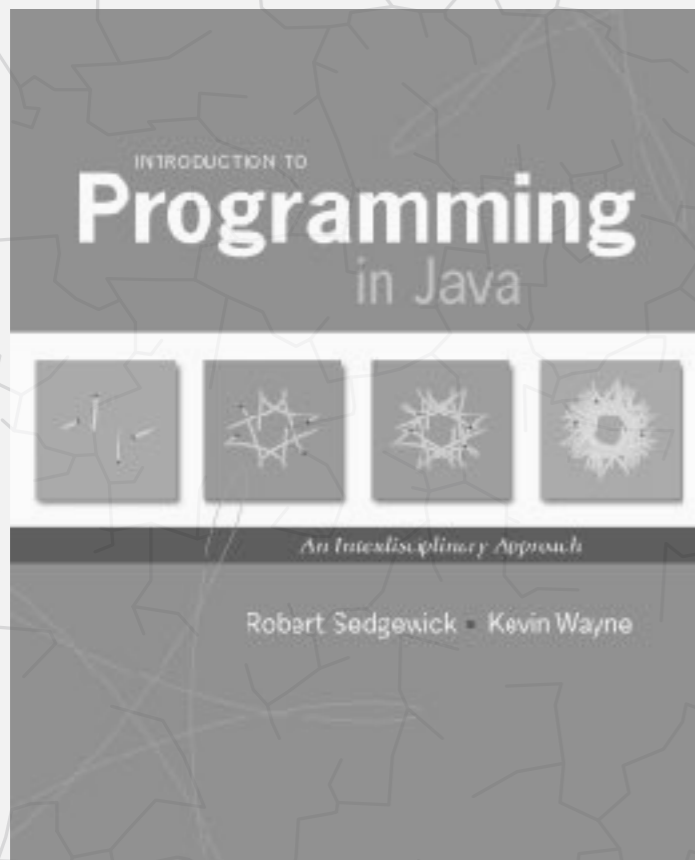


<http://introcs.cs.princeton.edu>

## ASSIGNMENT 2 TIPS AND

---

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



<http://introcs.cs.princeton.edu>

## ASSIGNMENT 2 TIPS AND

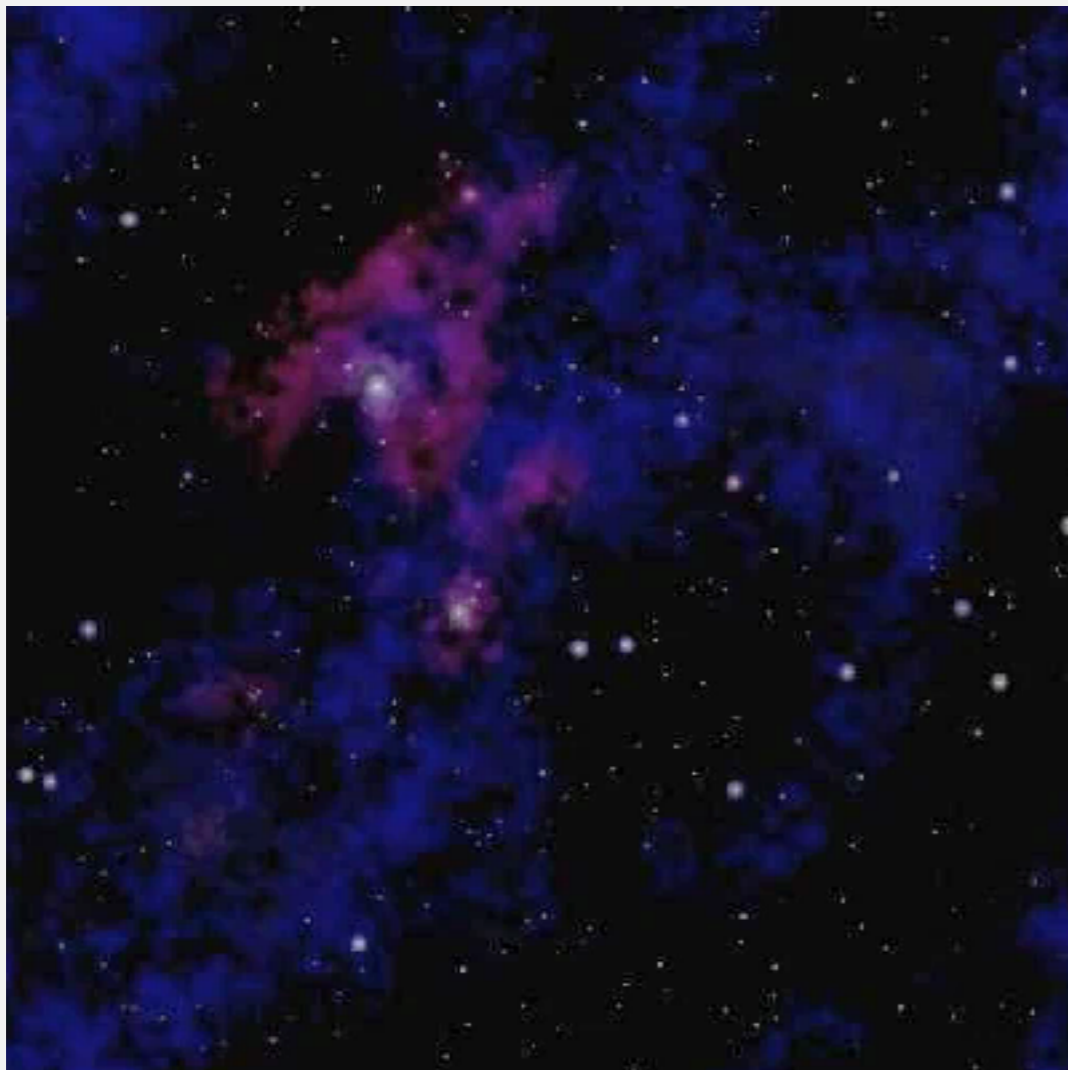
---

- ▶ *overview*
- ▶ *problem decomposition*
- ▶ *the physics*
- ▶ *bugs*
- ▶ *universes*

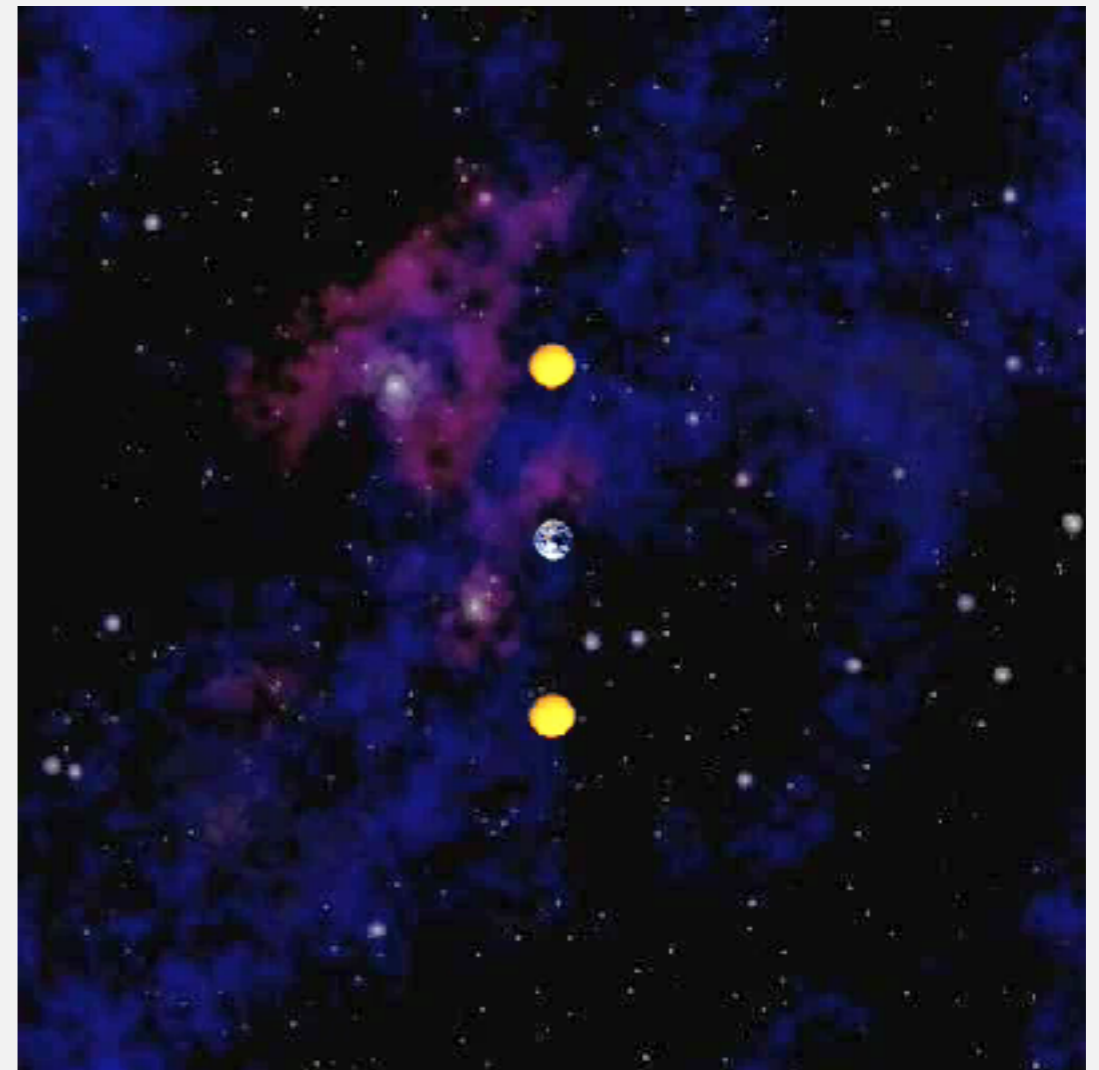
# N-body simulation

---

Simulate the motion of  $n$  bodies, subject to Newton's laws.



