tr>
<tr>
<td>2.7900e+11</td>
<td>0.0000e+00</td>
<td>0.0000e+00</td>
<td>2.4100e+04</td>
<td>6.4190e+23</td>
<td>mars.gif</td>
</tr>
<tr>
<td>5.7900e+10</td>
<td>0.0000e+00</td>
<td>0.0000e+00</td>
<td>4.7900e+04</td>
<td>3.3020e+23</td>
<td>mercury.gif</td>
</tr>
<tr>
<td>0.0000e+00</td>
<td>0.0000e+00</td>
<td>0.0000e+00</td>
<td>0.0000e+00</td>
<td>1.9890e+30</td>
<td>sun.gif</td>
</tr>
<tr>
<td>1.0820e+11</td>
<td>0.0000e+00</td>
<td>0.0000e+00</td>
<td>3.5000e+04</td>
<td>4.8690e+24</td>
<td>venus.gif</td>
</tr>
</tbody>
</table>

This file contains the sun and the inner 4 planets of our Solar System.

optional description (not needed for simulation)
```
**Standard input**

**Step 2.** Read universe from standard input.
- Read number of bodies \( n \) from standard input.
- Read \textit{radius} of universe standard input.
- Create 6 parallel arrays, each of length \( n \), to store the 6 pieces of information characterizing a body.
- Read data associated with each body and store in arrays.

**Hint.** Recall Students.java.

Q. How to test?

```
~/Desktop/nbody> java-introcs NBody 157788000.0 25000.0 < planets.txt
[no output]
```

A. Do Step 6 (print universe).
Step 6. Print universe to standard output.

- Print \( n \) and \( radius \).
- Write a loop to iterate over the 6 parallel arrays.
- Use StdOut.printf() for formatted output.

```shell
~/Desktop/nbody> java-introcs NBody 157788000.0 25000.0 < planets.txt
5
2.50e+11
1.4960e+11 0.0000e+00 0.0000e+00 2.9800e+04 5.9740e+24  earth.gif
2.2790e+11 0.0000e+00 0.0000e+00 2.4100e+04 6.4190e+23  mars.gif
5.7900e+10 0.0000e+00 0.0000e+00 4.7900e+04 3.3020e+23  mercury.gif
0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 1.9890e+30  sun.gif
1.0820e+11 0.0000e+00 0.0000e+00 3.5000e+04 4.8690e+24  venus.gif
```
N-body assignment: quiz 2

Which `StdOut.printf()` statement was used to print each body?

A. `StdOut.printf("%f %f %f %f %f %s\n", ...);`
B. `StdOut.printf("%1.4f %1.4f %1.4f %1.4f %1.4f %12s\n", ...);`
C. `StdOut.printf("%11.4f %11.4f %11.4f %11.4f %11.4f %12s\n", ...);`
D. `StdOut.printf("%11.4e %11.4e %11.4e %11.4e %11.4e %12s\n", ...);`

~/Desktop/nbody> java-introcs NBody 157788000.0 25000.0 < planets.txt
5
2.50e+11
1.4960e+11 0.0000e+00 0.0000e+00 2.9800e+04 5.9740e+24 earth.gif
2.2790e+11 0.0000e+00 0.0000e+00 2.4100e+04 6.4190e+23 mars.gif
5.7900e+10 0.0000e+00 0.0000e+00 4.7900e+04 3.3020e+23 mercury.gif
0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 1.9890e+30 sun.gif
1.0820e+11 0.0000e+00 0.0000e+00 3.5000e+04 4.8690e+24 venus.gif
Step 3. Initialize standard drawing.

- Enable double buffering. \(\xrightarrow{\text{call StdDraw.enableDoubleBuffering()}}\)
- Default \(x\)- and \(y\)-scale supports coordinates between 0 and 1.
- Change scale to be between \(-radius\) and \(+radius\). \(\xrightarrow{\text{call StdDraw.setXscale() and StdDraw.setYscale()}}\)

**Q.** How to test?
Standard audio

Step 4. Play music.

- Call StdAudio.playInBackground("2001.wav").
- Easy.
N-BODY ASSIGNMENT

- n-body simulation
- problem decomposition
- the physics
- bugs
- universes
The simulation loop (the "big time loop")

**Step 5.** Simulate the universe. At each time step $t$:

A. Calculate the net force on each body.

B. Update the velocities and positions.

C. Draw the universe.

**Q.** In which order should I implement these 3 sub-steps?

A. 5B, 5C, 5A because calculating forces is hardest.

**Q.** Can I interleave steps 5A, 5B, and 5C?

A. No. Not only is it bad design, but it ruins the physics.
Measuring time

Time loop. From $t = 0$ (inclusive) to $T$ (exclusive), incrementing by $\Delta t$.

**Hint.** Easy, but also easy to get wrong. $\Rightarrow$ Test!

