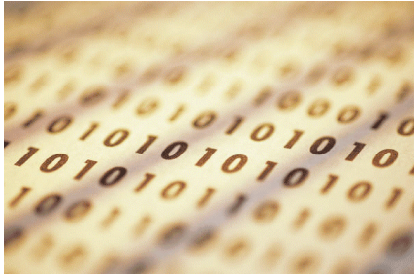


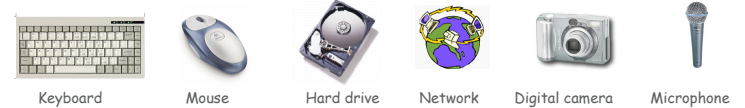
## 1.5 Input and Output



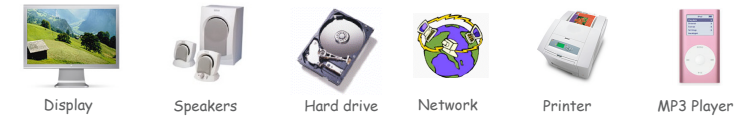
**Today's goal.** Java programs that interact with outside world.

## Input and Output

### Input devices.



### Output devices.



### Our approach.

- Define Java libraries of functions for input and output.
- Use operating system (OS) to connect Java programs to: file system, each other, keyboard, mouse, display, speakers.

## Terminal

**Terminal.** Application where you can type commands to control the operating system.

```
Terminal - tcsh - 65x12
[wayne@bicycle] ~/intros: javac RandomSeq.java
[wayne@bicycle] ~/intros: java RandomSeq 4
0.35603714028287214
0.9969546788376992
0.16163508427043993
0.8792203644361208
[wayne@bicycle] ~/intros: █
```

Mac OS X

```
C:\WINNT\System32\cmd.exe
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.
C:\>cd intros
C:\intros>cd hello
C:\intros\hello>javac HelloWorld.java
C:\intros\hello>java HelloWorld
Hello, World
C:\intros\hello>_
```

Microsoft Windows

## Command-Line Input and Standard Output

**Command-line input.** Read an integer  $N$  as command-line argument.

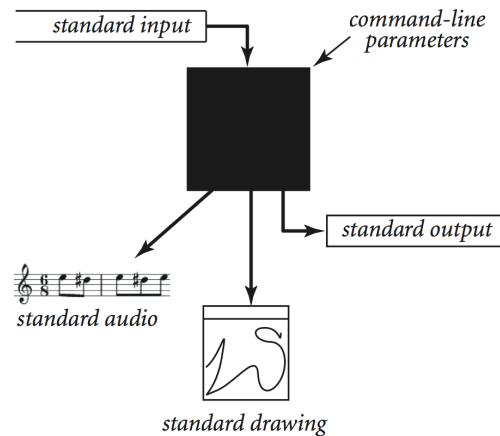
### Standard output.

- Flexible OS abstraction for output.
- In Java, output from `System.out.println()` goes to `stdout`.
- By default, `stdout` is sent to Terminal.

```
public class RandomSeq {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 0; i < N; i++)
            System.out.println(Math.random());
    }
}
```

```
% java RandomSeq 4
0.9320744627218469
0.4279508713950715
0.08994615071160994
0.6579792663546435
```

## New Bird's Eye View



7

## Standard Input

8

### Command-line Input vs. Standard Input

#### Command line inputs.

- Use command line inputs to read in a **few** user values.
- Not practical for many user inputs.
- Input entered **before** program begins execution.

#### Standard input.

- Flexible OS abstraction for input.
- By default, `stdin` is received from Terminal window.
- Input entered **while** program is executing.

### Standard Input

**Standard input.** We provide library `StdIn` to read text input.

**Standard output.** We provide library `StdOut` to write text output.

**To use.** Download `StdIn.java` and `StdOut.java` from oopsite and put in working directory (or use classpath).

```
public class Add {  
    public static void main(String[] args) {  
        StdOut.print("Type the first integer: ");  
        int x = StdIn.readInt();  
        StdOut.print("Type the second integer: ");  
        int y = StdIn.readInt();  
        int sum = x + y;  
        StdOut.out.println("Their sum is " + sum);  
    }  
}
```

```
% java Add  
Type the first integer: 1  
Type the second integer: 2  
Their sum is 3
```

9

10

## Twenty Questions

**Twenty questions.** User thinks of an integer between one and 1 million. Computer tries to guess it.

```
public class TwentyQuestions {
    public static void main(String[] args) {
        int lo = 1, hi = 1000000;
        while (lo < hi) {
            int mid = (lo + hi) / 2;
            StdOut.println("Is your number <= " + mid + "?");
            boolean response = StdIn.readBoolean();
            if (response) hi = mid;
            else lo = mid + 1;
        }
        StdOut.println("Your number is " + lo);
    }
}
```