planets.txt



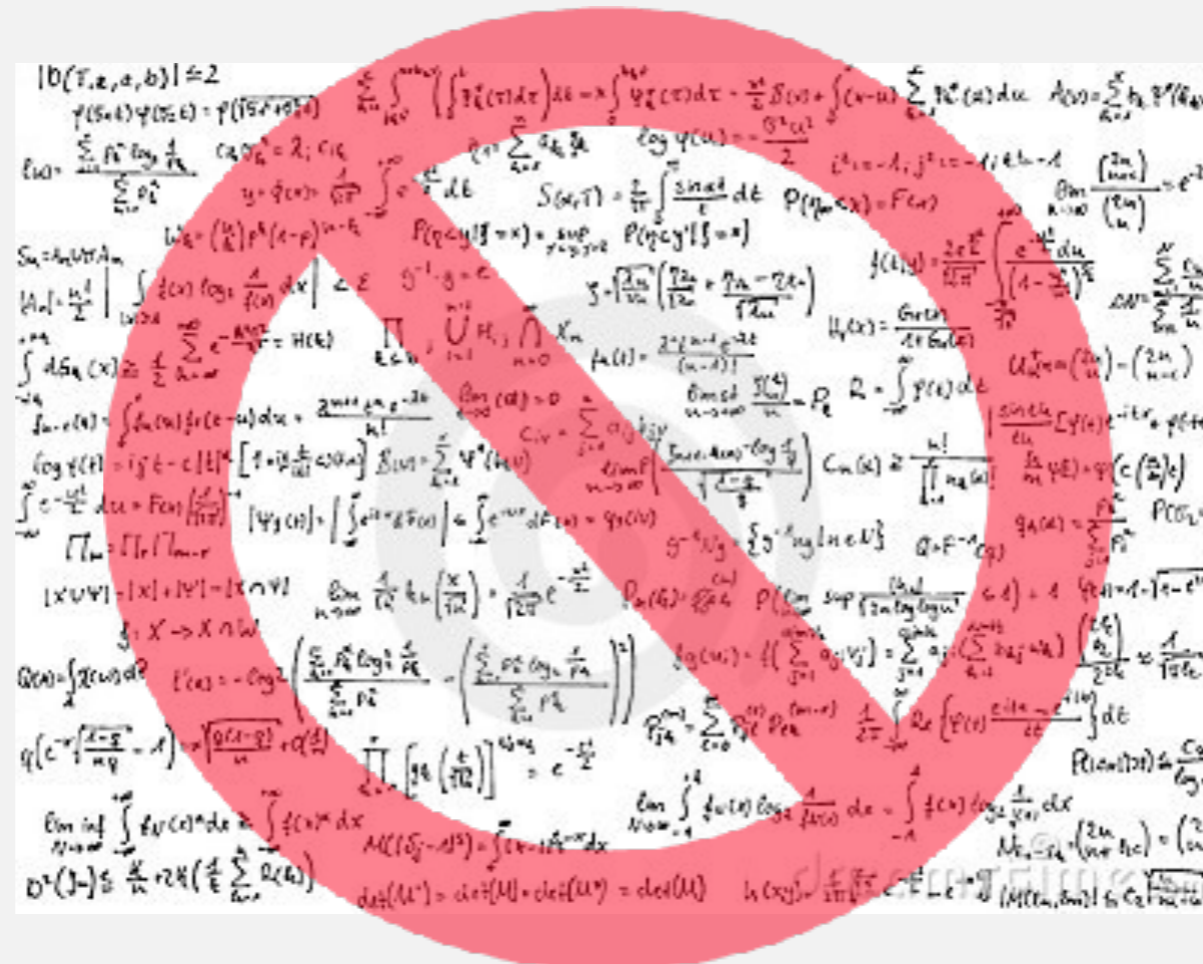
3body.txt

# 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.

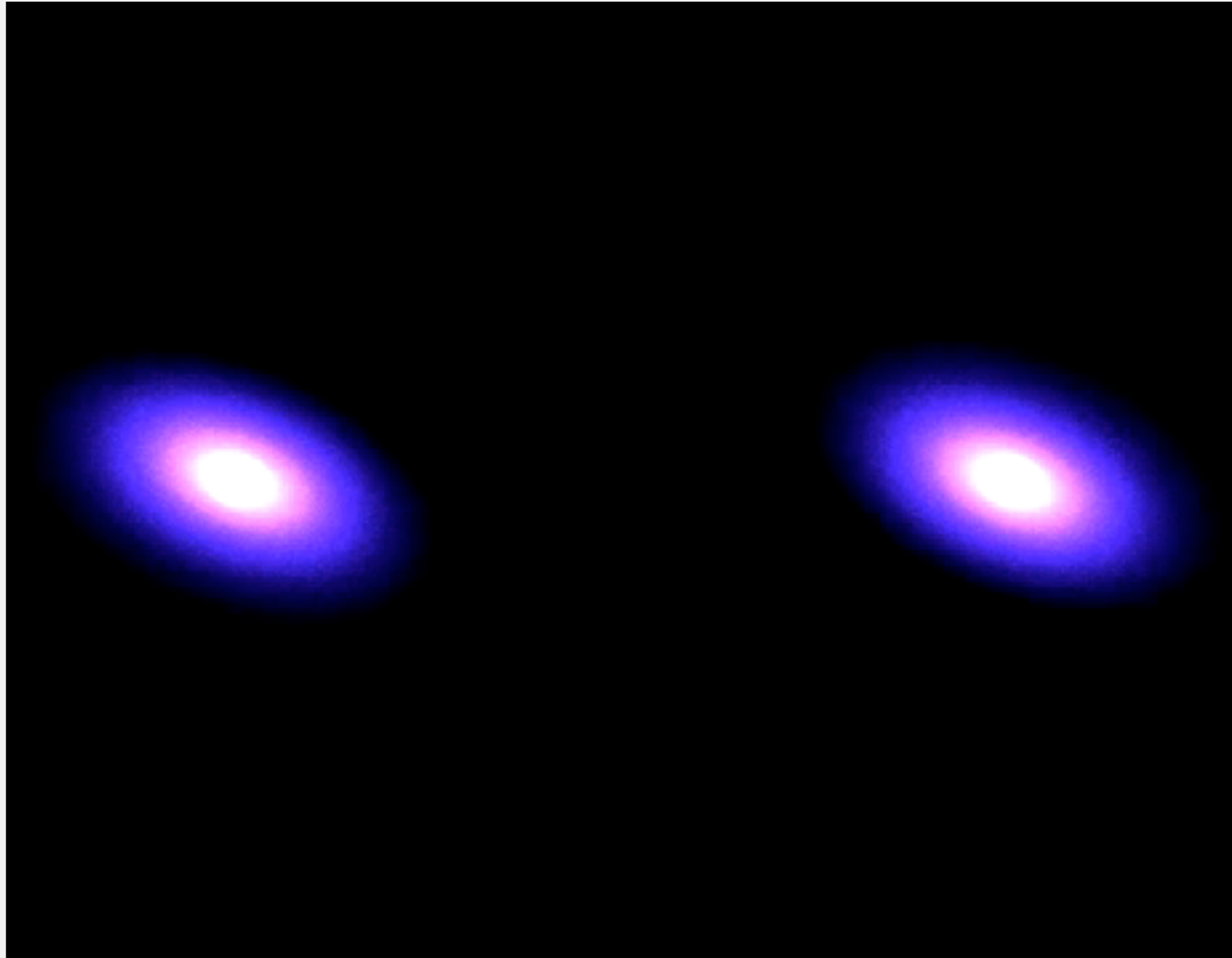


don't worry (this is not a math or physics course)

# Context

---

**Applications.** Cosmology, semiconductors, fluid dynamics, ....



[http://www.youtube.com/watch?v=ua7YIN4eL\\_w](http://www.youtube.com/watch?v=ua7YIN4eL_w)

# Programming goals

---

- Use standard input, standard output, and standard drawing for I/O.
- Use parallel arrays.
- Decompose a large program into small, manageable steps.

key to becoming  
a good programmer



# Before you begin

---

Carefully read assignment specification; skim checklist.

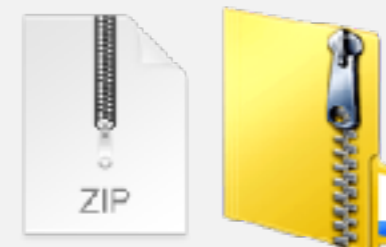
Check that standard libraries are available to Java.

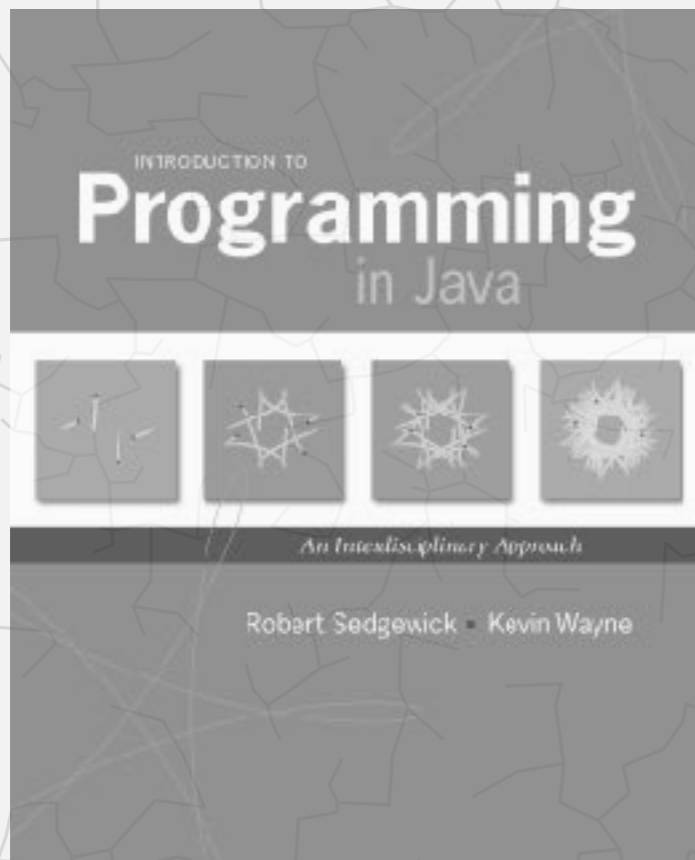
- Already configured if you used auto-installer.
- Remember to use `javac-introcs` and `java-introcs` at command line.
- To check installation, open command line and:
  - `% java-introcs StdIn`
  - `% java-introcs StdOut`
  - `% java-introcs StdDraw`
  - `% java-introcs StdAudio`

Useful programs from lecture/precept.

- `Students.java`
- `BouncingBallDeluxe.java`
- `Distinct.java`

Download sample data files and create working directory.





<http://introcs.cs.princeton.edu>

## ASSIGNMENT 2 TIPS AND

---

- ▶ *overview*
- ▶ *problem decomposition*
- ▶ *the physics*
- ▶ *bugs*
- ▶ *universes*



# Decompose problem into individual steps

---

Develop program incrementally, decomposing into 6 individual 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.

← physics localized to these steps  
(formulas provided)

**Advice.** Although final code will appear in order 1–6, we recommend implementing these steps in the order 1, 2, 6, 3, 4, 5B, 5C, 5A.

**Q.** Why?

**A.** Easier to test and debug.

# Start with comments

---

```
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 4. Play music on standard audio.  
  
        // 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 increment  $\Delta t$  from command line.
- Print values of each variable (as debugging aid).

**Note.** Easy, but you should still test it!

