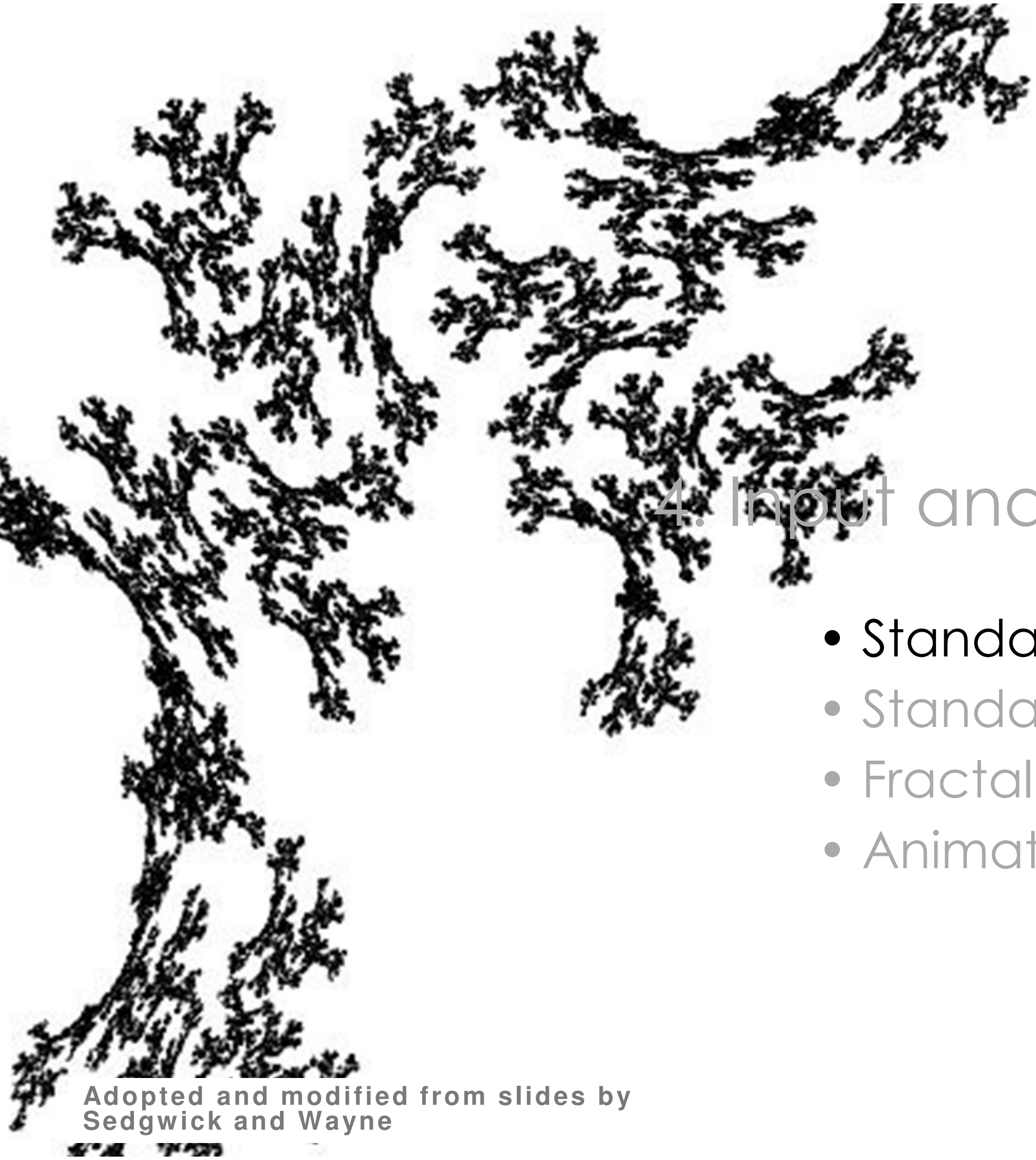
INTRODUCTION TO
COMPUTER SCIENCE
Rutgers University

6. Input / Output

http://introcs.cs.rutgers.edu



INTRODUCTION TO COMPUTER SCIENCE

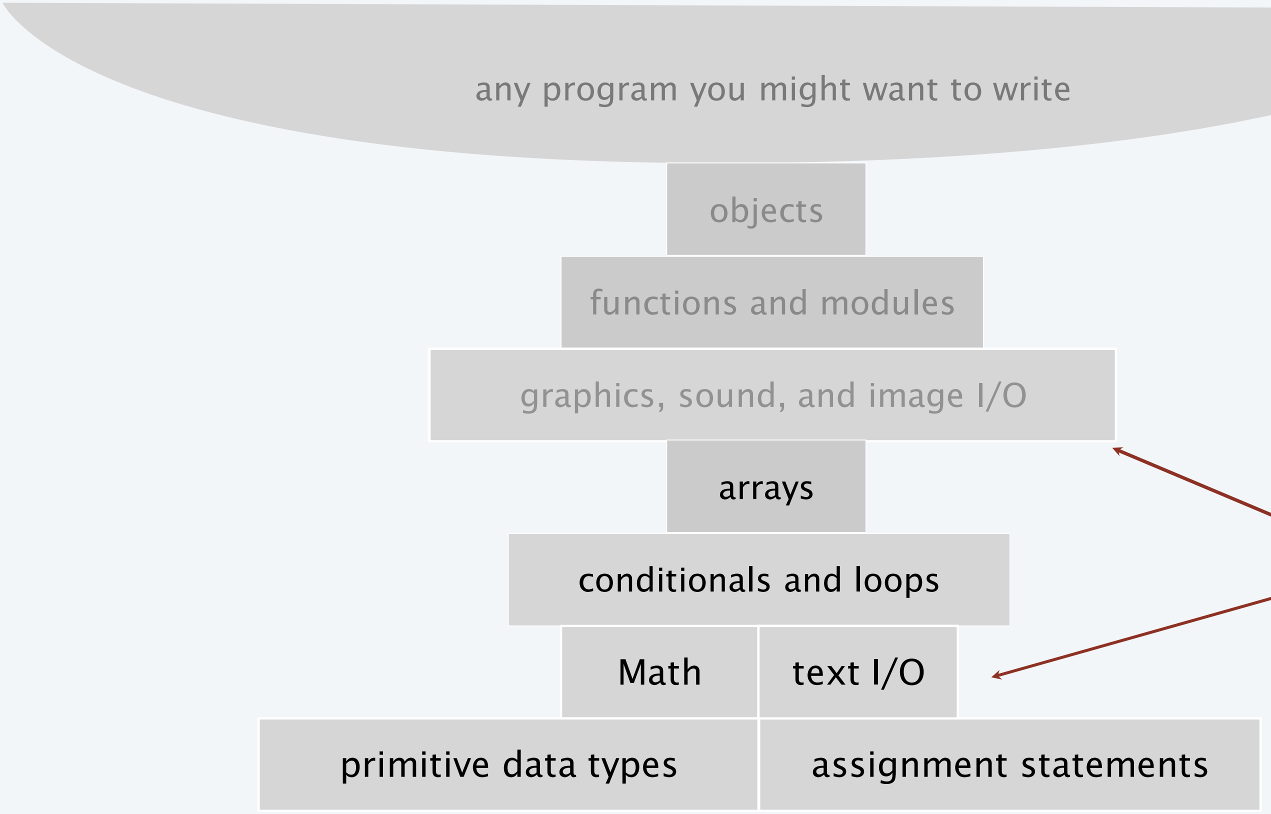
Rutgers University

4. Input and Output

- Standard input and output
- Standard drawing
- Fractal drawings (optional)
- Animation (optional)

Adopted and modified from slides by Sedgwick and Wayne

Basic building blocks for programming



any program you might want to write

objects

functions and modules

graphics, sound, and image I/O

arrays

conditionals and loops

Math

text I/O

primitive data types

assignment statements

Ability to interact with
the outside world

Input and output

Goal: Write Java programs that interact with the outside world via *input* and *output* devices.

Typical INPUT devices

Keyboard

Trackpad

Storage

Network

Camera

Microphone

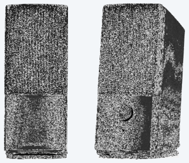
Typical OUTPUT devices

Display

Storage

Network

Printer


Speakers

Our approach.

- Define input and output *abstractions*.
- Use operating system (OS) functionality to connect our Java programs to actual devices.

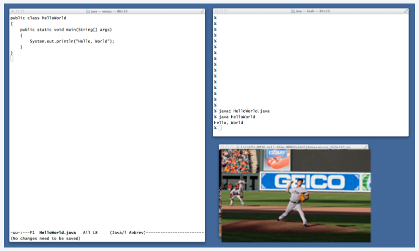
Abstraction

An *abstraction* is something that exists only as an idea.

Interested in thinking more deeply about this concept?
Consider taking a philosophy course.

Example: "Printing" is the idea of a program producing text as output.

Good abstractions *simplify* our view of the world, by *unifying* diverse real-world artifacts.



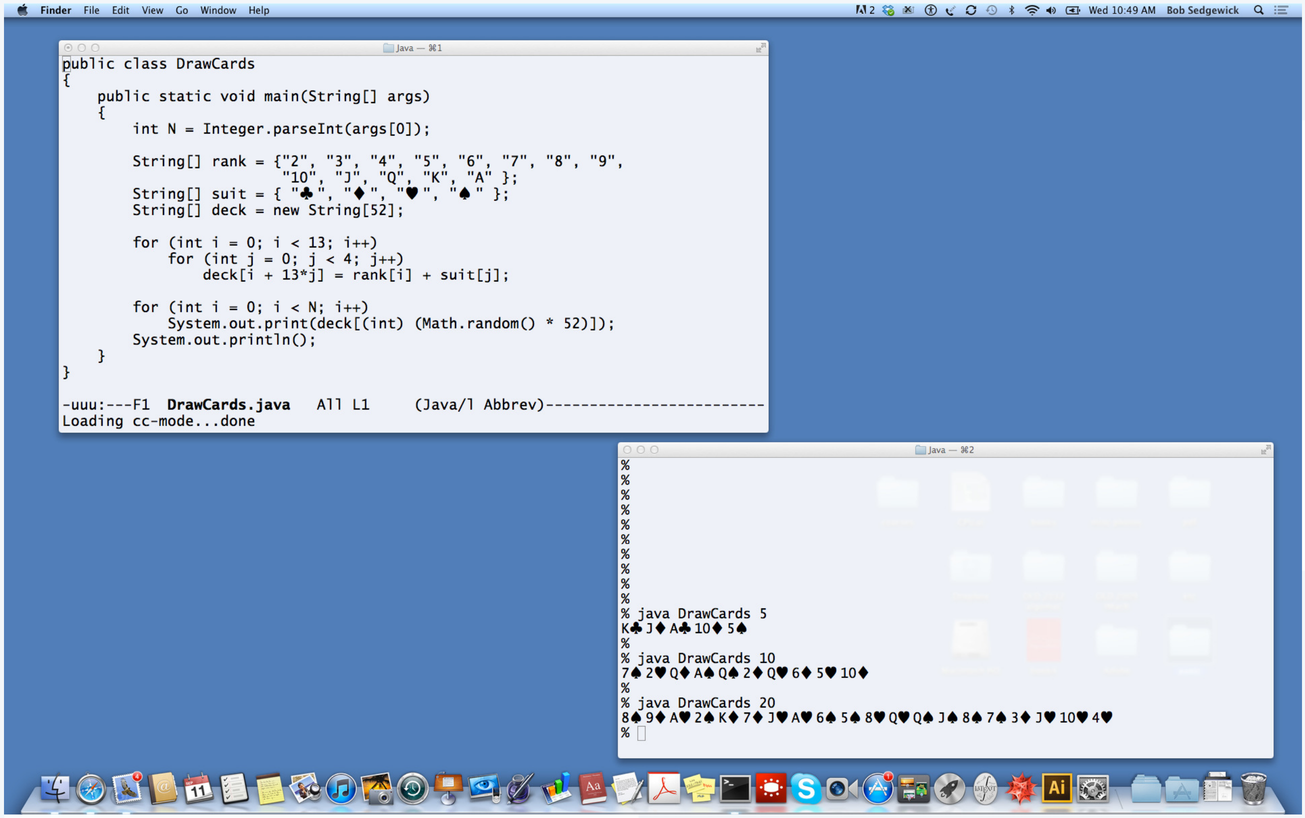
```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World");
    }
}

-uu-:---F1  HelloWorld.java   All L8     (Java/l Abbrev)------------------------
(No changes need to be saved)
```

```
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
%
% javac HelloWorld.java
% java HelloWorld
Hello, World
%
```

This lecture. Abstractions for delivering input to or receiving output from our programs.

