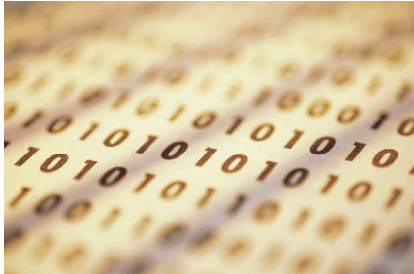


# Lecture 4: Input and Output



Today's goal: process huge amounts of data.

## Input and Output

### Input devices.



Keyboard



Mouse



Storage



Network



Digital camera



3D Scanner

### Output devices.



Display



Speakers



Storage



Network



Printer



MP3 Player

### Our approach.

- Define Java interfaces for input and output.
- Use operating system (OS) to connect Java programs to:
  - file system, each other, display

## Standard Output Abstraction

### Standard output.

- Flexible OS abstraction for output.
- In Java, output from `System.out.println` goes to `stdout`.
- By default, `stdout` is sent to Terminal window.
- Can save output in a file instead of printing to screen.
  - without changing Java program!

```
Terminal — tcsh (tty3)
[wayne:cycle] ~/intros> more HelloWorld.java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
[wayne:cycle] ~/intros> javac HelloWorld.java
[wayne:cycle] ~/intros> java HelloWorld
Hello, World
[wayne:cycle] ~/intros> java HelloWorld > output.txt
[wayne:cycle] ~/intros> more output.txt
Hello, World
[wayne:cycle] ~/intros>
```

```
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>
```

Terminal

## Standard Output

### Terminal output.

- Run program and print output to terminal window.

```
% java Random 4
90 84 75 83
```

### File output.

- Run program and use OS to *redirect* output to a file.

```
% java Random 4 > data.txt
% more data.txt
90 84 75 83
redirect stdout
```

```
public class Random {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 0; i < N; i++) {
            int r = (int) (Math.random() * 100);
            System.out.print(r + " ");
        }
        System.out.println();
    }
}
prints N random integers between 0 and 99
```

## Standard Input Abstraction

### 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.
- Java has built-in mechanisms for reading input from `stdin`.
- By default, `stdin` is received from Terminal window.
- Can read input from a file instead of typing at keyboard.
  - without changing Java program!
- Input entered *during* execution of program.
  - interactive input possible

5

## Standard Input

### Standard input.

- Java supports reading from `stdin`, but library is cumbersome.
- We provide simplified version in library `StdIn.java`.

```
public class Average {
    public static void main(String[] args) {
        double x, sum = 0.0;
        int N = 0;

        while (!StdIn.isEmpty()) {
            x = StdIn.readDouble();
            sum = sum + x;
            N++;
        }

        System.out.println(sum / N);
    }
}
```

6

## Standard Input

### Keyboard input.

- Run program and type data values in terminal, separated by whitespace.

```
% java Average
90
84
75
83
Ctrl-d  ← Unix EOF
85.543256
```

### File input.

- Redirect `stdin` to run program on data values stored in a file.

```
% more data.txt
90 84 75 83

% java Average < data.txt
85.543256
```

- Windows users: type `Ctrl-z` instead of `Ctrl-d`.

To execute, must have a copy of `StdIn.class` in current directory.

7

## Connecting Programs

### Brute force approach.

- Redirect output to one file, then
- Redirect input from that file.

```
% java Random 100000 > data.txt

% java Average < data.txt
49.36149
```

8

## Connecting Programs

### Pipes.

- OS abstraction to connect `stdout` of one command to `stdin` of another.
- Enables us to connect two different Java programs.
- Avoids creation of intermediate file `data.txt`.

```
% java Random 100 | java Average
50.24

% java Random 100000 | java Average
49.36149

% java Random 100000 | java Average
49.51199
```

← connect two different Java programs

```
% java Random 1000 | more
...
```

← connect one Java program with a built-in program to view results one screenful at a time

9

## "Standard Output" for Graphics

### Graphics output.

- Java supports graphical output, but library is cumbersome.
- We provide simplified library `StdDraw.java`, which can send output to the display or to a file.
- To use, put `StdDraw.java` and `Draw.java` in current directory.



10

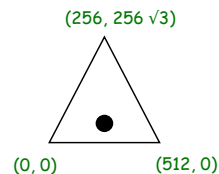
## Turtle Graphics

### Turtle graphics inspiration.

- Seymour Papert designed LOGO language to teach computing concepts to children.
- You command turtle to move, turn, and draw using **relative** coordinates.



```
StdDraw.penDown(); // put pen down
StdDraw.goForward(512); // forward 512
StdDraw.rotate(120); // rotate 120°
StdDraw.goForward(512); // forward 512
StdDraw.rotate(120); // rotate 120°
StdDraw.goForward(512); // forward 512
StdDraw.rotate(120); // rotate 120°
```



- Or fly using **absolute** coordinates, dropping colored spots from above.

```
StdDraw.penUp(); // put pen up
StdDraw.go(256, 200); // go to (256, 200)
StdDraw.spot(80); // drop spot of diameter 80
```

11

## Data Analysis

### Plotting points.

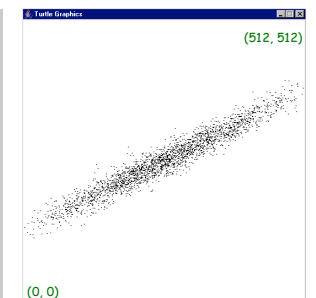
- Read in a sequence of  $(x, y)$  coordinates.
- Plot using our graphics library.
- Basis for data analysis.