```
% java-introcs NBody 10 1
```

```
tau = 10.0
```

```
dt = 1.0
```

```
% java-introcs NBody 157788000.0 25000.0
```

```
tau = 1.57788E8
```

```
dt = 25000.0
```

# Standard input

---

Step 2. Read universe from standard input.

% more planets.txt

5 ← number of bodies n

2.50e+11 ← radius of universe

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

initial x- and y-position

initial x- and y-velocity

mass

image filename

data for  
one body

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

optional description

# Standard input

---

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

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

**Hint.** Recall Students.java.

```
% java-introcs NBody 157788000.0 25000.0 < planets.txt  
[no output]
```

**Q.** How to test?

**A.** Do Step 6 (print universe).

# Standard output

---

**Step 6.** Print universe to standard output.

- Write a loop to iterate over the six (6) parallel arrays.
- Use `StdOut.printf()` for formatted output (see checklist for hint).

```
% 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
```

# Standard drawing

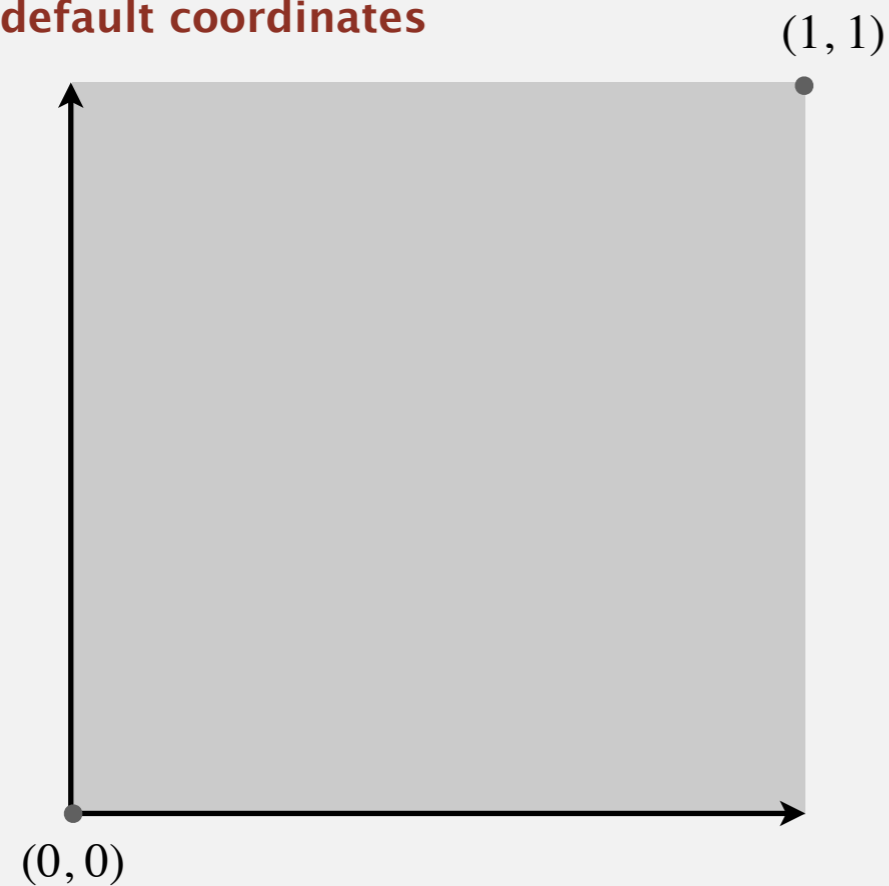
---

## Step 3. Initialize standard drawing.

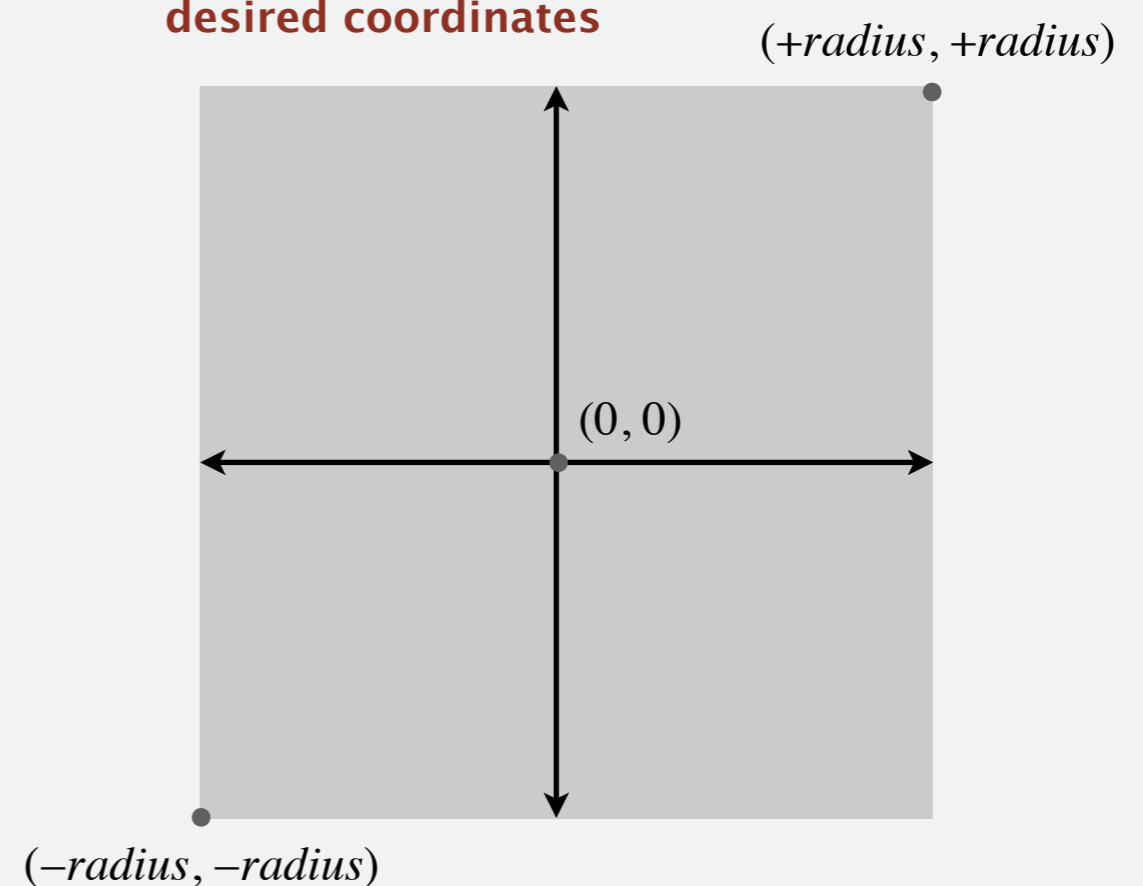
- Enable double buffering by calling `StdDraw.enableDoubleBuffering()`.
- Default  $x$ - and  $y$ -scale supports coordinates between 0 and 1;
- Change scale to be between  $-radius$  and  $+radius$ .

Hint: `StdDraw.setXscale()` and `StdDraw.setYscale()`.

default coordinates



desired coordinates



Q. How to test?

# Standard audio

---

## Step 4. Play music.

- Call `StdAudio.play("2001.wav")`.
- Easy (but **optional**).

**AUSO SPRACH ZARATHUSTRA!**

R. STRAUSS  
arr. by JASON LIN  
(LTS\_1996)

Sehr breit. ♩ = 69

Piano

*pp tremolo*

*P (eierlich)*

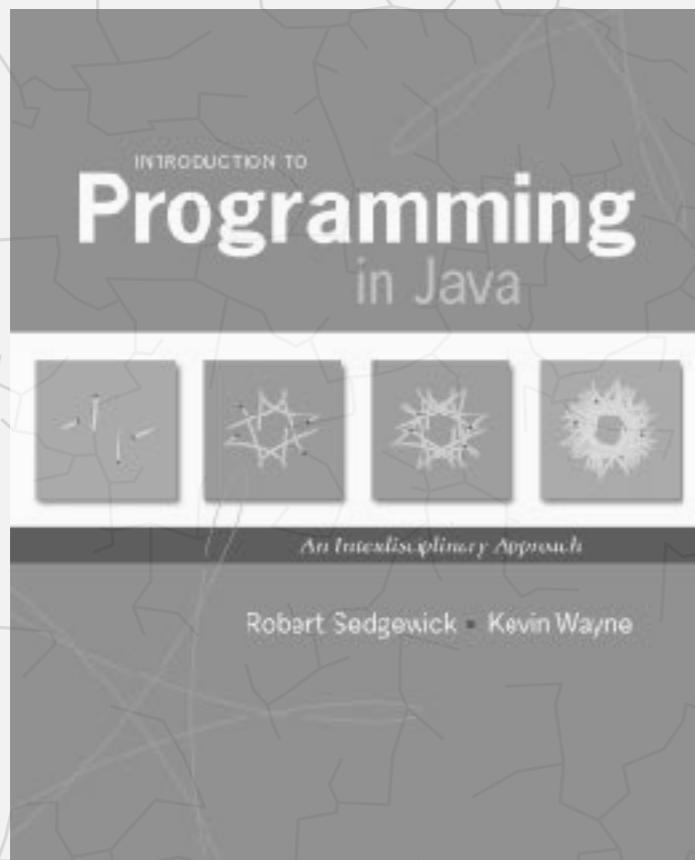
*f*

*p*

*sempre pp*







<http://introcs.cs.princeton.edu>

## ASSIGNMENT 2 TIPS AND

---

- ▶ *overview*
- ▶ *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.

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

**Hint.** See `BouncingBallDeluxe.java`.

# Measuring time

---

**Time loop.** From  $t = 0$  up to (but not including)  $T$ , incrementing by  $\Delta t$ .

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

```
% java-introcs NBody 23.0 2.5
```

```
% java-introcs NBody 25.0 2.5
```

**T = 23.0,  $\Delta t = 2.5$**

t = 0.0

t = 2.5

t = 5.0

t = 7.5

t = 10.0

t = 12.5

t = 15.0

t = 17.5

t = 20.0

t = 22.5

**T = 25.0,  $\Delta t = 2.5$**

t = 0.0

t = 2.5

t = 5.0

t = 7.5

t = 10.0

t = 12.5

t = 15.0

t = 17.5

t = 20.0

t = 22.5

don't include 25.0



# Updating the velocities and positions

---

**Step 5B.** [for now, 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:  $p_x = p_x + v_x \Delta t$ ,  $p_y = p_y + v_y \Delta t$ .

**Warning.** Cut-and-paste errors are common.

**Q.** How to test?

**A.** Artificial universe that is easy to check by hand.

```
% java-introcs NBody 192 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.