Abstractions plays an *essential* role in understanding computation.

Terminal. An abstraction for providing input and output to a program.



```
public class DrawCards
{
    public static void main(String[] args)
    {
        int N = Integer.parseInt(args[0]);

        String[] rank = {"2", "3", "4", "5", "6", "7", "8", "9",
                         "10", "J", "Q", "K", "A" };
        String[] suit = { "♣", "♦", "♥", "♠" };
        String[] deck = new String[52];

        for (int i = 0; i < 13; i++)
            for (int j = 0; j < 4; j++)
                deck[i + 13*j] = rank[i] + suit[j];

        for (int i = 0; i < N; i++)
            System.out.print(deck[(int) (Math.random() * 52)]);
        System.out.println();
    }
}

-uuu:---F1  DrawCards.java   All L1     (Java/l Abbrev)------------------------
Loading cc-mode...done
```

```
%
%
%
%
%
%
%
%
%
%
% java DrawCards 5
K♣ J♦ A♣ 10♦ 5♠
%
% java DrawCards 10
7♠ 2♥ Q♦ A♠ Q♠ 2♦ Q♥ 6♦ 5♥ 10♦
%
% java DrawCards 20
8♠ 9♦ A♥ 2♠ K♦ 7♦ J♥ A♥ 6♠ 5♠ 8♥ Q♥ Q♠ J♠ 8♠ 7♠ 3♦ J♥ 10♥ 4♥
% 
```

Finder File Edit View Go Window Help
Wed 10:49 AM Bob Sedgewick
Java — ⌘1
Java — ⌘2

```
% java DrawCards 10
7♠ 2♥ Q♦ A♠ Q♠ 2♦ Q♥ 6♦ 5♥ 10♦
```

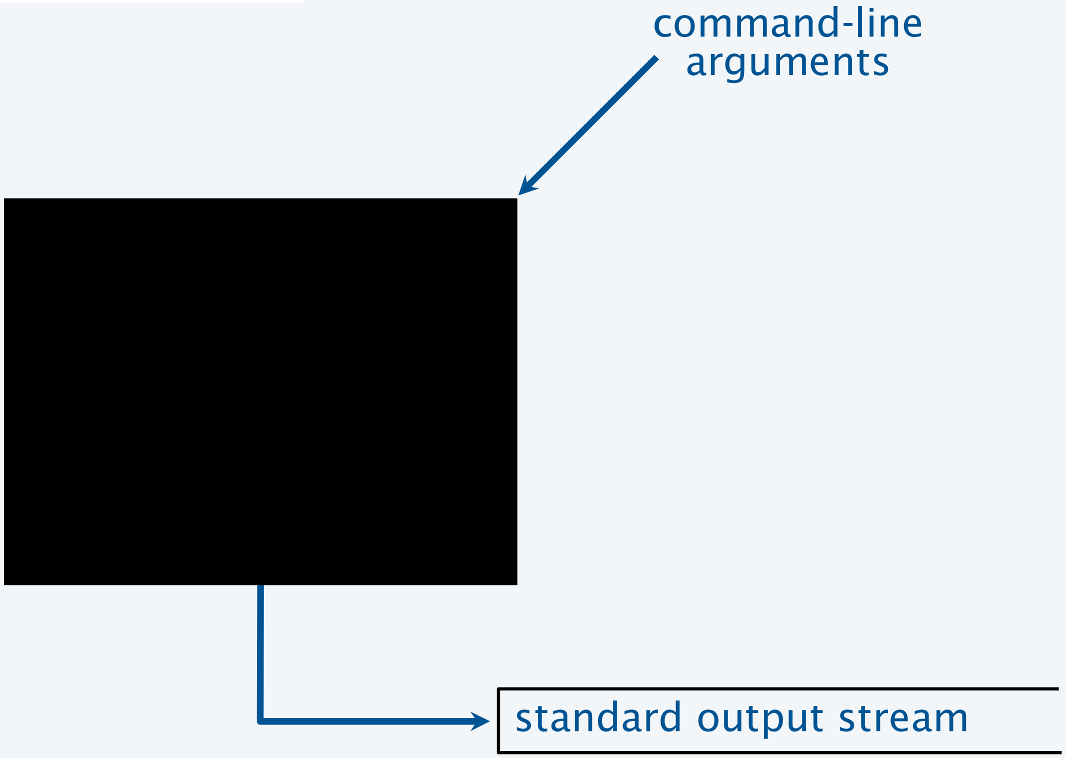
Input from command line

Output to standard output stream

Virtual VT-100 terminal

Input-output abstraction (so far)

A mental model of what a Java program does.



command-line arguments

standard output stream

Review: command-line input

LO 6.1abcd

Command-line input. An abstraction for providing arguments (strings) to a program.

Basic properties

- Strings you type after the program name are available as args[0], args[1], ... at *run* time.
- Arguments are available when the program *begins* execution.
- Need to call system conversion methods to convert the strings to other types of data.

```
public class RandomInt
{
   public static void main(String[] args)
   {
      int N = Integer.parseInt(args[0]);
      double r = Math.random();
      int t = (int) (r * N);
      System.out.println(t);
   }
}
```

```
% java RandomInt 6
3

% java RandomInt 10000
3184
```

Review: standard output

Infinity. An abstraction describing something having no limit.

Standard output stream. An abstraction for an infinite output sequence.

Basic properties

- Strings from System.out.println() are added to the end of the standard output stream.
- Standard output stream is sent to terminal application by default.

```
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());
   }
}
```

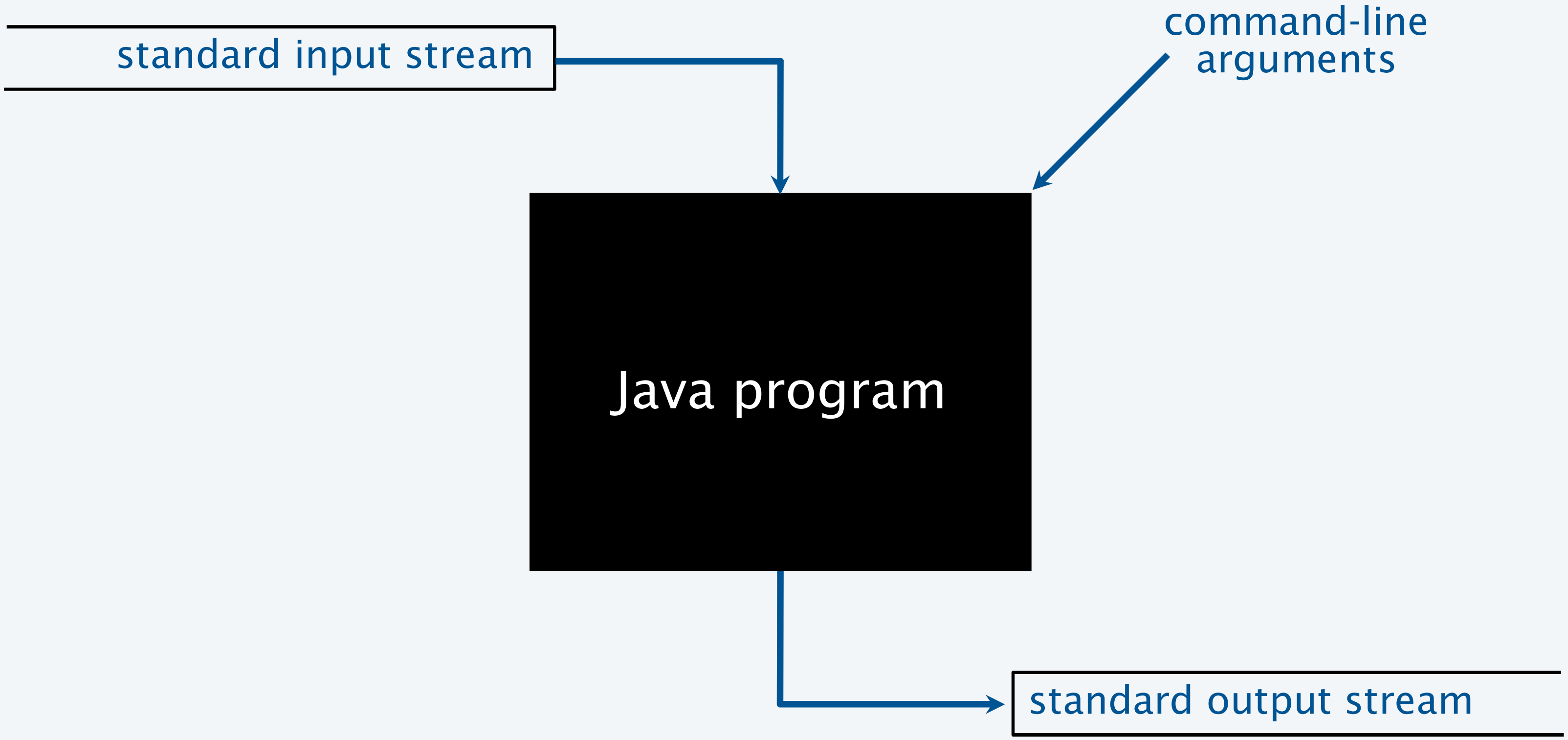
```
% java RandomSeq 4
0.9320744627218469
0.4279508713950715
0.08994615071160994
0.6579792663546435
```

```
% java RandomSeq 1000000
0.09474882292442943
0.2832974030384712
0.1833964252856476
0.2952177517730442
0.8035985765979008
0.7469424300071382
0.5835267075283997
0.3455279612587455
...
```

No limit on amount of output

Improved input-output abstraction

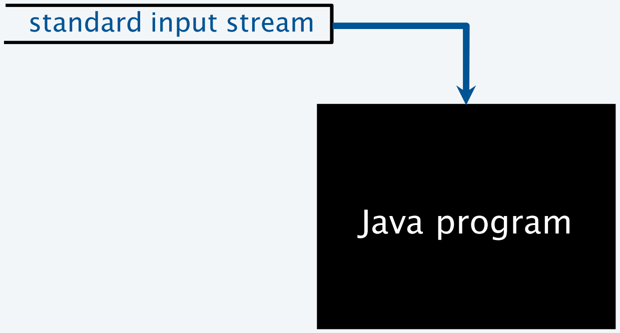
Add an infinite *input* stream.