2,500 pairs of real numbers



```
% java Plot < data.txt
```

```
public class Plot {
    public static void main(String args[]) {
        StdDraw.create(512, 512);
        while (!StdIn.isEmpty()) {
            double x = StdIn.readDouble();
            double y = StdIn.readDouble();
            StdDraw.go(x, y);
            StdDraw.spot(3);
        }
        StdDraw.show();
    }
}
```



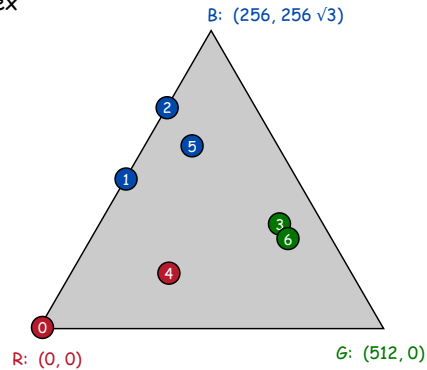
12

## Chaos Game

Game played 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 "dot" in color of vertex

Q. What picture emerges?



13

## Chaos Game

```
public class Chaos {
    public static void main(String args[]) {
        int N = Integer.parseInt(args[0]);           # dots
        double size = Double.parseDouble(args[1]);   diameter
        double x = 0.0, y = 0.0;
        double x0, y0;
        StdDraw.create(512, 512);

        for (int i = 0; i < N; i++) {                plot N points
            double r = Math.random();                pick random vertex
            if (r < 0.333) { x0 = 0.0; y0 = 0.0; }
            else if (r < 0.667) { x0 = 512.0; y0 = 0.0; }
            else { x0 = 256.0; y0 = 443.4; }
            x = (x0 + x) / 2;                         ← move halfway
            y = (y0 + y) / 2;                         ← 256√3
            StdDraw.go(x, y);                          (usually best to avoid "hardwired" constants)
            StdDraw.spot(size);
            StdDraw.pause(10);                         ← take a break for 10ms and display
        }
    }
}
```

14

## Barnsley Fern

Barnsley fern: chaos game with different rules.

Probability	New x	New y
2%	0.5	0.222x + 0.176y + 0.0893
5%	-0.139x + 0.263y + 0.57	0.246x + 0.224y - 0.036
13%	0.170x - 0.215y + 0.408	0.222x + 0.176y + 0.0893
70%	0.781x + 0.034y + 0.1075	-0.032x + 0.739y + 0.27

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



17

## Saving Graphics to a File

To produce an image file:

- Include drawing command as usual.
- For PNG: StdDraw.save("filename.png")
- For JPEG: StdDraw.save("filename.jpg")

Portable Network Graphics (PNG).

- Standard Web image format.
- Excellent for line art.

Joint Photographic Experts Group (JPEG).

- Compressed file format.
- Excellent for photographic images.

18

## Animation

### Animation loop.

- Move object.
- Draw object.
- Pause for a short while and display.
- Repeat.

### Example: bouncing ball.

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

19

## Bouncing Ball

```
import java.awt.Color; // needed to access color library

public class BouncingBall {
    public static void main(String[] args) {
        double rx = 0.480, ry = 0.860; // position
        double vx = 0.015, vy = 0.023; // velocity

        StdDraw.create(512, 512);
        StdDraw.setScale(-1.0, -1.0, 1.0, 1.0); // change coordinate system

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

            rx = rx + vx; // update position
            ry = ry + vy; // update position

            StdDraw.clear(Color.gray); // clear background to gray
            StdDraw.go(rx, ry); // draw image, centered on (rx, ry)
            StdDraw.spot(0.025); // pause for 50ms and display
            StdDraw.pause(50); // pause for 50ms and display
        }
    }
}
```

20

## Images and Sound Effects

### Images.

- Put .gif, .png, or .jpg file in same directory as Java source file.
- Use `StdDraw.spot` to draw it.

### Sound effects.

- Put .wav, .mid, or .au file in same directory as Java source file.
- Use `StdDraw.play` to play it.

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

- Replace `StdDraw.spot(0.025)` with:

```
StdDraw.spot("earth.gif");
```

- Add following code when collision detected:

```
StdDraw.play("laser.wav");
```

21

## User Interfaces

### Command line interface.

- User types commands at terminal.
- Easily customizable.
- Extends to complex command sequences.

### Point and click.

- User launches applications by clicking.
  - File → Open → HelloWorld.java
- Restricted to pre-packaged menu options.



### See "In the Beginning was the Command Line" by Neal Stephenson.

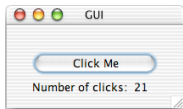
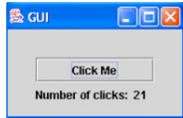
- <http://www.spack.org/words/commandline.html>

22

## Swing Graphical User Interface

"Swing" is Java's GUI.

- Buttons.
- Menus.
- Scrollbars.
- Toolbars.
- File choosers.



```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class GUI extends JFrame implements ActionListener {
    private int clicks = 0;
    private JLabel label = new JLabel("Number of clicks: 0 ");

    public GUI() {
        JButton button = new JButton("Click Me");
        button.addActionListener(this);
        JPanel panel = new JPanel();
        panel.setBorder(BorderFactory.createEmptyBorder(9, 9, 9, 9));
        panel.setLayout(new GridLayout(0, 1));
        panel.add(button);
        panel.add(label);
        getContentPane().add(panel, BorderLayout.CENTER);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("GUI");
        pack();
        show();
    }

    public void actionPerformed(ActionEvent e) {
        clicks++;
        label.setText("Number of clicks: " + clicks);
    }

    public static void main(String[] args) {
        GUI gui = new GUI();
    }
}
```

A sample Swing application

Don't worry about details now!