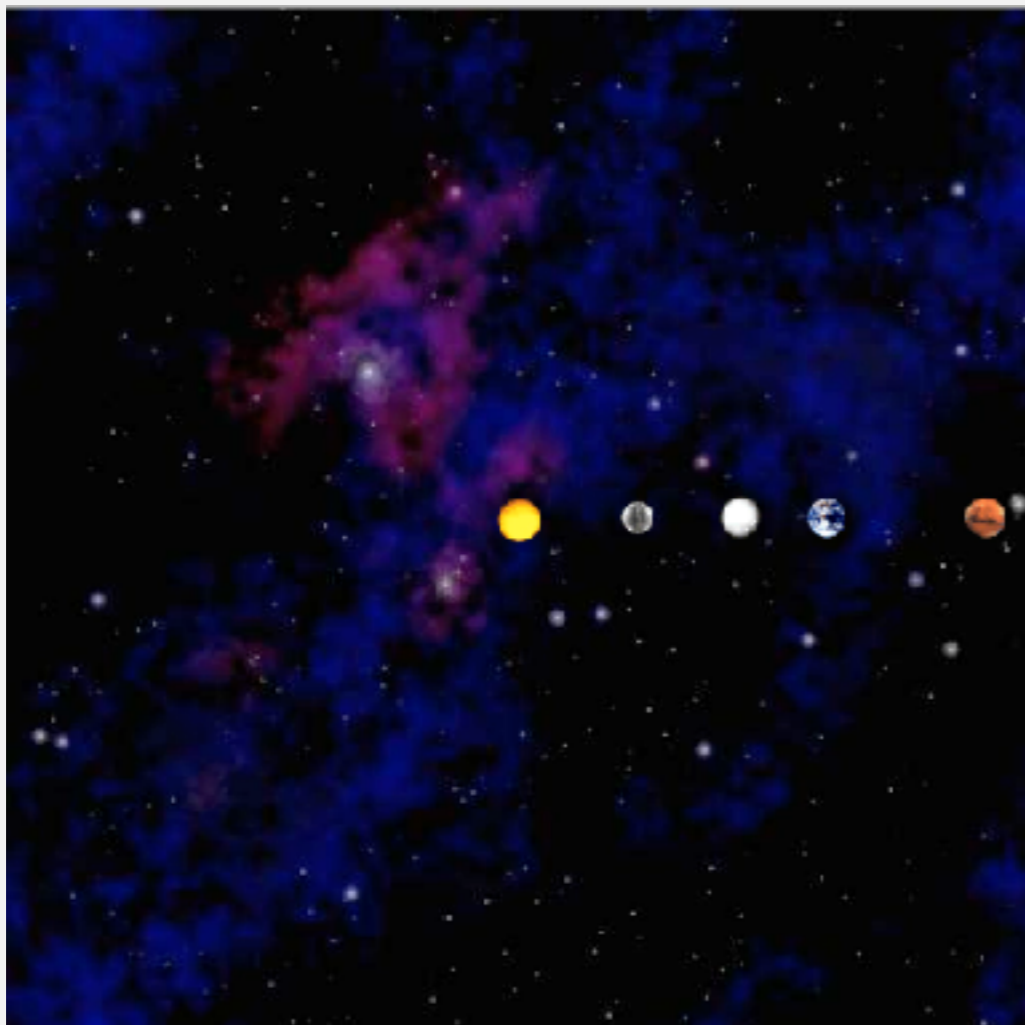
- Draw background image.
- Write loop to display  $n$  bodies.
- Call `StdDraw.show()` to display results on screen.
- Call `StdDraw.pause(20)` to control animation speed.

# Drawing the universe

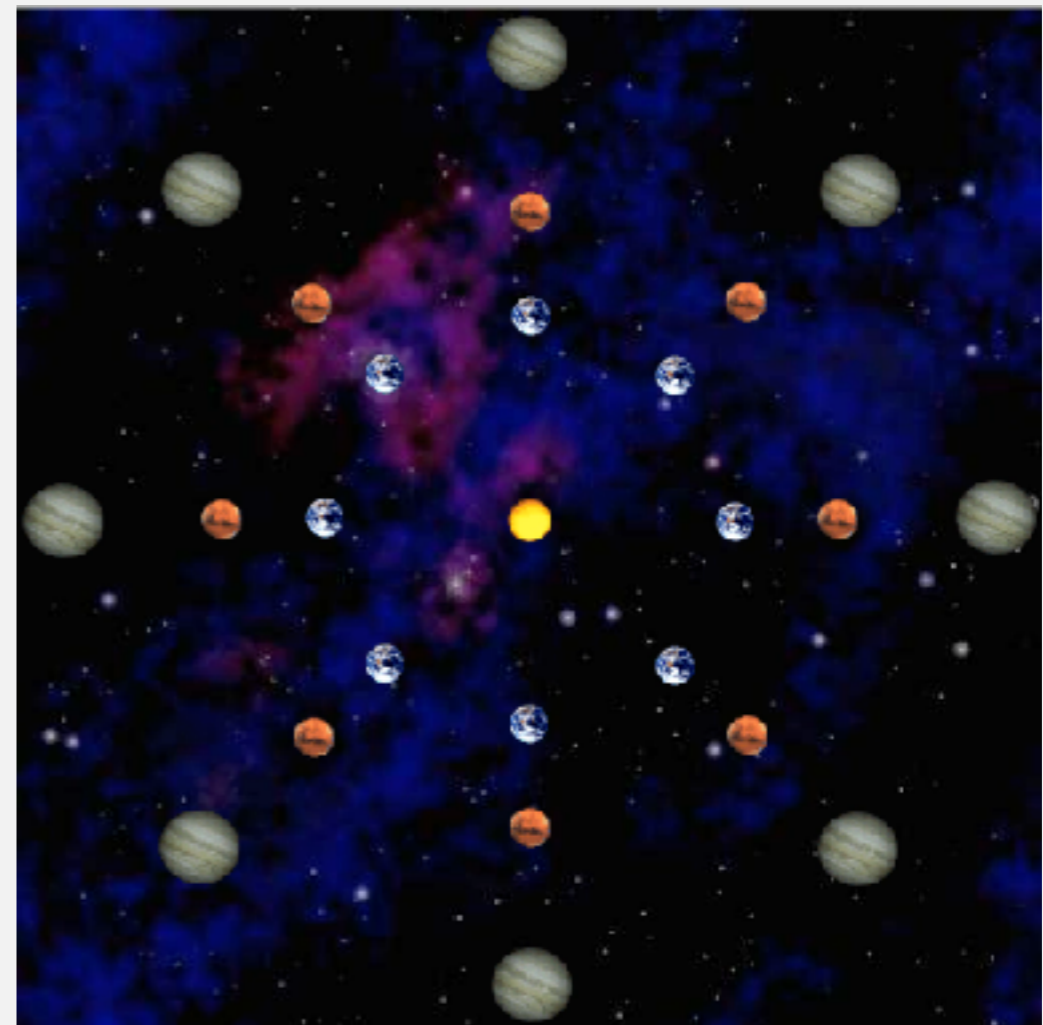
---

## 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

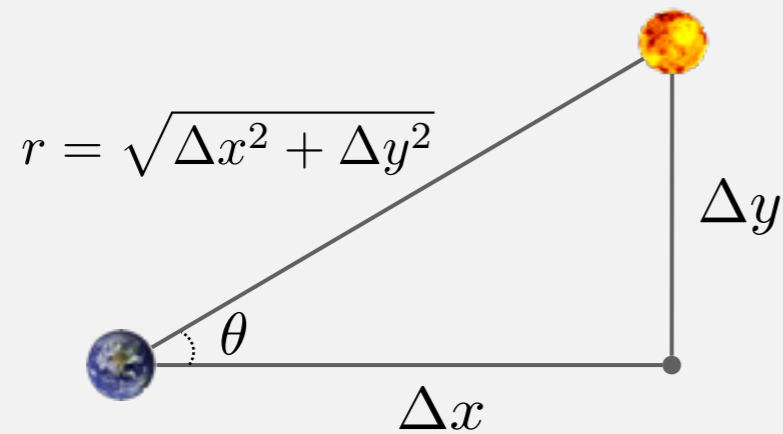
# Calculating the force (between two bodies at time t)

---

## Step 5A.

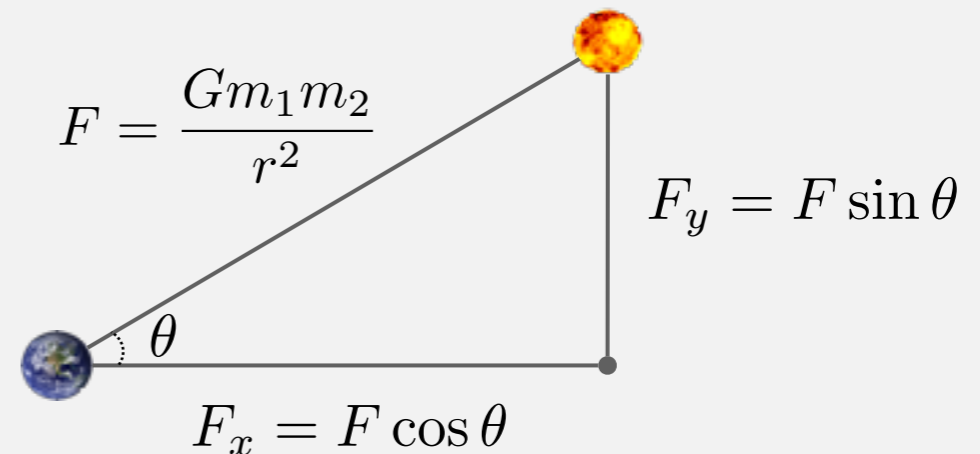
- Apply Newton's law of gravity.
- A bit of high-school trig (formulas provided).

distance between two bodies



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

force between two bodies



# Calculating the force (between all pairs of bodies at time t)

---

Principle of superposition. Add all pairwise forces.

$$\vec{F}_{earth} = \vec{F}_{mars \rightarrow earth} + \vec{F}_{mercury \rightarrow earth} + \vec{F}_{sun \rightarrow earth} + \vec{F}_{venus \rightarrow earth}$$

How to implement?

- Need two extra arrays `fx[]` and `fy[]`. Why?
- Need to examine all pairs of bodies, ala `Distinct.java`.

Warmup. Enumerate all pairs of bodies.