**Binary search.** Each question removes half of possible remaining values.

**Consequence.** Always succeeds after 20 questions.

$2^{20} \approx 1$  million

invariant: user's number  
always between lo and hi

11

## Redirection and Piping

## Averaging A Stream of Numbers

**Average.** Read in real numbers, and print their average.

```
public class Average {
    public static void main(String[] args) {
        double sum = 0.0;
        int N = 0;
        while (!StdIn.isEmpty()) {
            double x = StdIn.readDouble();
            sum = sum + x;
            N++;
        }
        StdOut.println(sum / N);
    }
}
```

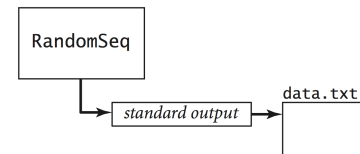
```
% java Average
10.0 5.0 6.0
3.0 7.0 32.0
<Ctrl-d>
10.5
```

<Ctrl-d> is OS X/Linux/Unix EOF  
<Ctrl-z> is Windows analog  
currently no DrJava analog

12

## Redirecting Standard Output

**Redirecting standard output.** Use OS directive to send standard output to a file for permanent storage (instead of terminal window).



```
% java RandomSeq 1000 > data.txt
```

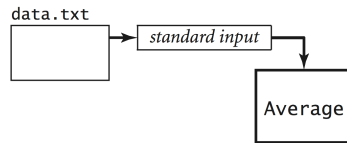
redirect stdout

13

14

## Redirecting Standard Input

**Redirecting standard input.** Use OS directive to read standard input from a file (instead of terminal window).



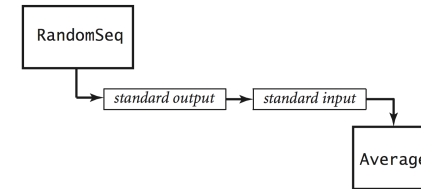
```
% more < data.txt
0.5475375782884312
0.4971087292684019
0.23123808041753813
...
% java Average < data.txt
0.4947655567740991
```

↙ redirect stdin

15

## Connecting Programs

**Piping.** Use OS directive to make the standard output of one program become the standard input of another.



```
% java RandomSeq 1000000 | java Average
0.4997970473016028
```

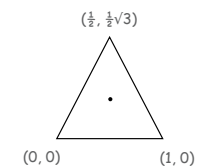
16

## Standard Drawing

**Standard drawing.** We provide library `StdDraw` to plot graphics. To use, download `StdDraw.java` and put in working directory.

```
public class Triangle {
    public static void main(String[] args) {
        double t = Math.sqrt(3.0) / 2.0;
        StdDraw.line(0.0, 0.0, 1.0, 0.0);
        StdDraw.line(1.0, 0.0, 0.5, t);
        StdDraw.line(0.5, t, 0.0, 0.0);
        StdDraw.point(0.5, t/3.0);
    }
}
```

```
% java Triangle
```



17

18

## Data Visualization

**Plot filter.** Read in a sequence of (x, y) coordinates from standard input, and plot using standard drawing.

```
public class PlotFilter {
    public static void main(String[] args) {
        double xmin = StdIn.readDouble();
        double ymin = StdIn.readDouble();
        double xmax = StdIn.readDouble();
        double ymax = StdIn.readDouble();
        StdDraw.setXscale(xmin, xmax);
        StdDraw.setYscale(ymin, ymax);

        while (!StdIn.isEmpty()) {
            double x = StdIn.readDouble();
            double y = StdIn.readDouble();
            StdDraw.point(x, y);
        }
    }
}
```

rescale coordinate system

read in points, and plot them

19

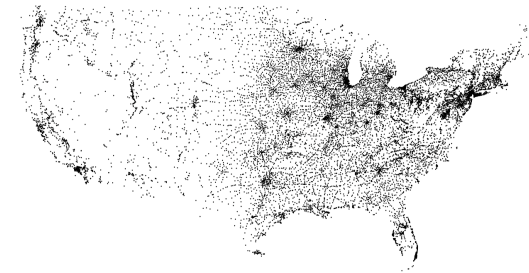
## Data Visualization

```
more < USA.txt
669905.0 247205.0 1244962.0 490000.0
1097038.8890 245552.7780
1103961.1110 247133.3330
1104677.7780 247205.5560
...

java PlotFilter < USA.txt
```

bounding box

coordinates of 13,509 US cities



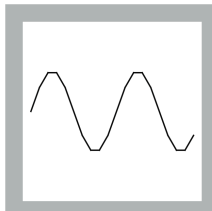
20

## Plotting a Function

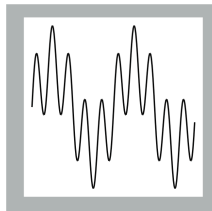
```
double[] a = new double[N+1];
for (int i = 0; i <= N; i++)
    a[i] = Math.sin(4*Math.PI*i/N) + Math.sin(20*Math.PI*i/N);

StdDraw.setXscale(0, N);
StdDraw.setYscale(-2.0, +2.0);
for (int i = 0; i < N; i++)
    StdDraw.line(i, a[i], i+1, a[i+1]);
```