Standard input

Infinity. An abstraction describing something having no limit.

Standard input stream. An abstraction for an infinite *input* sequence.



Advantages over command-line input

- Can provide new arguments *while* the program is executing.
- No limit on the amount of data we can input to a program.
- Conversion to primitive types is explicitly handled (stay tuned).

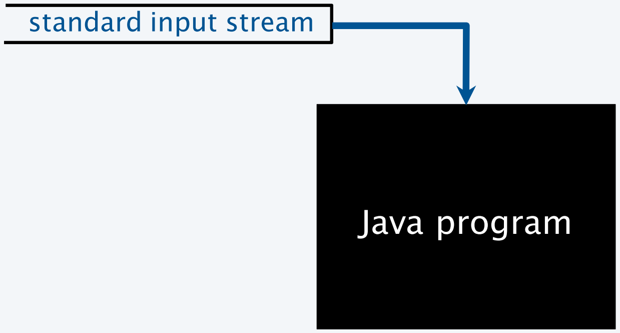
StdIn library (book version)

LO 6.1e

Developed for this book, but broadly useful

- Implement abstractions invented for UNIX in the 1970s.
- Available for download at book site.
- Included in introcs software you downloaded at the beginning of the course.

public class StdIn	
boolean isEmpty()	true *iff no more values*
int readInt()	*read a value of type* int
double readDouble()	*read a value of type* double
long readLong()	*read a value of type* long
boolean readBoolean()	*read a value of type* boolean
char readChar()	*read a value of type* char
String readString()	*read a value of type* String
String readAll()	*read the rest of the text*



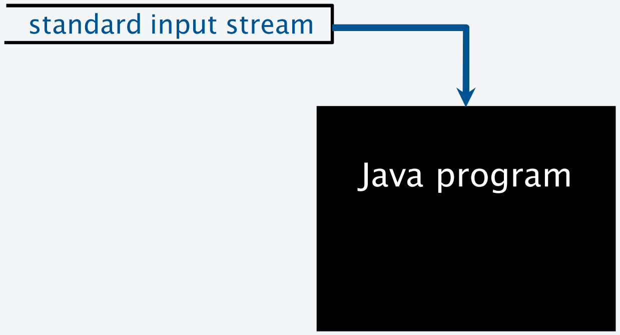
standard input stream

Java program

StdIn library (processing chars and Strings)

LO 6.1fg

public class StdIn	
boolean hasNextChar()	true *iff standard input has more characters*
char readChar()	*read a value of type* char and returns
boolean hasNextLine()	true *iff standard input has more input lines*
String readLine()	*read remaining portion of line as type* String
String readAll()	*read remaining portion of input as type String*



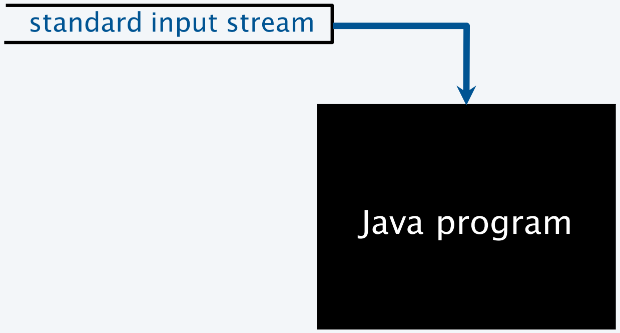
standard input stream

Java program

StdIn library

LO 6.1h

public class StdIn	
double[] readAllDoubles()	*read all remaining doubles and return in an array*
int[] readAllInts()	*read all remaining ints and return in an array*
long[] readAllLongs()	*read all remaining longs and return in an array*
String[] readAllStrings()	*read all remaining Strings and return in an array*
String[] readAllLines()	*read all remaining lines and return in an array of Strings*



standard input stream

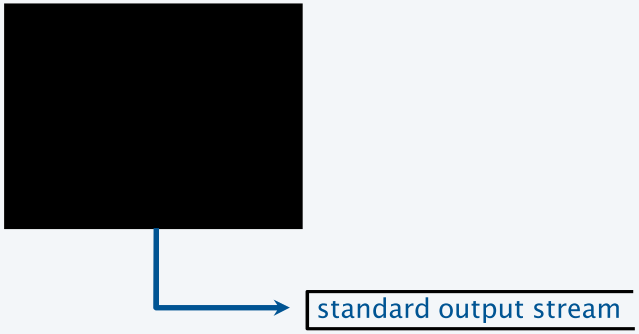
Java program

StdOut library

Developed for this course, but broadly useful

- Implement abstractions invented for UNIX in the 1970s.
- Included in introcs software you downloaded at the beginning of the course.

public class StdOut	
void print(String s)	*put* s *on the output stream*
void println()	*put a newline on the output stream*
void println(String s)	*put* s, *then a newline on the stream*
void printf(String f, ...)	*formatted output*



Q. These are the same as System.out. Why not just use System.out?

A. We provide a consistent set of simple I/O abstractions in one place.

A. We can make output *independent* of system, language, and locale.

use StdOut
from now on

StdIn/StdOut warmup

Interactive input

- Prompt user to type inputs on standard input stream.
- Mix input stream with output stream.

```
public class AddTwo
{
  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.println("Their sum is " + sum);
  }
}
```

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

StdIn application: average the numbers on the standard input stream

Average

- Read a stream of numbers.
- Compute their average.

Q. How do I specify the end of the stream?

A. <Ctrl-d> (standard for decades).

A. <Ctrl-z> (Windows).

```
public class Average
{
   public static void main(String[] args)
   {
      double sum = 0.0;  // cumulative total
      int n = 0;         // number of values
      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
```

Key points

- No limit on the size of the input stream.
- Input and output can be interleaved.

Summary: prototypical applications of standard output and standard input

StdOut: Generate a stream of random numbers

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

StdIn: Compute the average of a stream of numbers

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

Both streams are *infinite* (no limit on their size).

Q. Do I always have to type in my input data and print my output?

A. No! Keep data and results in *files* on your computer, or use *piping* to connect programs.

Redirection: keep data in files on your computer

Redirect standard output to a file

```
% java RandomSeq 1000000 > data.txt

% more data.txt
0.09474882292442943
0.2832974030384712
0.1833964252856476
0.2952177517730442
0.8035985765979008
0.7469424300071382
0.5835267075283997
0.3455279612587455
...
```

"redirect standard output to"

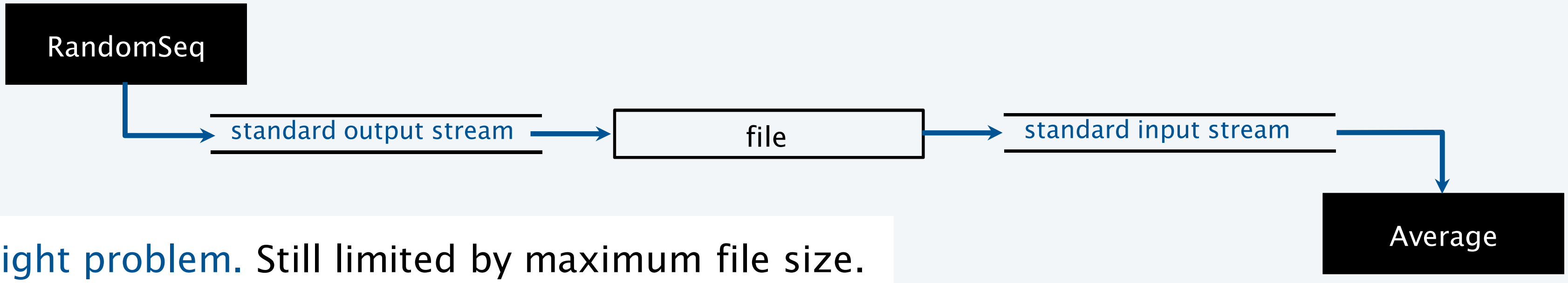
LO 6.1j

Redirect from a file to standard input

```
% java Average < data.txt
0.4947655567740991
```

"take standard input from"

LO 6.1k



Slight problem. Still limited by maximum file size.

Piping: entirely avoid saving data

Q. There's no room for a huge file on my computer. Now what?

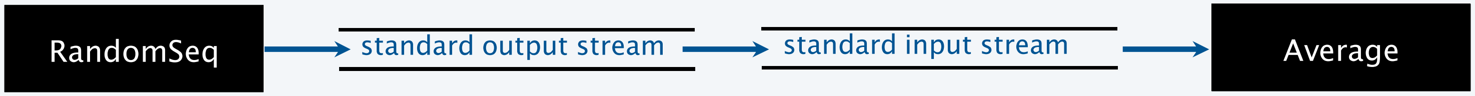
A. No problem! Use *piping*.

Piping. Connect standard output of one program to standard input of another.

set up a pipe

```
% java RandomSeq 1000000 | java Average
0.4997970473016028

% java RandomSeq 1000000 | java Average
0.5002071875644842
```

LO 6.1l



Critical point. No limit *within programs* on the amount of data they can handle.

It is the job of the *system* to collect data on standard output and provide it to standard input.

Streaming algorithms

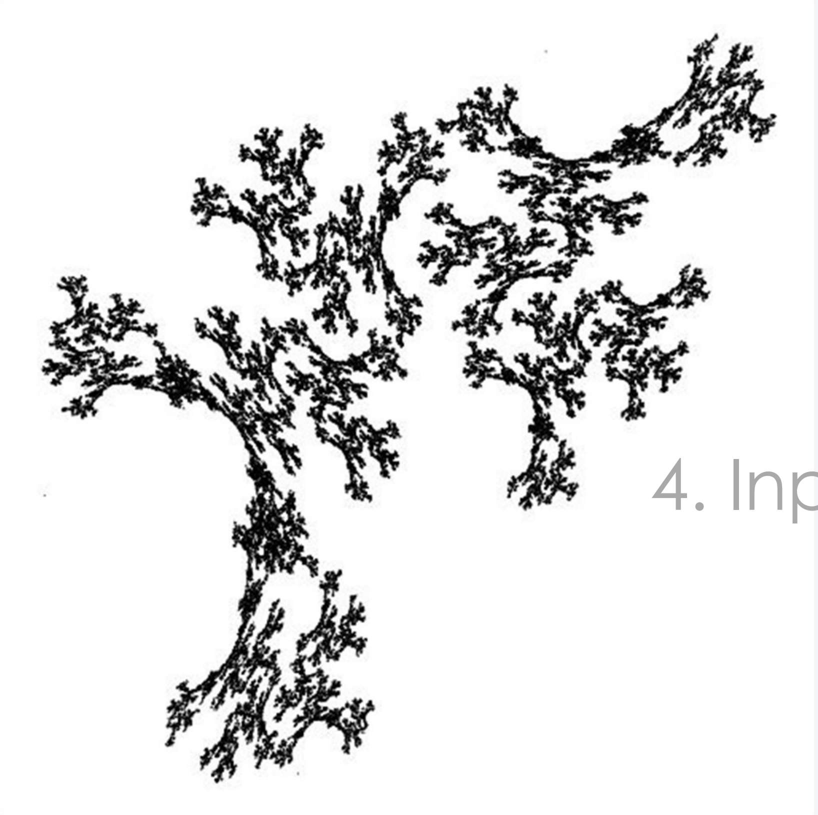
Early computing

- Amount of available memory was much smaller than amount of data to be processed.
- *But* dramatic increases happened every year.
- Redirection and piping enabled programs to handle much more data than computers could store.