**n = 5**

0-1 0-2 0-3 0-4

1-0 1-2 1-3 1-4

2-0 2-1 2-3 2-4

3-0 3-1 3-2 3-4

4-0 4-1 4-2 4-3

**n = 4**

0-1 0-2 0-3

1-0 1-2 1-3

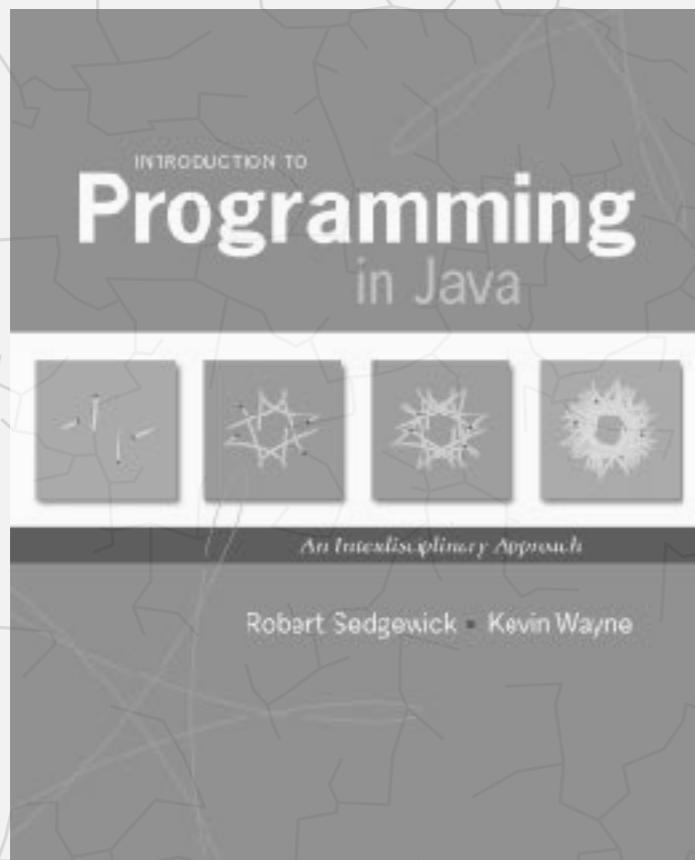
2-0 2-1 2-3

3-0 3-1 3-2



don't include 0-0, 1-1, 2-2, or 3-3





<http://introcs.cs.princeton.edu>

## ASSIGNMENT 2 TIPS AND


---

- ▶ *overview*
- ▶ *problem decomposition*
- ▶ *the physics*
- ▶ *bugs*
- ▶ *universes*

# Advice

---

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



key to becoming  
a good programmer

# Command-line bug

---

```
% java-introcs NBody 157788000.0 25000.0 > planets.txt
```

```
<Ctrl-C>
```

```
% 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)
```

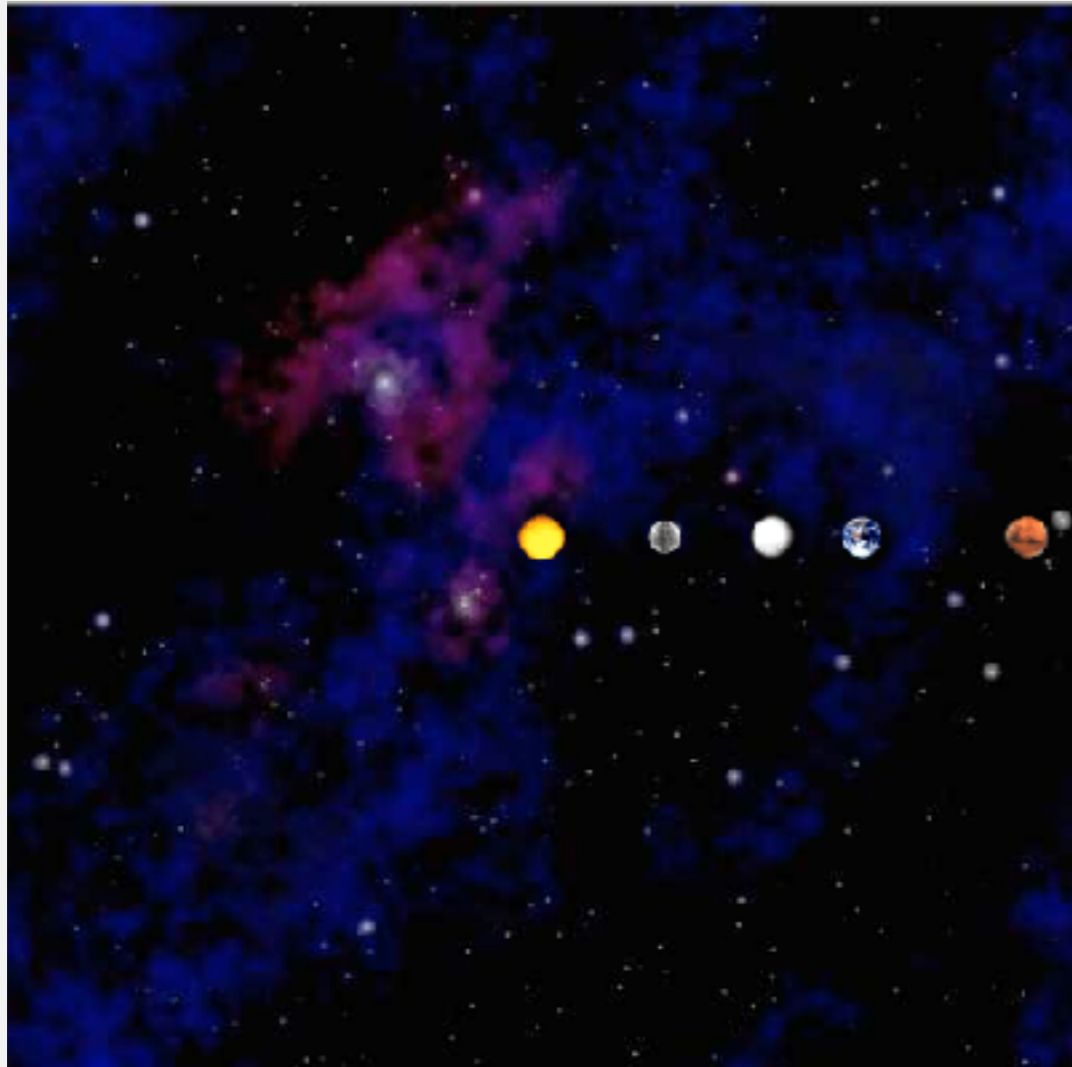
```
    at NBody.main(NBody.java:54)
```

```
% more planets.txt
```

