1. Announcements
2. A Tour of the Command Line
3. N-Body
raise your hand and ask

ask on Ed
(use ❤️ to upvote)
HELP!

Office Hours

Intro COS Lab Schedule
For help with debugging programming assignments: Intro COS Lab, located in Lewis Library

McGraw Tutoring Schedule
For help with understanding concepts: McGraw

Instructors Office Hours Schedule

COS 126 Office Hours - In-Person and Zoom Links

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>Sep 1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>9am Drop in - Fri</td>
<td>9am Prachi OH</td>
<td>9am Prachi OH</td>
<td>9am Prachi OH</td>
<td>9am Prachi OH</td>
<td>9am Prachi OH</td>
<td>9am Prachi OH</td>
</tr>
<tr>
<td>1pm Pradnya's OH</td>
<td>10:15am Sebastian</td>
<td>10am Owen's OH</td>
<td>10:30am Owen's OH</td>
<td>10am Owen's OH</td>
<td>10am Owen's OH</td>
<td>10am Owen's OH</td>
</tr>
<tr>
<td>+4 more</td>
<td>+4 more</td>
<td>+4 more</td>
<td>+4 more</td>
<td>+4 more</td>
<td>+4 more</td>
<td>+4 more</td>
</tr>
<tr>
<td>3pm Donna In-Person</td>
<td>3pm Maya Gupta</td>
<td>3pm Nick Sudarski</td>
<td>3pm Nick Sudarski</td>
<td>3pm Nick Sudarski</td>
<td>3pm Nick Sudarski</td>
<td>3pm Nick Sudarski</td>
</tr>
<tr>
<td>+4 more</td>
<td>+4 more</td>
<td>+4 more</td>
<td>+4 more</td>
<td>+4 more</td>
<td>+4 more</td>
<td>+4 more</td>
</tr>
</tbody>
</table>
Tour of the command-line

Open IntelliJ → Open p04-stdin-stddraw project → Select case-study-terminal → Open terminal
N-BODY ASSIGNMENT

- n-body simulation
- problem decomposition
- the physics
- bugs
- universes
Problem. Simulate the motion of \( n \) bodies, subject to Newton’s laws.
Physics and math

Newton’s law of gravity. \[ F = \frac{Gm_1 m_2}{r^2} \]

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, ....

https://www.youtube.com/watch?v=ua7YIN4eL_w
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.
    Including writing comments before you code!

key to becoming a good programmer
Before you begin

Read. Assignment specification (and watch lectures/do reading)

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

- BouncingBallDeluxe.java
- Students.java

Check that stdlib.jar is available to Java → nbody> java-introcs StdDraw

- 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.

nement specifi
and watch lectures/do reading)

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

- BouncingBallDeluxe.java
- Students.java

Check that stdlib.jar is available to Java → nbody> java-introcs StdDraw

- 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 the 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!

```
nbody> java-introcs NBody 10 1
tau = 10.0
delta = 1.0
nbody> java-introcs NBody 157788000.0 25000.0
tau = 1.57788E8
delta = 25000.0
```
### Step 2. Read universe from standard input.

**nbody> more planets.txt**

<table>
<thead>
<tr>
<th>n</th>
<th>radius of universe</th>
<th>data for one body</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.50e+11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4960e+11 0.0000e+00 0.0000e+00 2.9800e+04 5.9740e+24</td>
<td>earth.gif</td>
<td></td>
</tr>
<tr>
<td>2.2790e+11 0.0000e+00 0.0000e+00 2.4100e+04 6.4190e+23</td>
<td>mars.gif</td>
<td></td>
</tr>
<tr>
<td>5.7900e+10 0.0000e+00 0.0000e+00 4.7900e+04 3.3020e+23</td>
<td>mercury.gif</td>
<td></td>
</tr>
<tr>
<td>0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 1.9890e+30</td>
<td>sun.gif</td>
<td></td>
</tr>
<tr>
<td>1.0820e+11 0.0000e+00 0.0000e+00 3.5000e+04 4.8690e+24</td>
<td>venus.gif</td>
<td></td>
</tr>
</tbody>
</table>

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

**Step 2.** Read universe from standard input.
- Read number of bodies $n$ from standard input.
- Read *radius* of universe from 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?

**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.

```
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", ...);`

https://introcs.cs.princeton.edu/java/11cheatsheet/#printf

formatted printing.

```

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. 📣 call StdDraw.enableDoubleBuffering()
- Default $x$- and $y$-scale supports coordinates between 0 and 1.
- Change scale to be between $-radius$ and $+radius$. 📣 call StdDraw.setXscale() and StdDraw.setYscale()

**Q.** How to test?

- For desired coordinates: (0, 0), (1, 1), (++radius, ++radius), (++radius, ++radius)
- For default coordinates: (0, 0), (1, 1), (++radius, ++radius), (++radius, ++radius)
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.

similar to BouncingBallDeluxe.java
(but no forces to update)

even though they will appear
in your code in the order 5A, 5B, 5C

need position of all bodies at time $t$
(not some at time $t$ and others at time $t + \Delta t$)
Measuring time

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

Hint. Easy, but also easy to get wrong. ⇒ Test!

```
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  # don't include 13.0
  t = 12.5
```

```
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: $v_x = v_x + a_x \Delta t$, $v_y = v_y + a_y \Delta t$.
- Update the position of each body: $p_x = p_x + v_x \Delta t$, $p_y = p_y + v_y \Delta t$.

**Q.** How to test?

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

```bash
nbody> cat 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
nbody> java-introcs NBody 191 1 < 3body-zero-gravity.txt
3
5.12e+02
 1.9100e+02 1.9100e+02 1.0000e+00 1.0000e+00 1.0000e-30  earth.gif
 5.1000e+02 1.9100e+02 2.0000e+00 1.0000e+00 1.0000e-40  venus.gif
 1.9100e+02 5.1000e+02 1.0000e+00 2.0000e+00 1.0000e-50  mars.gif
```
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

planets should move off screen with constant velocity
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. $m \times n$
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).

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

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

\[ F = \frac{Gm_1m_2}{r^2} \]

\[ F_x = F \cos \theta \]

\[ F_y = F \sin \theta \]

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 \( f_x[] \) and \( f_y[] \). 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
```

Don't include 0–0, 1–1, 2–2, or 3–3.
N-BODY ASSIGNMENT

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

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

Take my advice....
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)
Other universes

suninterference.txt
(created by Kevin Henneck)

awesome.txt
(created by John Colosimo)
Next Week:
How to take a programming exam
MAY THE FORCE BE WITH YOU!!!