Modern computing

- Amount of available memory *is* much smaller than amount of data to be processed.
- Dramatic increases *still* happen every year.
- *Streaming algorithms* enable our programs to handle much more data than our computers can store.

Lesson. Avoid limits *within your program* whenever possible.



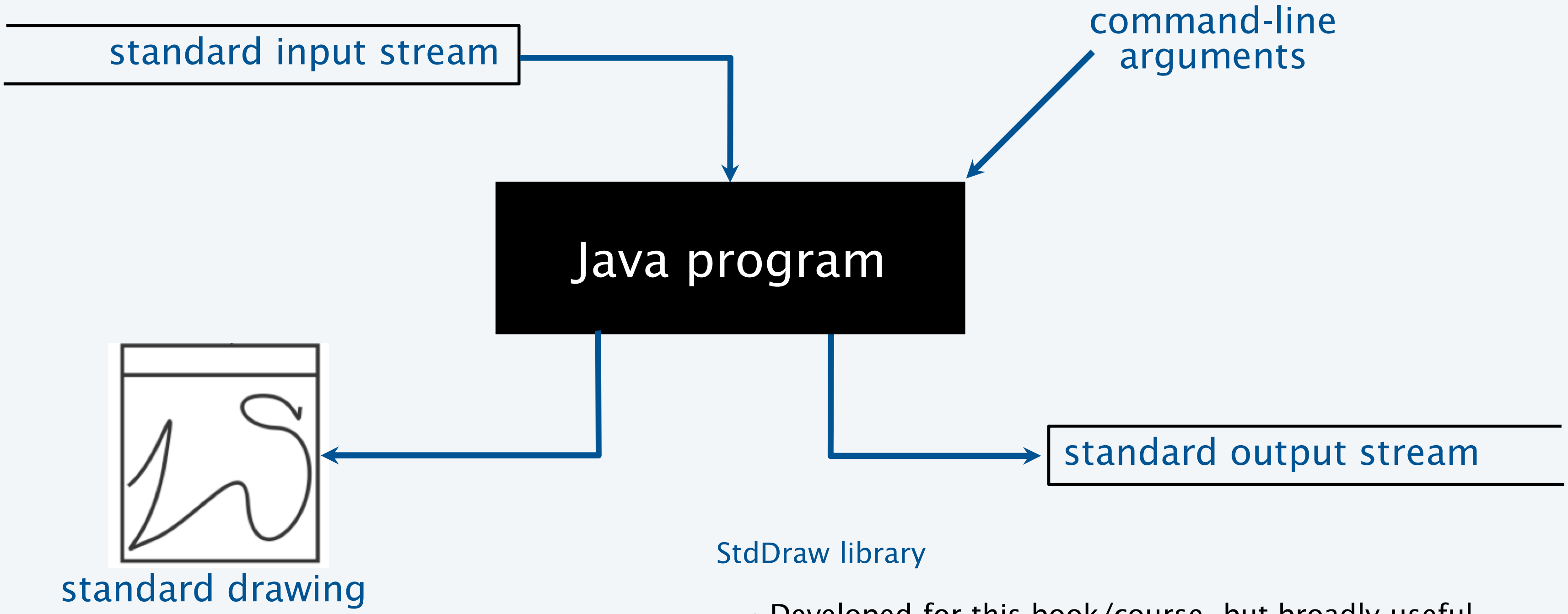
INTRODUCTION TO
COMPUTER SCIENCE
Rutgers University

4. Input and Output

- Standard input and output
- Standard drawing
- Fractal drawings
- Animation

Further improvements to our I/O abstraction

Add the ability to create a *drawing*.



StdDraw library

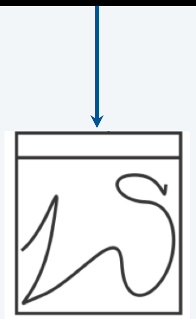
- Developed for this book/course, but broadly useful.
- Available for download at booksite.
- Included in introcs software.

StdDraw library

public class StdDraw	
void line(double x0, double y0, double x1, double y1)	
void point(double x, double y)	
void text(double x, double y, String s)	
void circle(double x, double y, double r)	
void square(double x, double y, double r)	
void polygon(double x, double y, double r)	
void picture(double x, double y, String filename)	*place* .gif, .jpg *or* .png *file*
void setPenRadius(double r)	
void setPenColor(Color c)	
void setXscale(double x0, double x1)	*reset x range to* [x_0, x_1)
void setYscale(double y0, double y1)	*reset y range to* [y_0, y_1)
void show(int dt)	*show all; pause* dt *millisecs*

also filledCircle(), filledSquare(),
and filledPolygon()

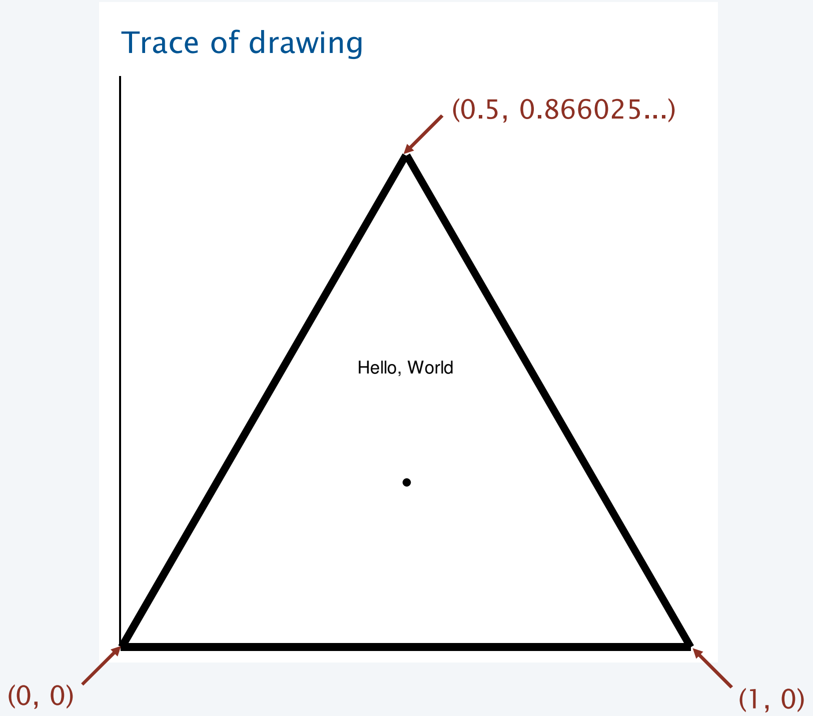
[redacted]



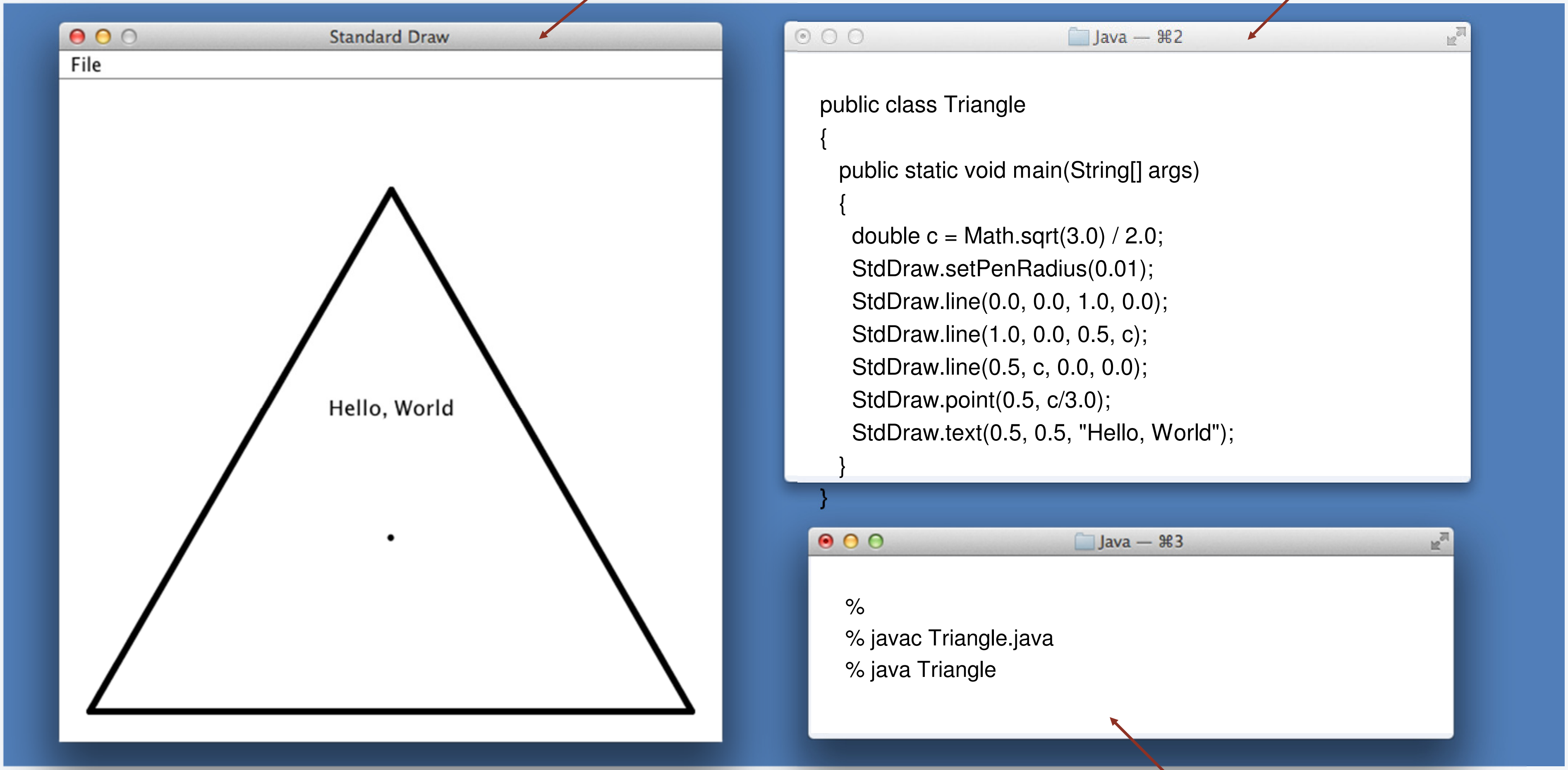
standard drawing

“Hello, World” for StdDraw

```
public class Triangle
{
   public static void main(String[] args)
   {
      double c = Math.sqrt(3.0) / 2.0;
      StdDraw.setPenRadius(0.01);
      StdDraw.line(0.0, 0.0, 1.0, 0.0);
      StdDraw.line(1.0, 0.0, 0.5, c);
      StdDraw.line(0.5, c, 0.0, 0.0);
      StdDraw.point(0.5, c/3.0);
      StdDraw.text(0.5, 0.5, "Hello, World");
   }
}
```