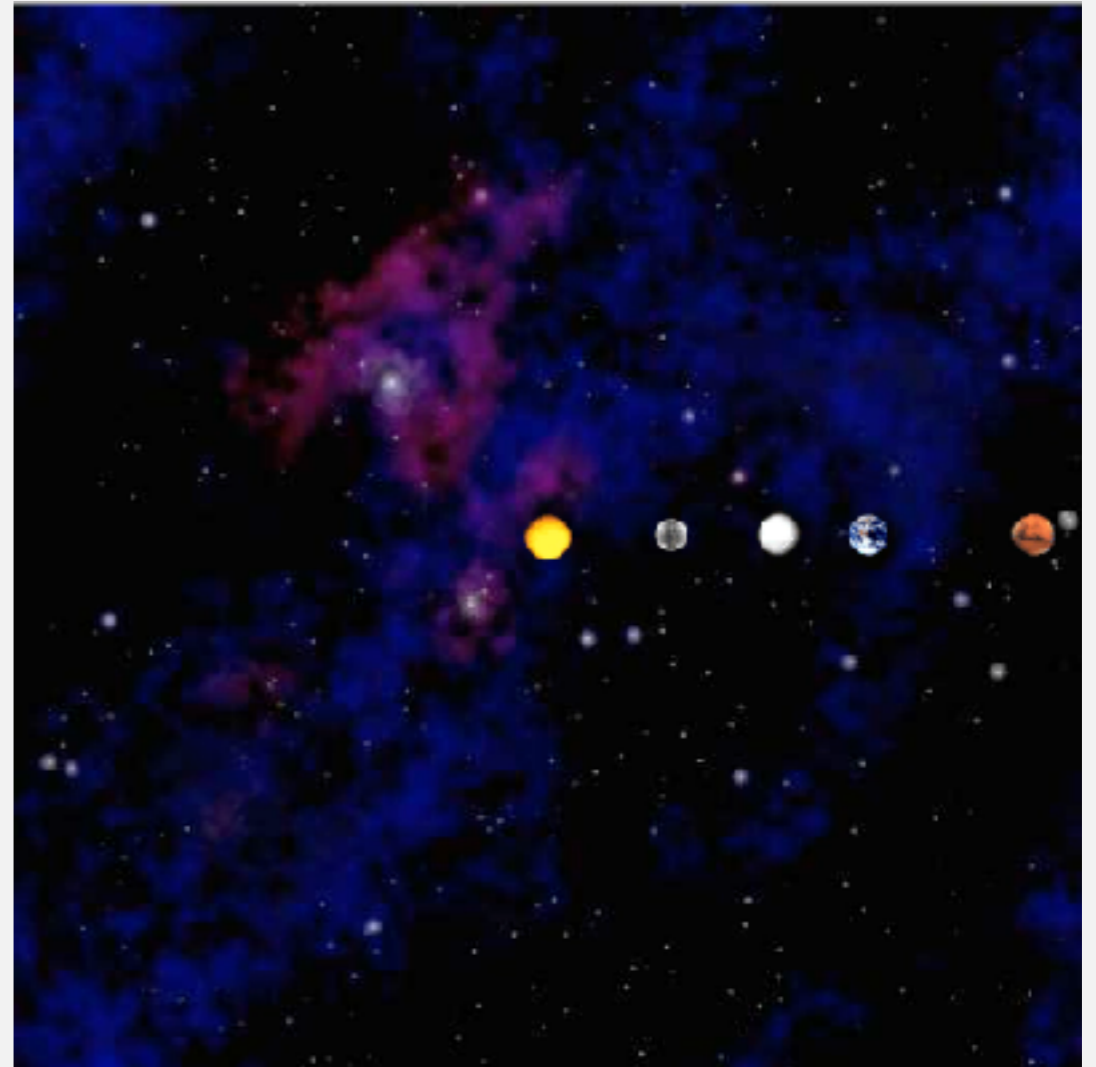
```
[it's empty - you erased it!]
```

# Visual bugs

---



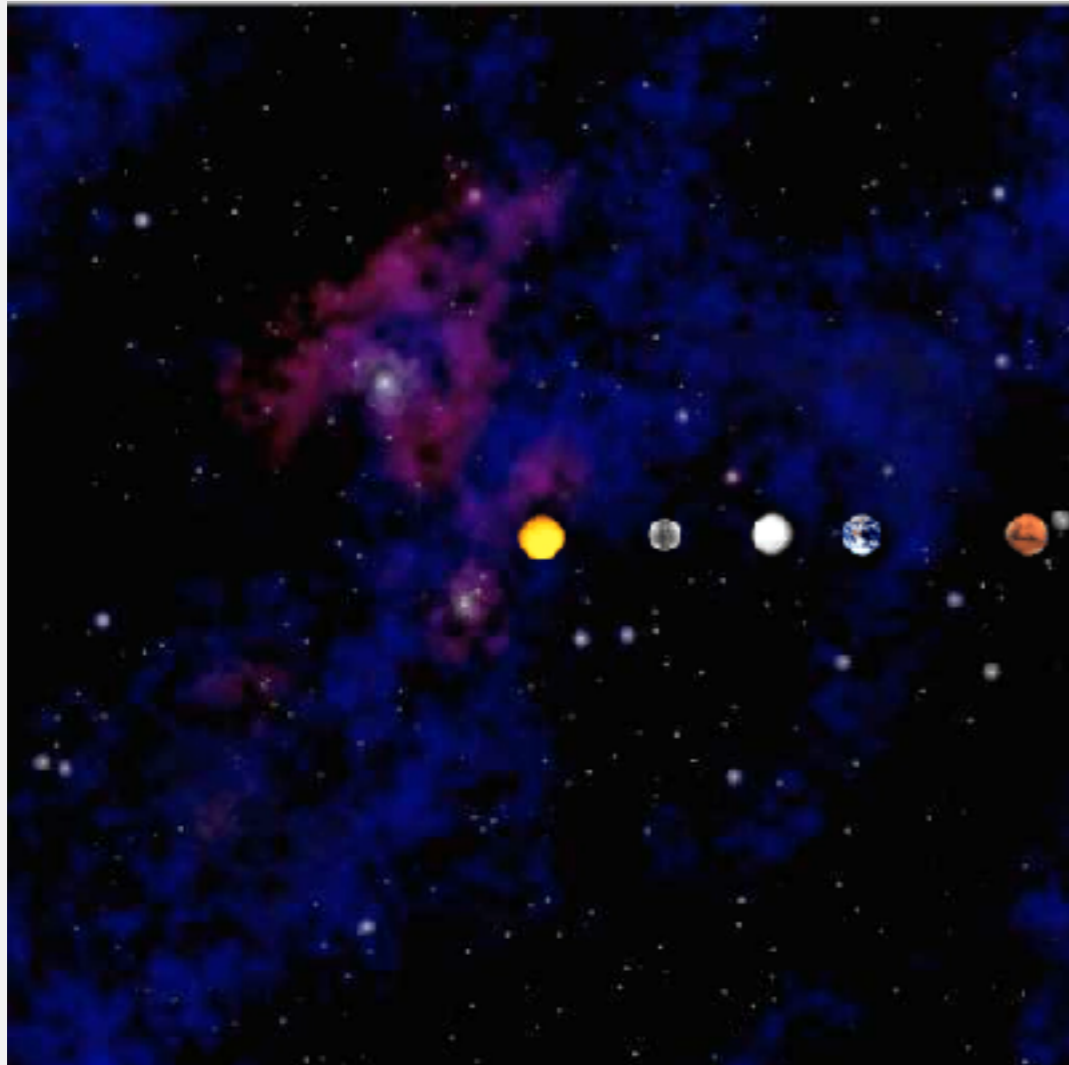
no motion



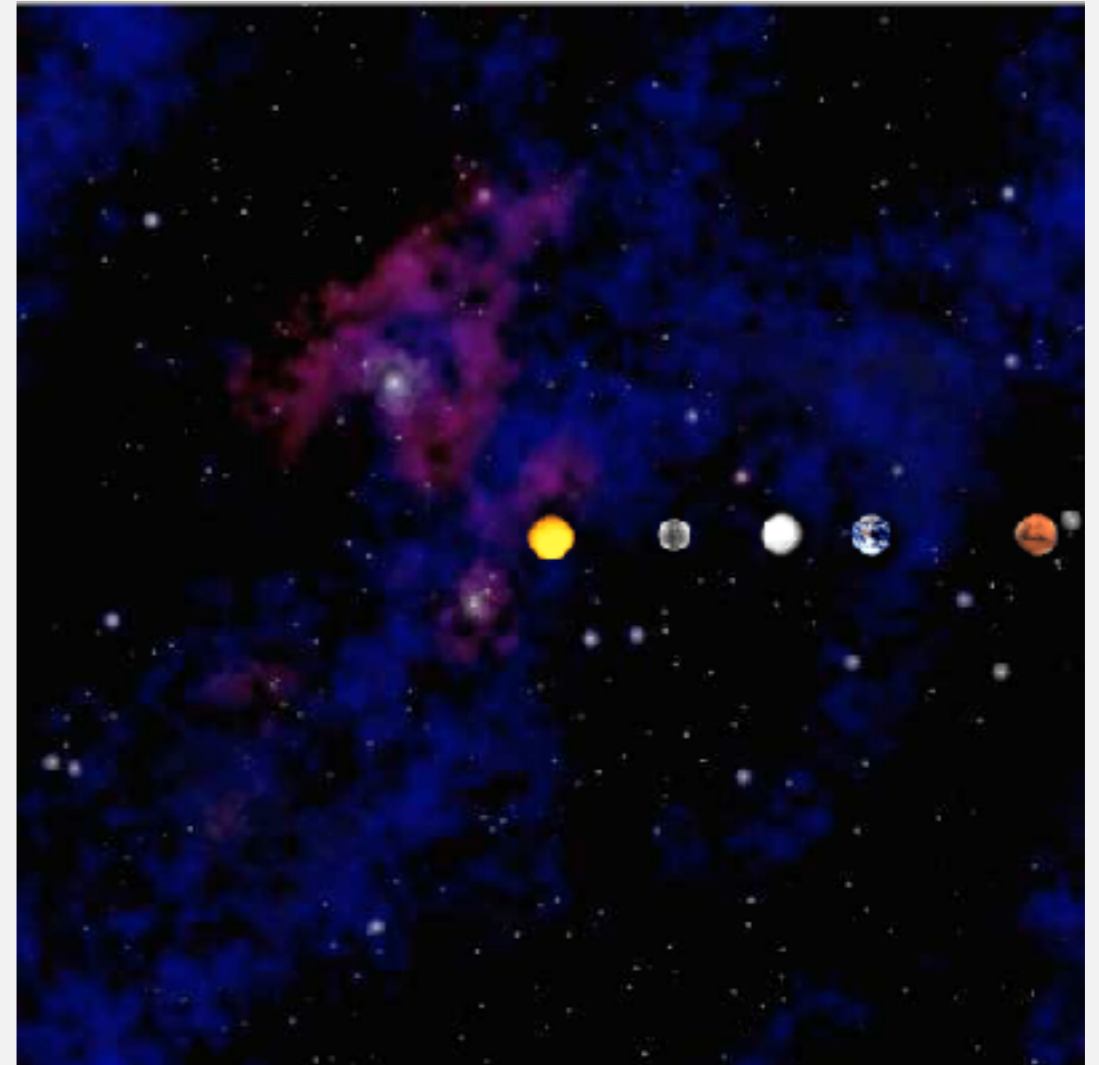
no gravity

# Visual bugs

---