N = 20



N = 200



$$y = \sin 4x + \sin 20x$$

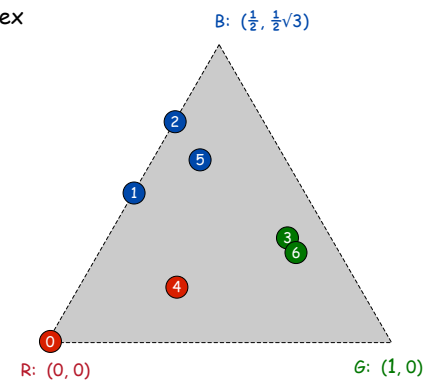
21

## Chaos Game

**Chaos game.** Play on equilateral triangle, with vertices R, G, B.

- Start at R.
- Repeat the following N times:
  - pick a random vertex
  - move halfway between current point and vertex
  - draw a point in color of vertex

Q. What picture emerges?



22

## Chaos Game

```

public class Chaos {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        double[] cx = { 0.000, 1.000, 0.500 };
        double[] cy = { 0.000, 0.000, 0.866 };

        double x = 0.0, y = 0.0;
        for (int i = 0; i < N; i++) {
            int r = (int) (Math.random() * 3);
            x = (x + cx[r]) / 2.0;
            y = (y + cy[r]) / 2.0;
            StdDraw.point(x, y);
        }
    }
}

```

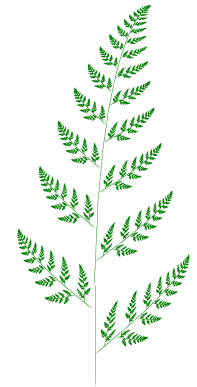
$\frac{1}{2}\sqrt{3}$   
(avoid hardwired constants like this)

between 0 and 2

## Barnsley Fern

Barnsley fern. Play chaos game with different rules.

probability	new x	new y
2%	.50	.27y
15%	$-.14x + .26y + .57$	$.25x + .22y - .04$
13%	$.17x - .21y + .41$	$.22x + .18y + .09$
70%	$.78x + .03y + .11$	$-.03x + .74y + .27$



- Q. What does computation tell us about nature?
- Q. What does nature tell us about computation?

20<sup>th</sup> century sciences. Formulas.

21<sup>st</sup> century sciences. Algorithms?

23

26

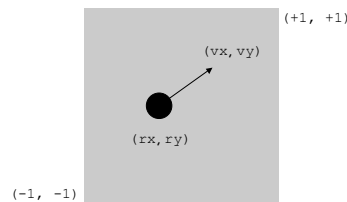
## Animation

Animation loop. Repeat the following:

- Clear the screen.
- Move the object.
- Draw the object.
- Display and pause for a short while.

Ex. Bouncing ball.

- Ball has position  $(rx, ry)$  and constant velocity  $(vx, vy)$ .
- Detect collision with wall and reverse velocity.



27

## Bouncing Ball

```

public class BouncingBall {
    public static void main(String[] args) {
        double rx = .480, ry = .860;
        double vx = .015, vy = .023;
        double radius = .05;

        StdDraw.setXscale(-1.0, +1.0);
        StdDraw.setYscale(-1.0, +1.0);

        while(true) {
            if (Math.abs(rx + vx) > 1.0) vx = -vx;
            if (Math.abs(ry + vy) > 1.0) vy = -vy;

            rx = rx + vx;
            ry = ry + vy;

            StdDraw.clear(StdDraw.GRAY);
            StdDraw.setPenColor(StdDraw.BLACK);
            StdDraw.filledCircle(rx, ry, radius);
            StdDraw.show(50);
        }
    }
}

```

position  
constant velocity  
radius

rescale coordinates

bounce

update position

clear background  
draw the ball

turn on animation mode:  
display and pause for 50ms

28

## Special Effects

**Images.** Put `.gif`, `.png`, or `.jpg` file in the working directory and use `StdDraw.picture()` to draw it.

**Sound effects.** Put `.wav`, `.mid`, or `.au` file in the working directory and use `StdAudio.play()` to play it.

**Ex.** Modify `BouncingBall` to display image and play sound upon collision.

- Replace `StdDraw.filledCircle()` with:

```
StdDraw.picture(rx, ry, "earth.gif");
```

- Add following code when collision with wall is detected:

```
StdAudio.play("boing.wav");
```