“Hello, World” for StdDraw



window for standard drawing

virtual terminal for editor

```
public class Triangle
{
  public static void main(String[] args)
  {
    double c = Math.sqrt(3.0) / 2.0;
    StdDraw.setPenRadius(0.01);
    StdDraw.line(0.0, 0.0, 1.0, 0.0);
    StdDraw.line(1.0, 0.0, 0.5, c);
    StdDraw.line(0.5, c, 0.0, 0.0);
    StdDraw.point(0.5, c/3.0);
    StdDraw.text(0.5, 0.5, "Hello, World");
  }
}
```

```
%
% javac Triangle.java
% java Triangle
```

virtual terminal for OS commands

StdDraw application: data visualization

```
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);
      }
   }
}
```

read coords of bounding box

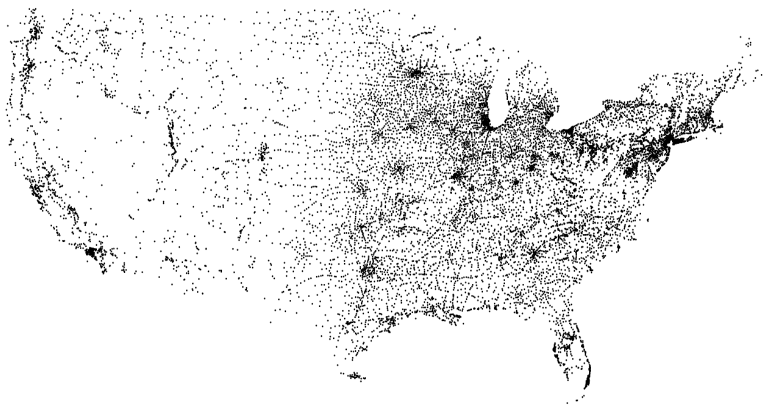
rescale

read and plot a point

bounding box coords

```
% 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
```

sequence of point coordinates (13,509 cities)



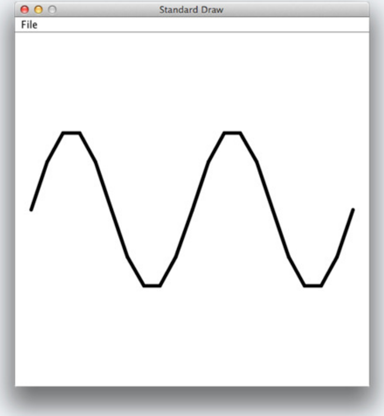
StdDraw application: plotting a function

Goal. Plot $y = \sin(4x) + \sin(20x)$ in the interval $(0, \pi)$.

Method. Take *N* samples, regularly spaced.

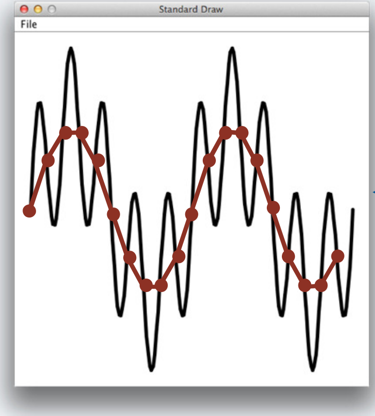
```
public class PlotFunctionEx
{
  public static void main(String[] args)
  {
    int N = Integer.parseInt(args[0]);
    double[] x = new double[N+1];
    double[] y = new double[N+1];
    for (int i = 0; i <= N; i++)
    {
      x[i] = Math.PI * i / N;
      y[i] = Math.sin(4*x[i]) + Math.sin(20*x[i]);
    }
    StdDraw.setXscale(0, Math.PI);
    StdDraw.setYscale(-2.0, +2.0);
    for (int i = 0; i < N; i++)
      StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
  }
```

```
% java PlotFunctionEx 20
```

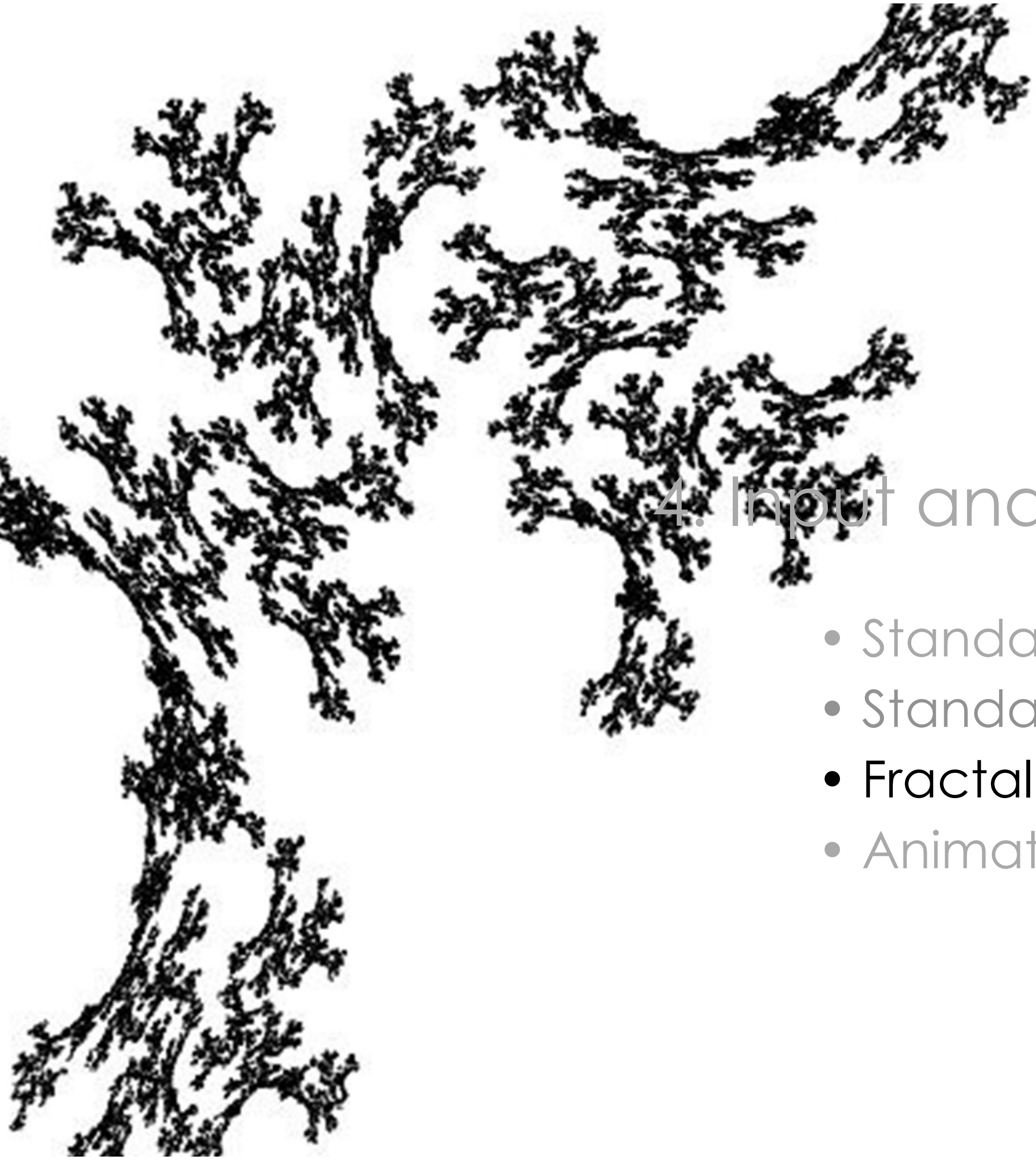


Lesson 1: Plotting is easy. →

```
% java PlotFunctionEx 200
```



← Lesson 2: Take a sufficiently large sample—otherwise you might miss something!



INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

4. Input and Output

- Standard input and output
- Standard drawing
- Fractal drawings (optional)
- Animation

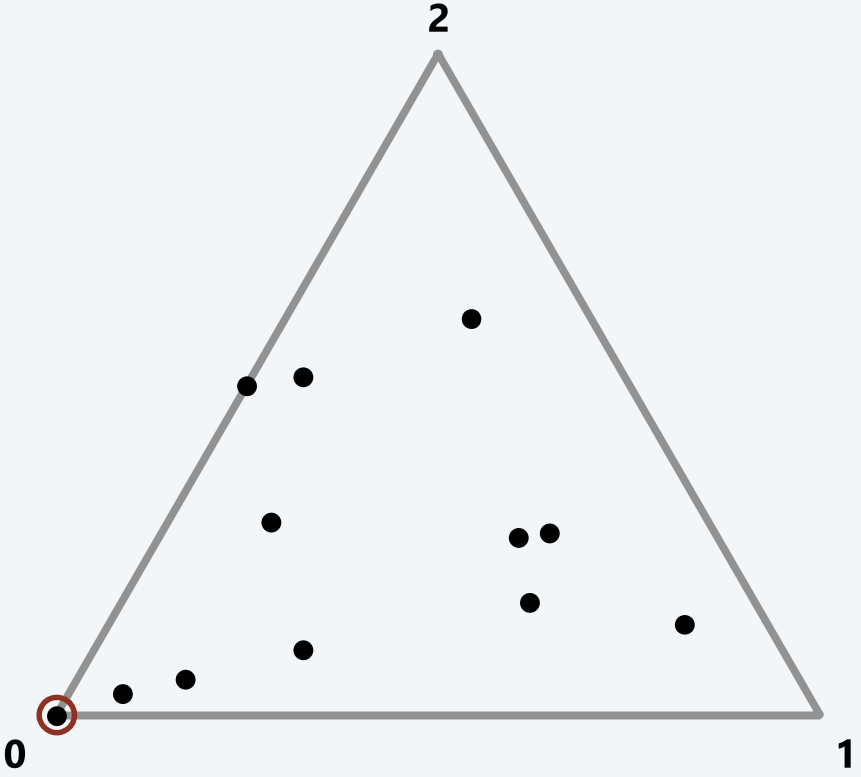
StdDraw application: a random game

Draw an equilateral triangle, number the vertices 0, 1, 2 and make 0 the *current point.*

- Pick a vertex at random.
- Draw a point halfway between that vertex and the current point.
- Repeat.

vertex	*ID*	*probability*	*new x*	*new y*
(0, 0)	0	1/3	.5*x*	.5*y*
(1, 0)	1	1/3	.5*x* + .5	.5*y*
(.5, √3/2)	2	1/3	.5*x* + .25	.5*y* + .433

0	1	2	3	4	5	6	7	8	9	10
2	**1**	**2**	**0**	**1**	**0**	**0**	**0**	**2**	**1**	**1**