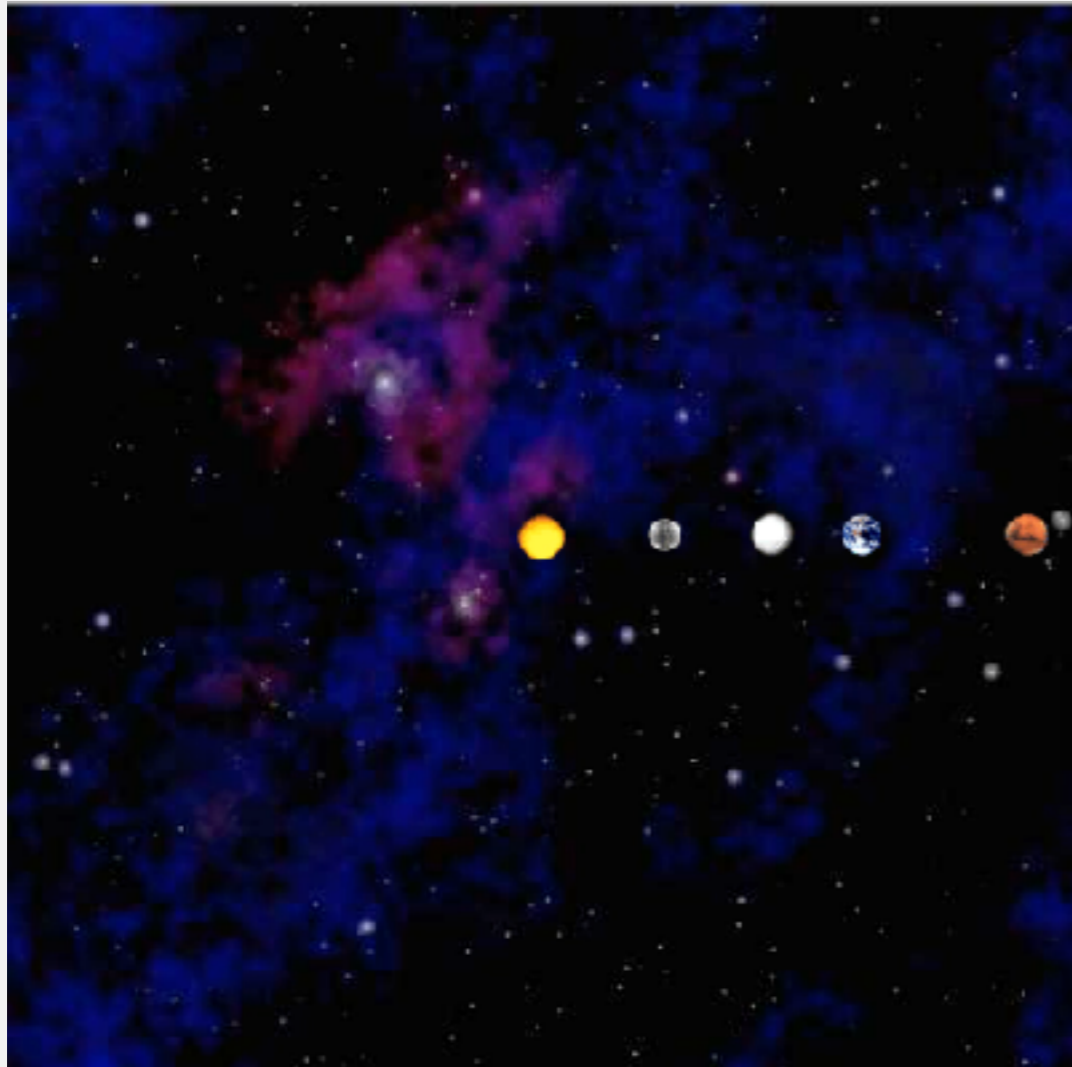
no double buffering



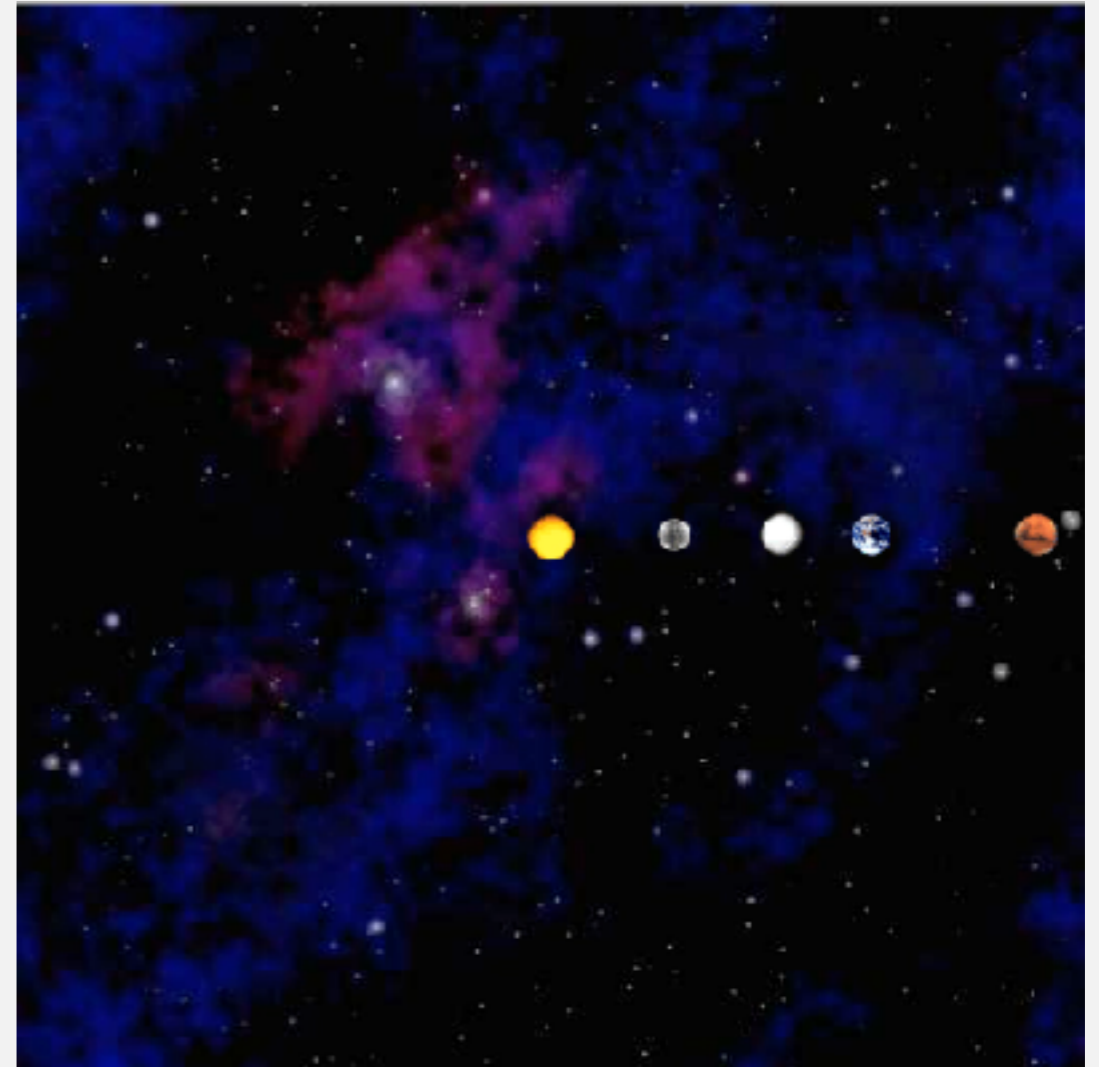
planets repel one another

# Visual bugs

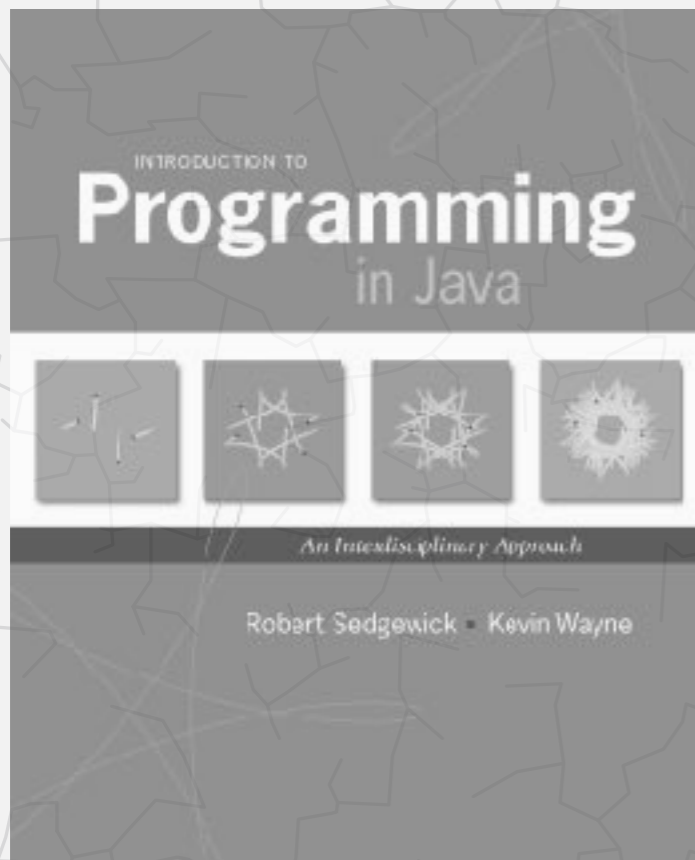
---



wrong force loop



cut-and-paste error (x vs. y)