```
~/Desktop/nbody> java-introcs NBody 13.0 2.5 < planets.txt
  t = 0.0
  t = 2.5
  t = 5.0
  t = 7.5
  t = 10.0
  t = 12.5

~/Desktop/nbody> java-introcs NBody 10.0 2.5 < planets.txt
  t = 0.0
  t = 2.5
  t = 5.0
  t = 7.5  \< don't include 10.0
```
Updating the velocities and positions

**Step 5B.** [for now, all forces and accelerations are 0]

- Update the velocity of each body: \( v_x = v_x + a_x \Delta t, \quad v_y = v_y + a_y \Delta t. \)
- Update the position of each body: \( p_x = p_x + v_x \Delta t, \quad p_y = p_y + v_y \Delta t. \)

**Q.** How to test?

**A.** Use an artificial universe that is easy to check by hand.

```bash
~/Desktop/nbody> more 3body-zero-gravity.txt
3
512
  0  0  1  1  1e-30  earth.gif
128  0  2  1  1e-40  venus.gif
  0 128  1  2  1e-50  mars.gif
```

```bash
~/Desktop/nbody> java-intros NBody 191 1 < 3body-zero-gravity.txt
3
5.12e+02
  1.9200e+02  1.9200e+02  1.0000e+00  1.0000e+00  1.0000e-30  earth.gif
  5.1200e+02  1.9200e+02  2.0000e+00  1.0000e+00  1.0000e-40  venus.gif
  1.9200e+02  5.1200e+02  1.0000e+00  2.0000e+00  1.0000e-50  mars.gif
```
Drawing the universe

Step 5C.

- Draw background image.
- Write loop to display $n$ bodies.
- Call `StdDraw.show()` to display results on screen.
- Call `StdDraw.pause(20)` to control animation speed.

planets.txt  kaleidoscope.txt
If there are $n$ bodies and $m$ time steps, how many times in total should my program call each of `StdDraw.show()` and `StdDraw.pause()`?

A. 1
B. $m$
C. $n$
D. $mn$
Calculating the force (between two bodies at time $t$)

**Step 5A.**

- Apply Newton’s law of gravity.
- A bit of high-school geometry (formulas provided).

**distance between two bodies**

$$r = \sqrt{\Delta x^2 + \Delta y^2}$$

$$\cos \theta = \frac{\Delta x}{r}, \quad \sin \theta = \frac{\Delta y}{r}$$

**force between two bodies**

$$F = \frac{Gm_1 m_2}{r^2}$$

$$F_x = F \cos \theta$$

$$F_y = F \sin \theta$$

**Remark.** $\Delta x$ and $\Delta y$ are signed numbers. $\Delta x$ is positive for force of sun on earth; $\Delta x$ is negative for force of earth on sun.
Calculating the force (between all pairs of bodies at time t)

Principle of superposition. Add all pairwise forces.

\[
\vec{F}_{\text{earth}} = \vec{F}_{\text{mars}\rightarrow\text{earth}} + \vec{F}_{\text{mercury}\rightarrow\text{earth}} + \vec{F}_{\text{sun}\rightarrow\text{earth}} + \vec{F}_{\text{venus}\rightarrow\text{earth}}
\]

How to implement?

- Need two extra arrays fx[] and fy[]. Why?
- Need to examine all pairs of bodies.

Warmup. Enumerate all pairs of bodies.

```bash
~/Desktop/nbody> javac-introcs AllPairs.java
~/Desktop/nbody> java-introcs AllPairs 4
0-1 0-2 0-3
1-0 1-2 1-3
2-0 2-1 2-3
3-0 3-1 3-2
```

~don't include 0-0, 1-1, 2-2, or 3-3~
Which of these steps will result in a triple nested loop?

A. Step 2.
B. Step 5A.
C. Step 5B.
D. Step 5C.
E. None of the above.
N-BODY ASSIGNMENT

- n-body simulation
- problem decomposition
- the physics
- bugs
- universes
Advice

- Develop code incrementally; test after each step.
- Test, test, test.
- Take your time!
- Start early!
- Seek help when you get stuck.
- Write outline of code (using comments) first; fill in code later.

Take my advice....
Command-line bug

```
~/Desktop/nbody> java-introcs NBody 157788000.0 25000.0 > planets.txt
<Ctrl-C>

~/Desktop/nbody> java-introcs NBody 157788000.0 25000.0 < planets.txt
Exception in thread "main" java.util.NoSuchElementException
    at java.util.Scanner.throwFor(Scanner.java:907)
    at java.util.Scanner.next(Scanner.java:1530)
    at java.util.Scanner.nextInt(Scanner.java:2160)
    at java.util.Scanner.nextInt(Scanner.java:2119)
    at StdIn.readInt(StdIn.java:319)
    at NBody.main(NBody.java:54)

~/Desktop/nbody> more planets.txt
[it's empty; you overwrote it!]
```
Visual bugs

- no motion

- no gravity (or forces)
Visual bugs

no double buffering

planets repel one another
(sign error)
Visual bugs

wrong force loop
(only some pairs checked)

cut-and-paste error
(x vs. y)
N-BODY ASSIGNMENT

- n-body simulation
- problem decomposition
- the physics
- bugs
- universes
Other universes

planetsparty.txt
(created by Mary Fan)

twinbinaries.txt
(created by David Costanzo)
Other universes

chaosblossum.txt
(created by Erik Keselica)

galaxy.txt
(created by Matt Tilghman)