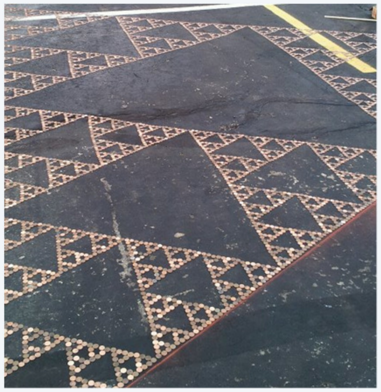


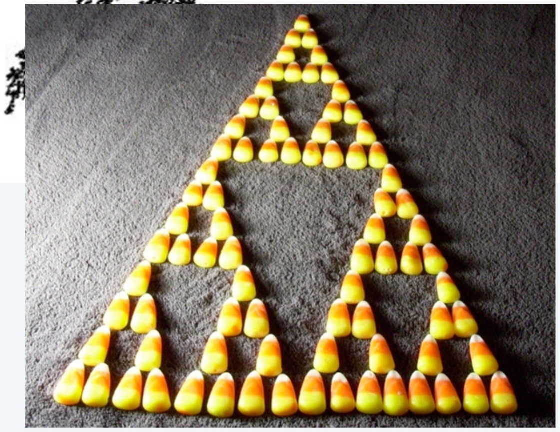
StdDraw application: a random game

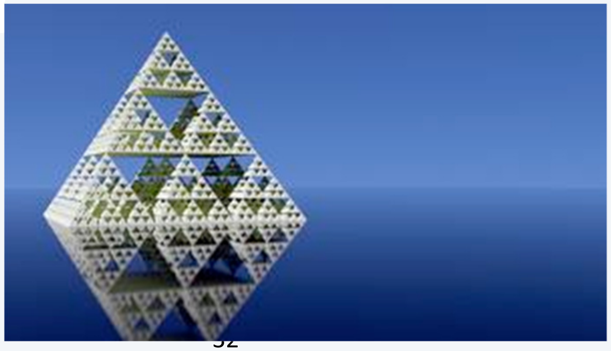
```
public class Chaos
{
   public static void main(String[] args)
   {
      int trials = Integer.parseInt(args[0]);

      double c = Math.sqrt(3.0) / 2.0;
      double[] cx = { 0.000, 1.000, 0.500 };
      double[] cy = { 0.000, 0.000, c };

      StdDraw.setPenRadius(0.01);
      double x = 0.0, y = 0.0;
      for (int t = 0; t < trials; t++)
      {
         int r = (int) (Math.random() * 3);
         x = (x + cx[r]) / 2.0;
         y = (y + cy[r]) / 2.0;
         StdDraw.point(x, y);
      }
   }
}
```

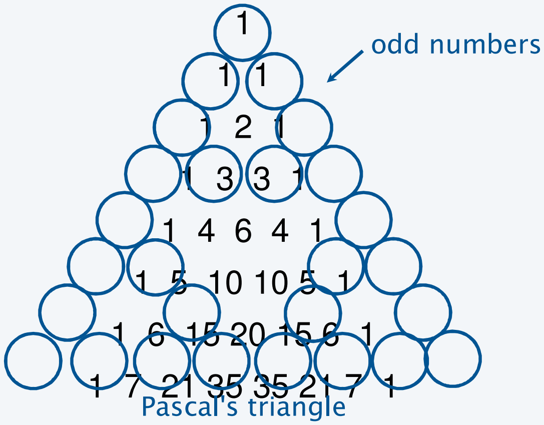
```
% java Chaos 10000
```

Sierpinski triangles in the wild









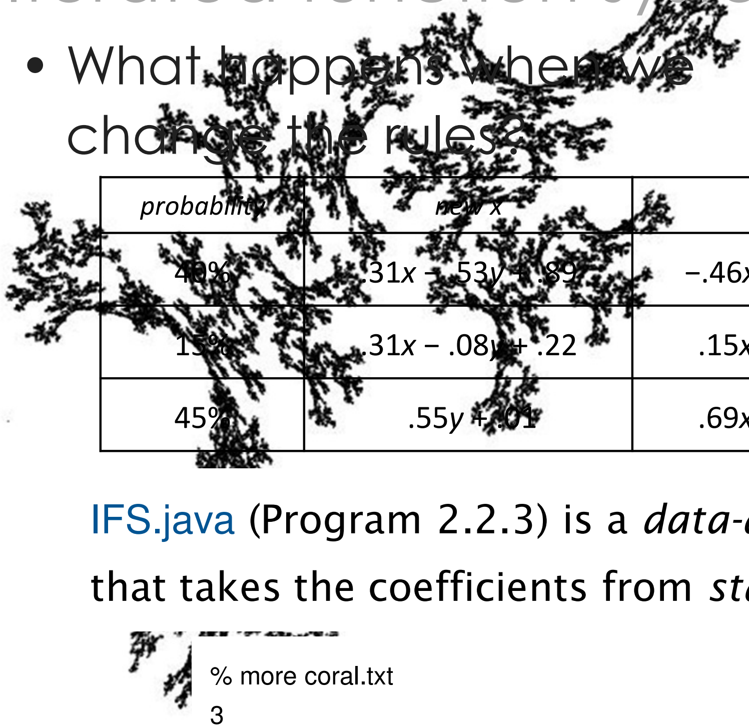
odd numbers

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1

Pascal's triangle

Iterated function systems

- What happens when we change the rules?

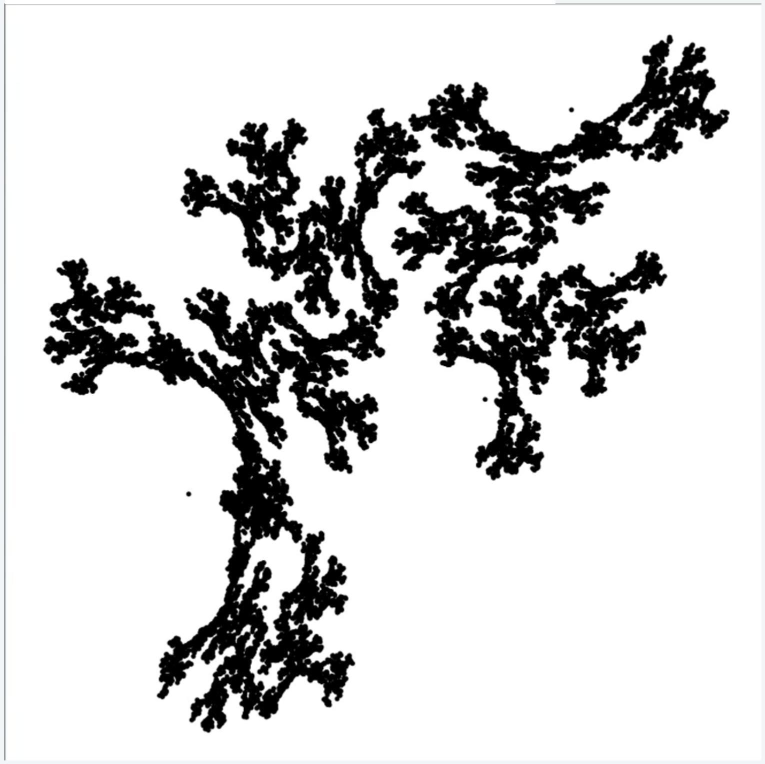


probability	*new x*	*new y*
40%	.31*x* − .53*y* + .89	−.46*x* − .29y + 1.10
15%	.31*x* − .08*y* + .22	.15*x* − .45*y* + .34
45%	.55*y* + .01	.69*x* − .20*y* + .38

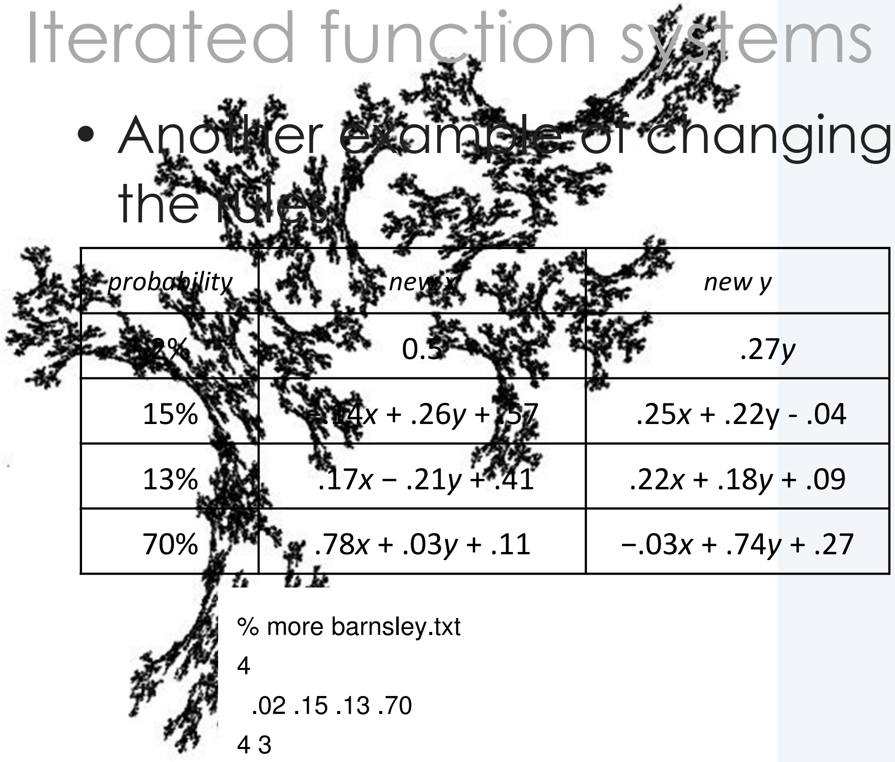
IFS.java (Program 2.2.3) is a *data-driven* program that takes the coefficients from *standard input.*

```
% more coral.txt
3
  0.40  0.15  0.45
3 3
  0.307692 -0.531469  0.8863493
  0.307692 -0.076923  0.2166292
  0.000000  0.545455  0.0106363
3 3
 -0.461538 -0.293706  1.0962865
  0.153846 -0.447552  0.3383760
  0.692308 -0.195804  0.3808254
```

```
% java IFS 10000 < coral.txt
```



Iterated function systems

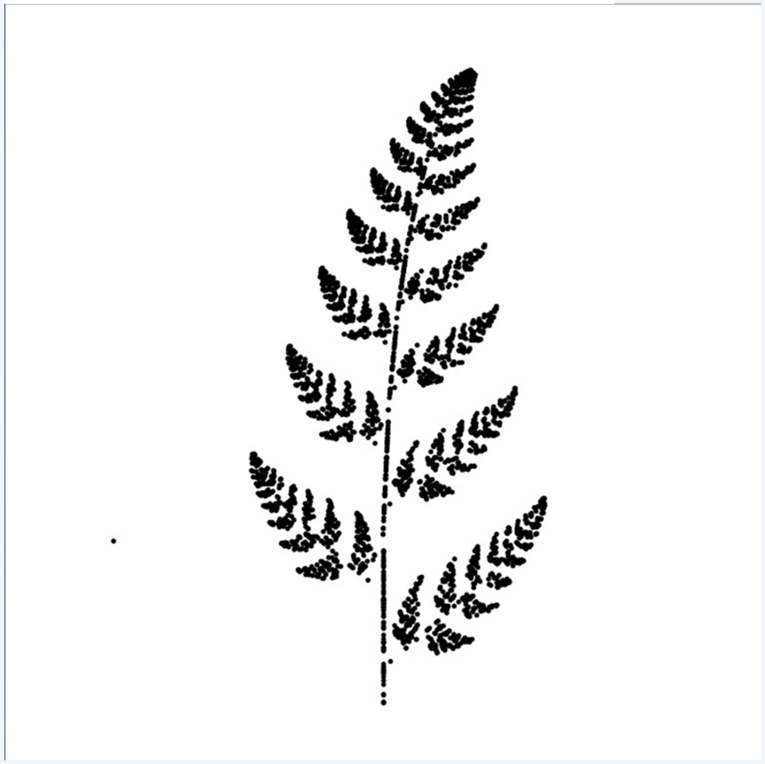


- Another example of changing the rules

probability	*new x*	*new y*
2%	0.5	.27*y*
15%	−.14*x* + .26*y* + .57	.25*x* + .22y - .04
13%	.17*x* − .21*y* + .41	.22*x* + .18*y* + .09
70%	.78*x* + .03*y* + .11	−.03*x* + .74*y* + .27

```
% more barnsley.txt
4
  .02 .15 .13 .70
4 3
  .000  .000  .500
 -.139  .263  .570
  .170 -.215  .408
  .781  .034  .1075
4 3
  .000  .270  .000
  .246  .224 -.036
  .222  .176  .0893
 -.032  .739  .270
```

```
% java IFS 10000 < barnsley.txt
```



Iterated function systems

Simple iterative computations yield patterns that are remarkably similar to those found in the natural world.