<http://introcs.cs.princeton.edu>

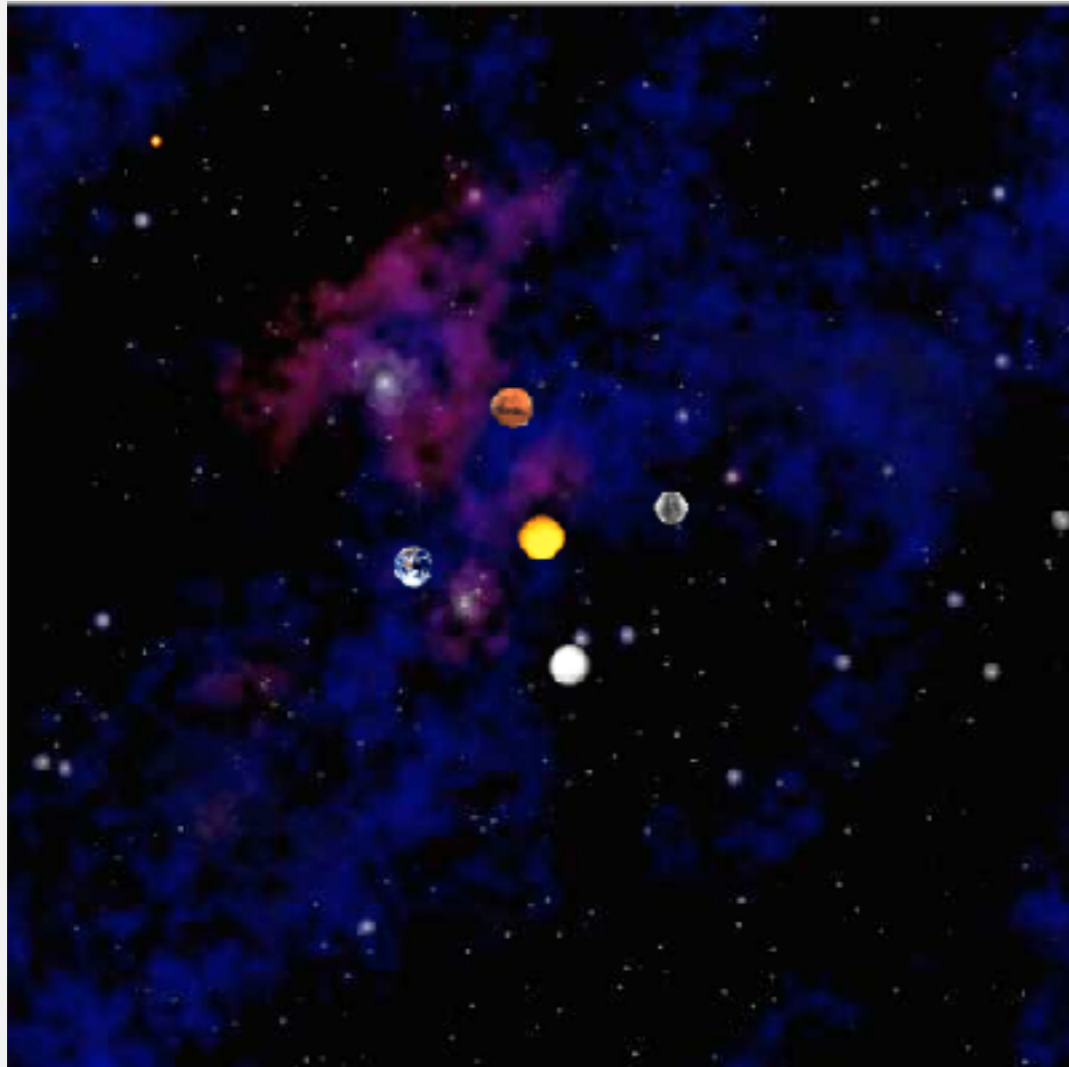
## ASSIGNMENT 2 TIPS AND

---

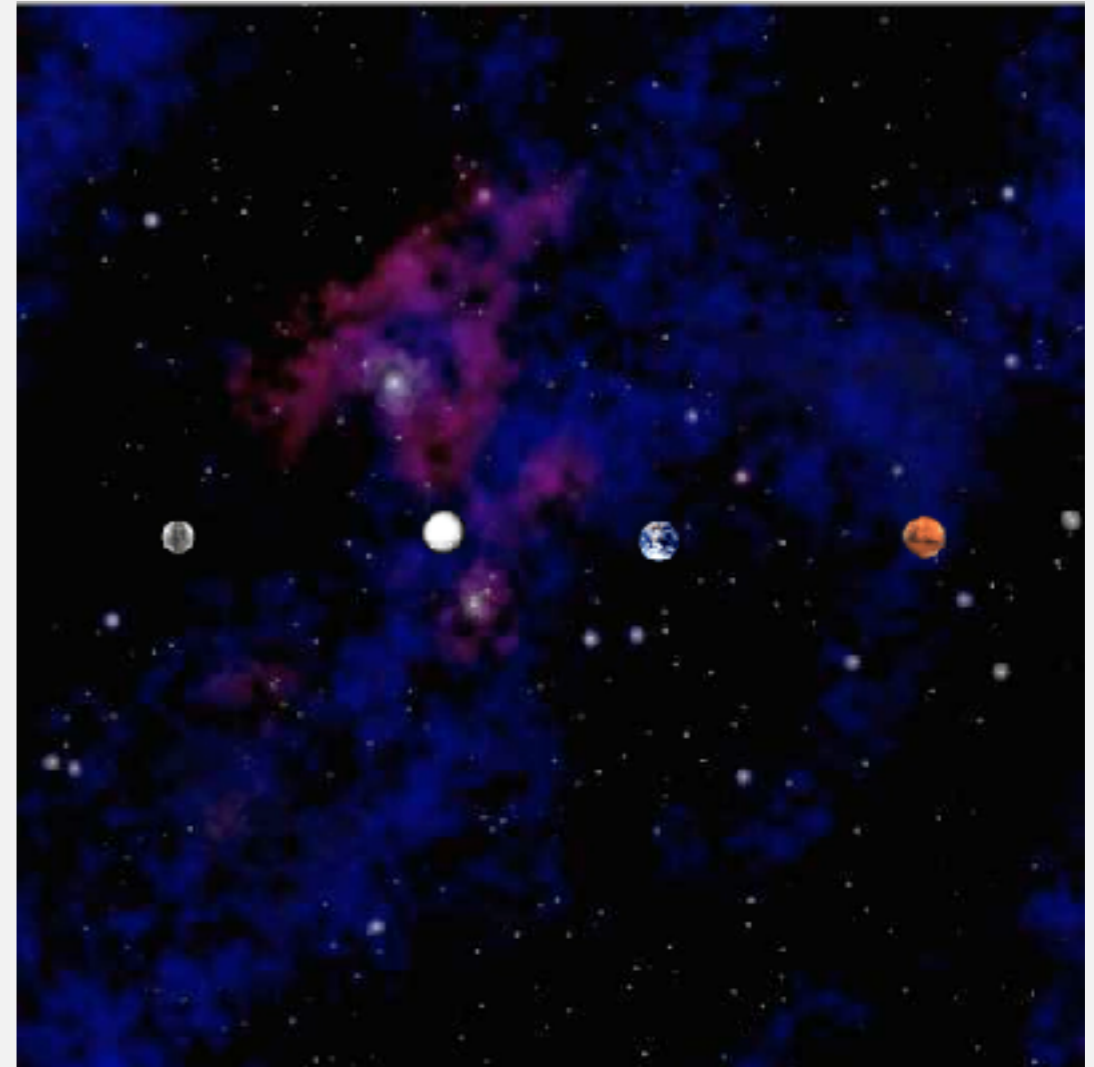
- ▶ *overview*
- ▶ *problem decomposition*
- ▶ *the physics*
- ▶ *bugs*
- ▶ *universes*

# Other universes

---



**planetsparty.txt**  
(created by Mary Fan)

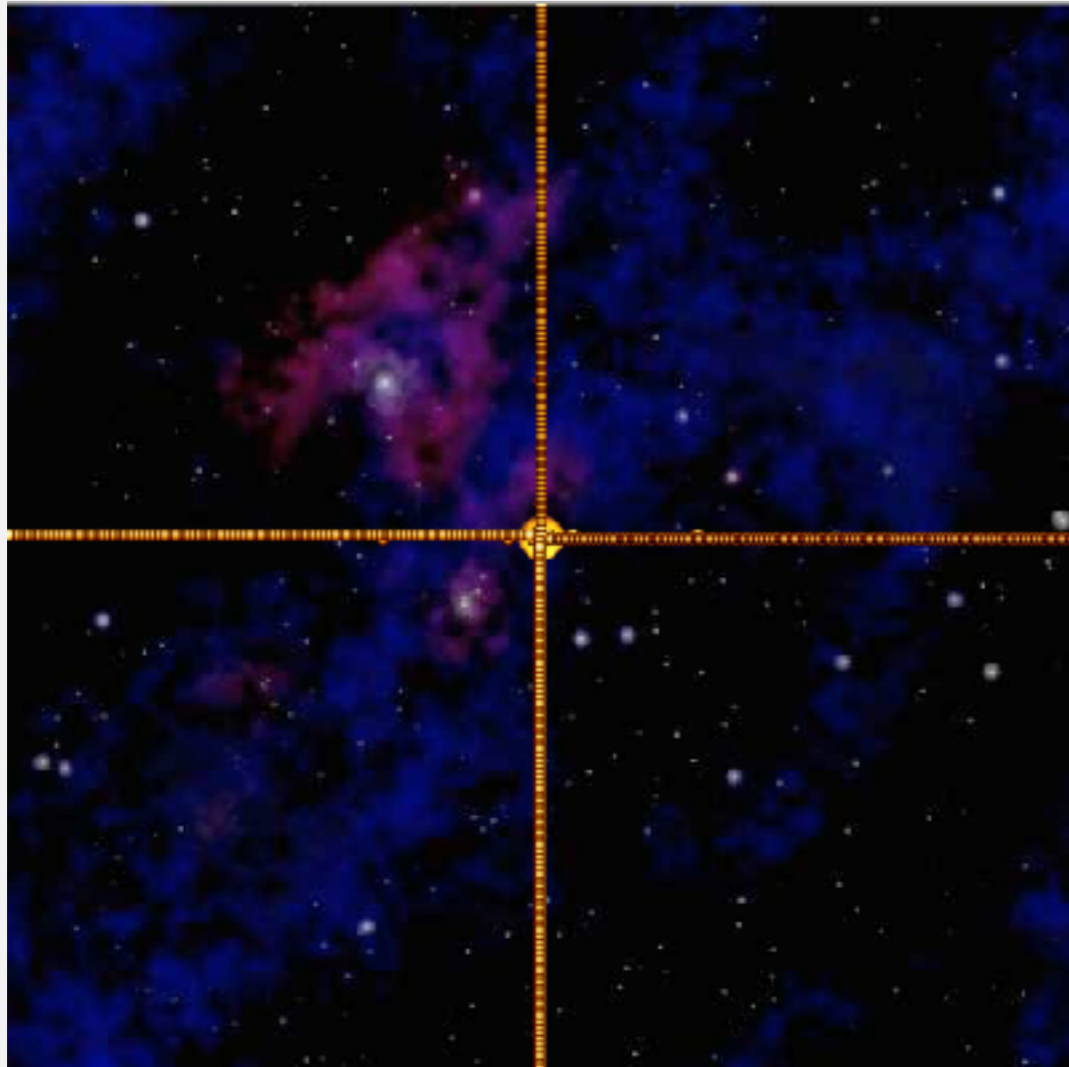


**twinbinaries.txt**  
(David Costanzo)

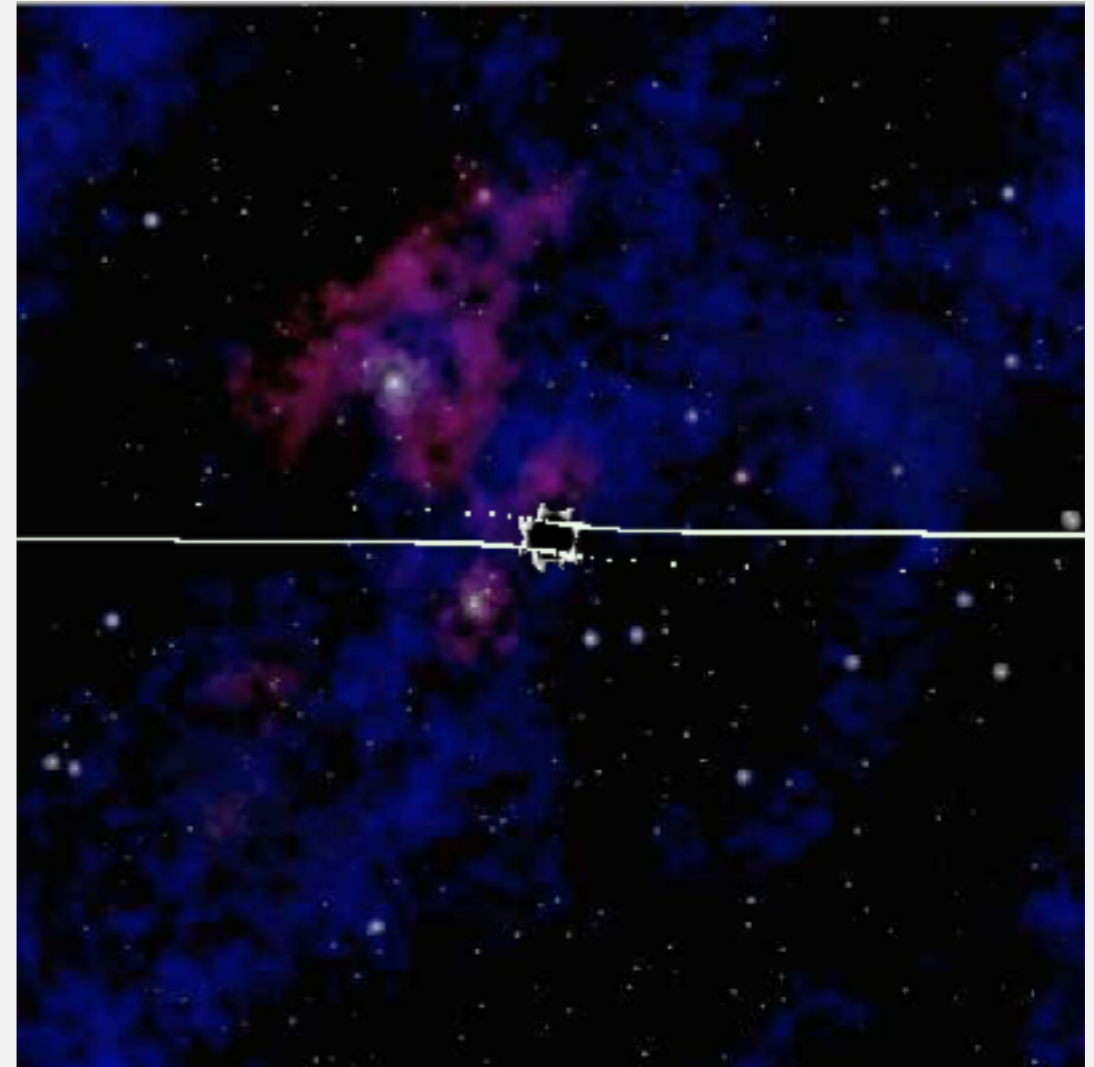


# Other universes

---



**chaosblossum.txt**  
(created by Erik Keselica)



**galaxy.txt**  
(created by Matt Tilghman)