Q. What does computation tell us about nature?

Q. What does nature tell us about computation?

20th century sciences. Formulas.

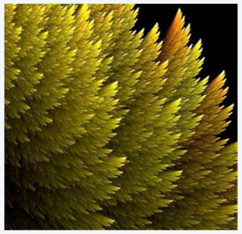
21st century sciences. Algorithms?

Note. You have seen many practical applications of integrated function systems, in movies and games.

an IFS fern

a real fern

a real plant

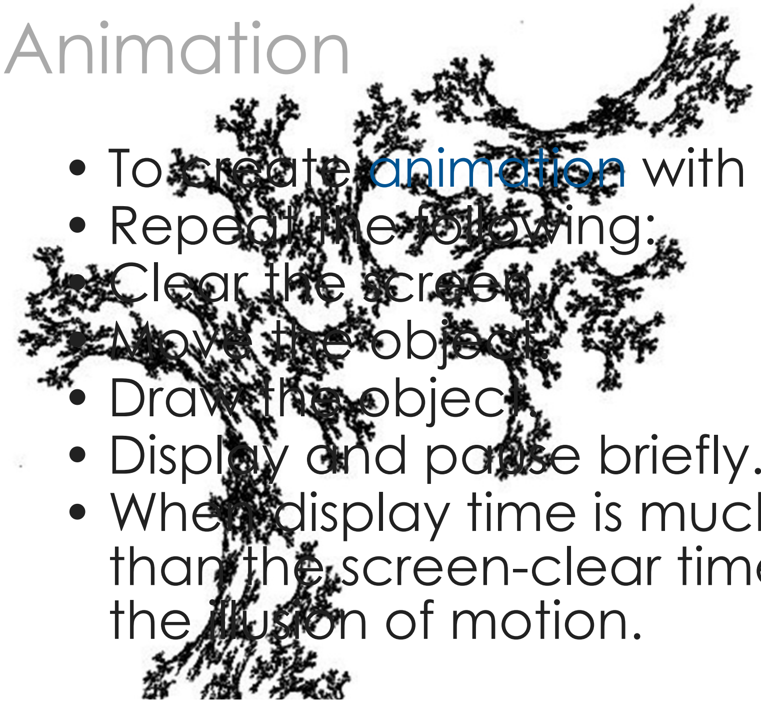

an IFS plant

INTRODUCTION TO
COMPUTER SCIENCE
Rutgers University

4. Input and Output

- Standard input and output
- Standard drawing
- Fractal drawings
- Animation (optional)

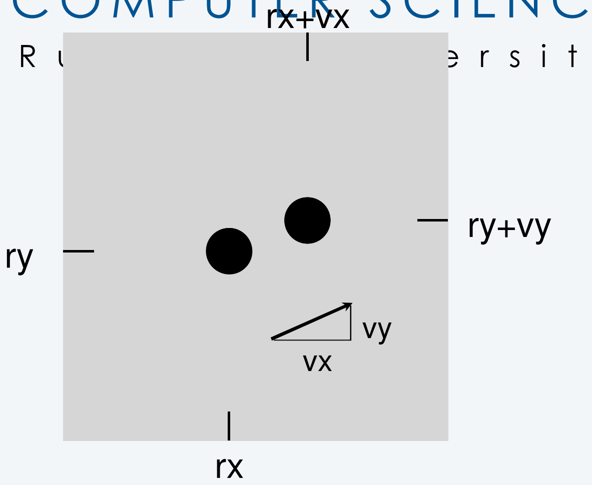
Animation



- To create animation with StdDraw.
- Repeat the following:
- Clear the screen.
- Move the object.
- Draw the object.
- Display and pause briefly.
- When display time is much greater than the screen-clear time, we have the illusion of motion.

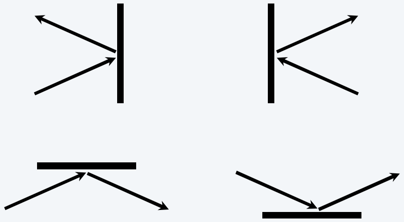
INTRODUCTION TO COMPUTER SCIENCE

R ersity



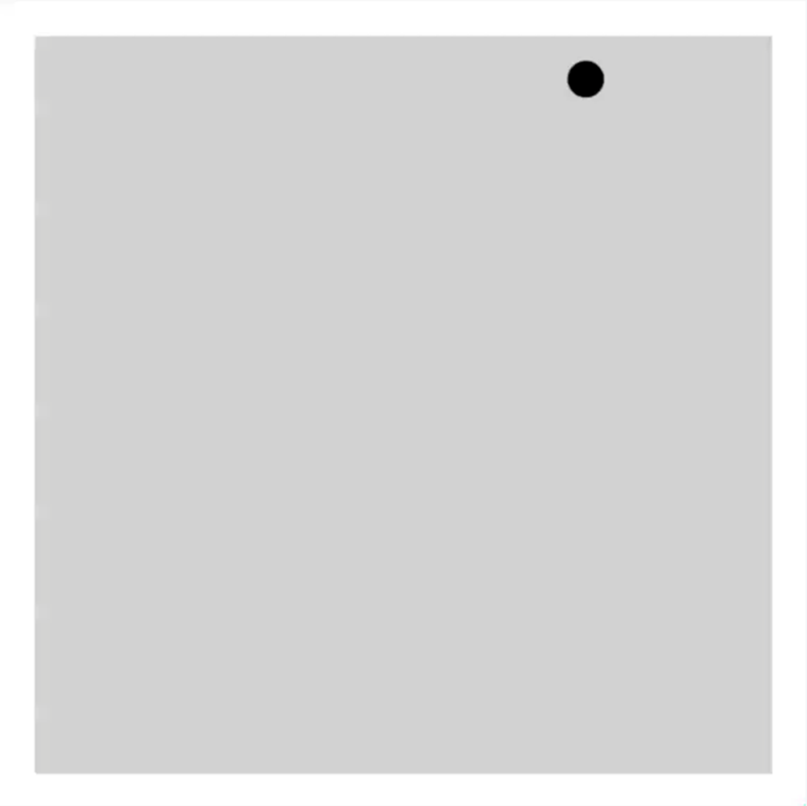
Bouncing ball.

- Ball has position (rx, ry) and constant velocity (vx, vy).
- To *move* the ball, update position to (rx+vx, ry+vy).
- If the ball hits a *vertical* wall, set vx to -vx.
- If the ball hits a *horizontal* wall, set vy to -vy.



```
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)
    {
      StdDraw.setPenColor(StdDraw.LIGHT_GRAY);
      StdDraw.filledSquare(0.0, 0.0, 1.0);
      if (Math.abs(rx + vx) + radius > 1.0) vx = -vx;
      if (Math.abs(ry + vy) + radius > 1.0) vy = -vy;
      rx = rx + vx;
      ry = ry + vy;
      StdDraw.setPenColor(StdDraw.BLACK);
      StdDraw.filledCircle(rx, ry, radius);
      StdDraw.show(20);
    }
  }
}
```

```
% java BouncingBall
```



Pop quiz on animation

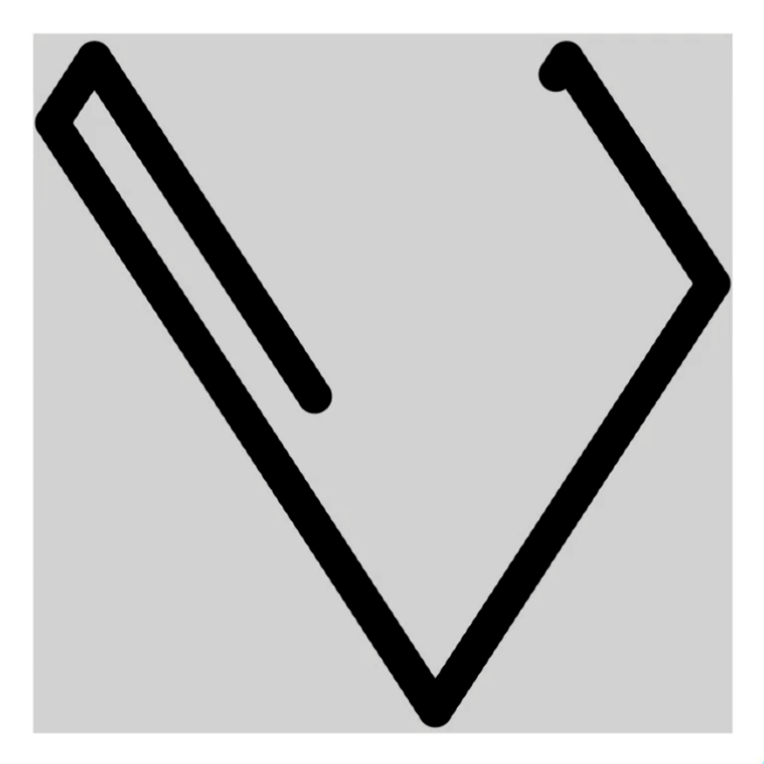
Q. What happens if we move *clear the screen* out of the loop?

```
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)
    {
      StdDraw.setPenColor(StdDraw. LIGHT_GRAY);
      StdDraw.filledSquare(0.0, 0.0, 1.0);


      if (Math.abs(rx + vx) + radius > 1.0) vx = -vx;
      if (Math.abs(ry + vy) + radius > 1.0) vy = -vy;
      rx = rx + vx;
      ry = ry + vy;
      StdDraw.setPenColor(StdDraw.BLACK);
      StdDraw.filledCircle(rx, ry, sz);
      StdDraw.show(20);
    }
```

Pop quiz on animation

Q. What happens if we move *clear the screen* out of the loop?

```
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);
    StdDraw.setPenColor(StdDraw. LIGHT_GRAY);
    StdDraw.filledSquare(0.0, 0.0, 1.0);
    while(true)
    {
      if (Math.abs(rx + vx) + radius > 1.0) vx = -vx;
      if (Math.abs(ry + vy) + radius > 1.0) vy = -vy;
      rx = rx + vx;
      ry = ry + vy;
      StdDraw.setPenColor(StdDraw.BLACK);
      StdDraw.filledCircle(rx, ry, radius);
      StdDraw.show(20);
    }
  }
}
```



A. We see the ball's entire path.

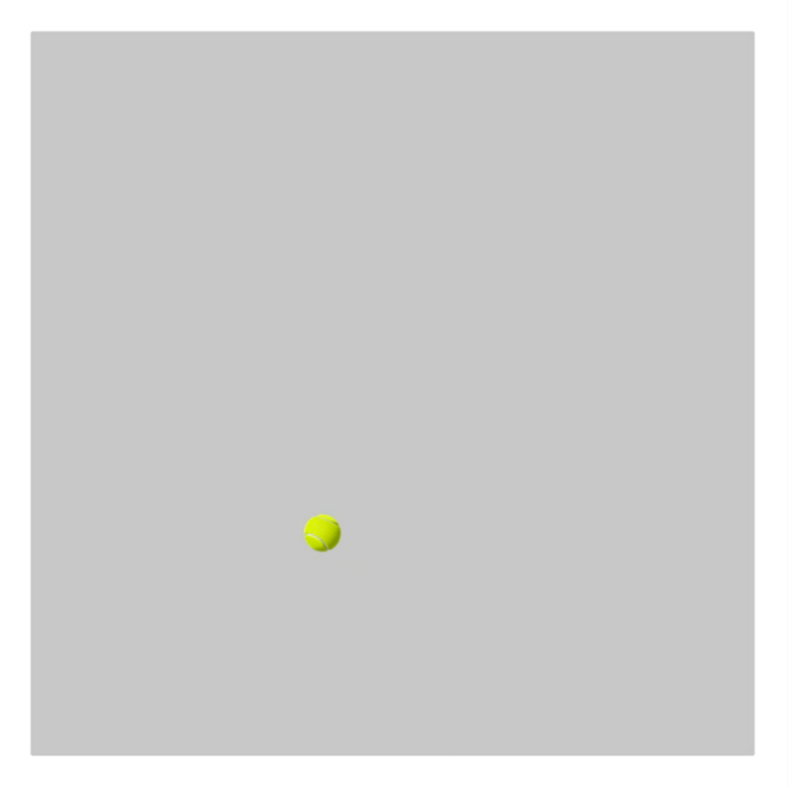
Deluxe bouncing ball

INTRODUCTION TO
COMPUTER SCIENCE

sity

```
public class BouncingBallDeluxe
{
   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)
      {
        StdDraw.setPenColor(StdDraw. LIGHT_GRAY);
        StdDraw.filledSquare(0.0, 0.0, 1.0);
        if (Math.abs(rx + vx) + radius > 1.0)
        {  StdAudio.play("pipebang.wav"); vx = -vx;  }
        if (Math.abs(ry + vy) + radius > 1.0)
        {  StdAudio.play("pipebang.wav"); vy = -vy;  }
        rx = rx + vx;
        ry = ry + vy;
        StdDraw.picture(rx, ry, "TennisBall.png");
        StdDraw.show(20);
      }
   }
```

```
% java BouncingBallDeluxe
```



Stay tuned to next lecture for full description of StdAudio.

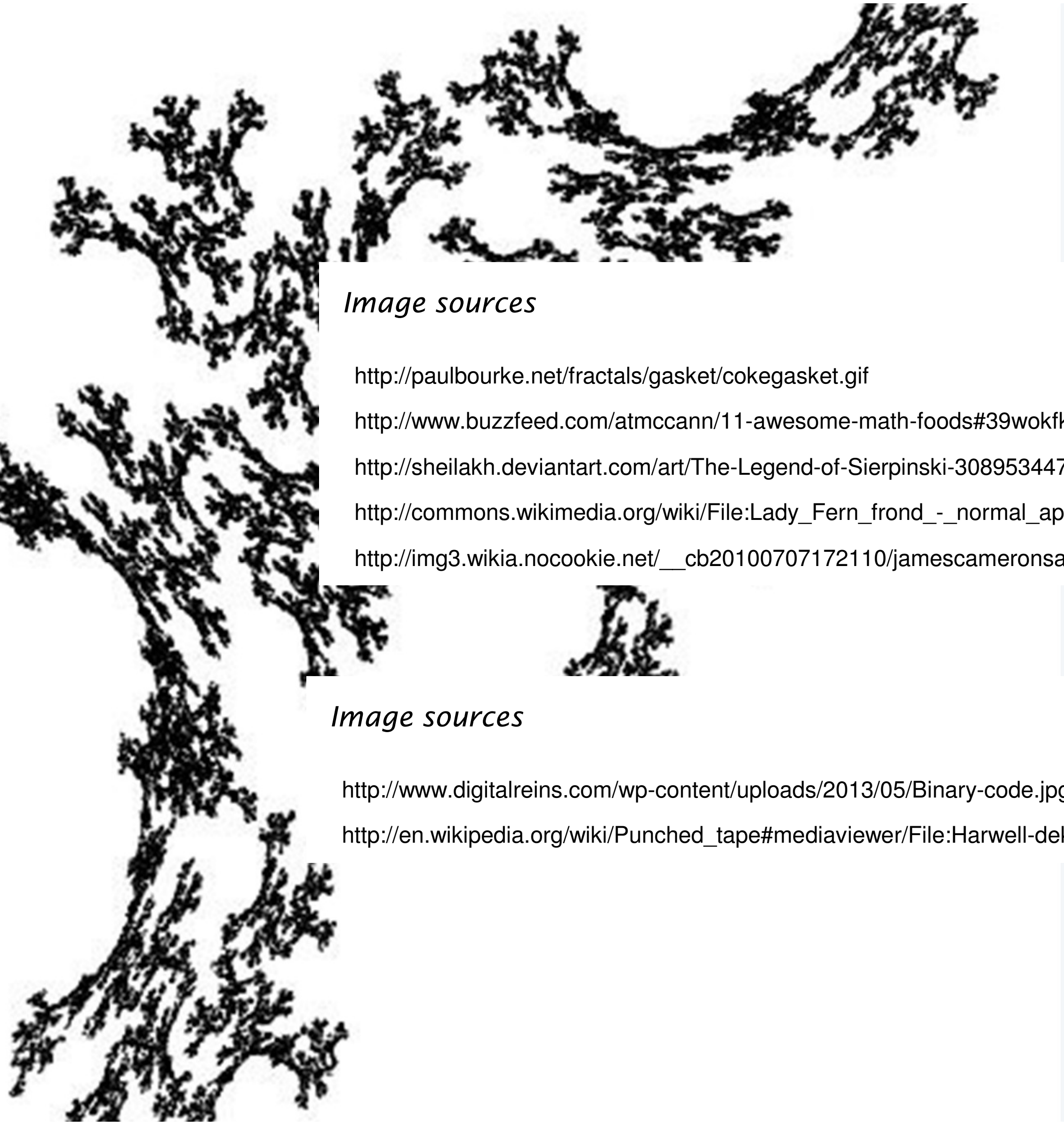


Image sources

http://paulbourke.net/fractals/gasket/cokegasket.gif

http://www.buzzfeed.com/atmccann/11-awesome-math-foods#39wokfk

http://sheilakh.deviantart.com/art/The-Legend-of-Sierpinski-308953447

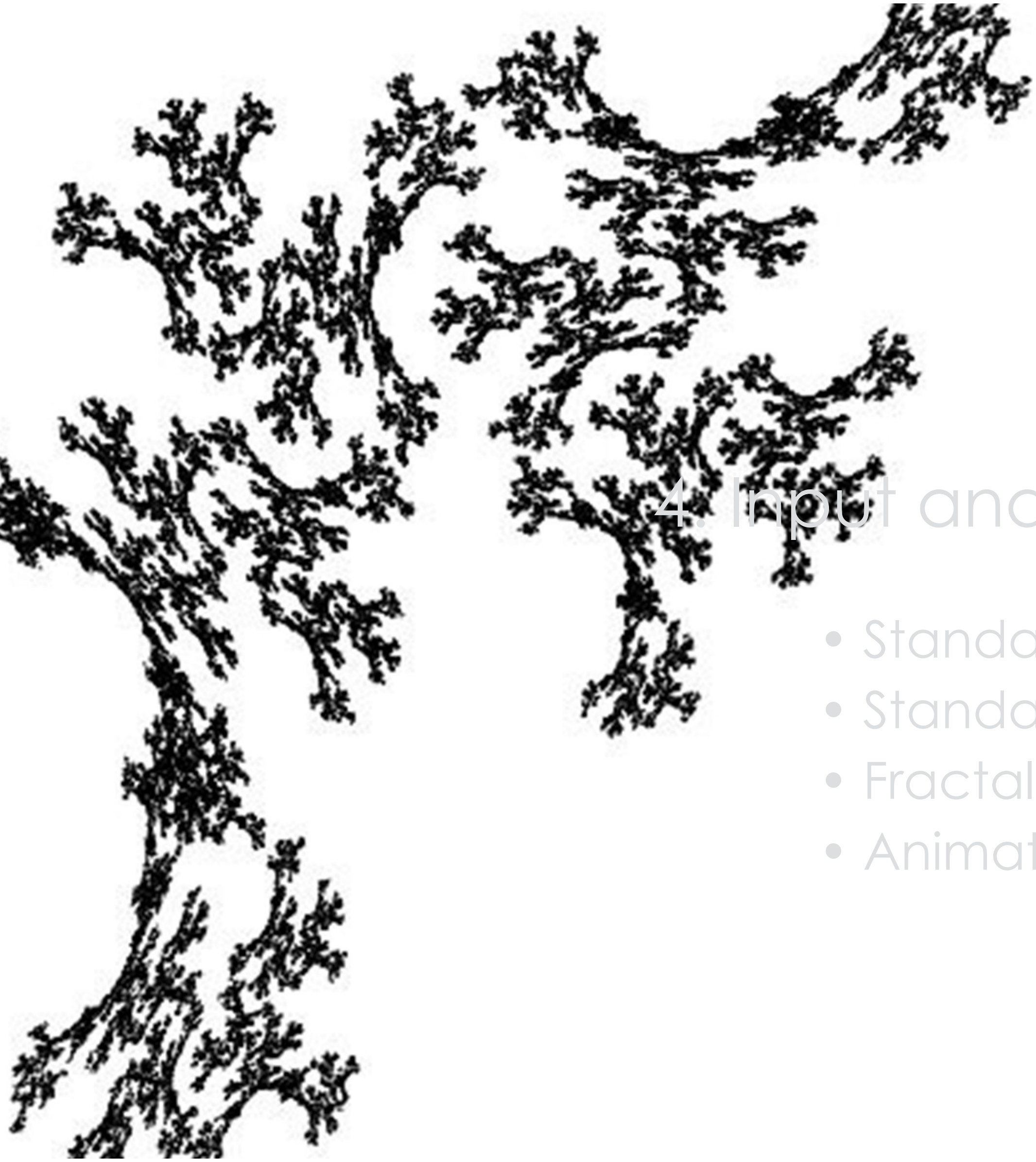
http://commons.wikimedia.org/wiki/File:Lady_Fern_frond_-_normal_appearance.jpg

http://img3.wikia.nocookie.net/__cb20100707172110/jamescameronsavatar/images/e/e1/Avatar_concept_art-3.jpg

Image sources

http://www.digitalreins.com/wp-content/uploads/2013/05/Binary-code.jpg

http://en.wikipedia.org/wiki/Punched_tape#mediaviewer/File:Harwell-dekatron-witch-10.jpg



INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

4. Input and Output

- Standard input and output
- Standard drawing
- Fractal drawings
- Animation

INTRODUCTION TO COMPUTER SCIENCE

Rutgers University

6. Input / Output