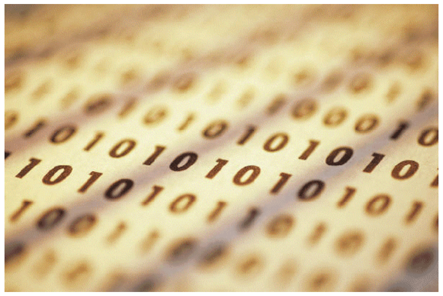
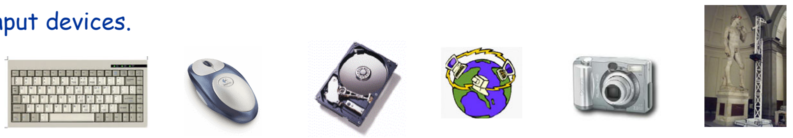
2.4 Input and Output



Today's goal: process huge amounts of data.

Input and Output

Input devices.

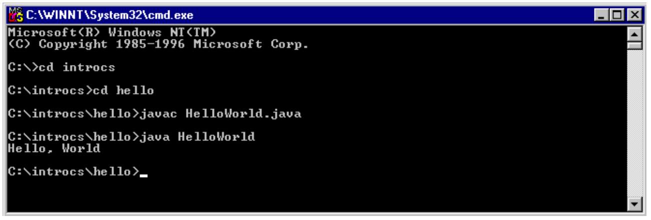


Output devices.

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!



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>cd introcs

C:\introcs>cd hello

C:\introcs\hello>javac HelloWorld.java

C:\introcs\hello>java HelloWorld
Hello, World

C:\introcs\hello>_
```

Terminal

Standard Output

```
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();
   }
}
```

command line input

prints N random integers between 0 and 99

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

Standard Input Abstraction

Command line inputs.

- Use command line inputs to read in a few user values.
- Not practical for many user inputs.

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!

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 += x;
         N++;
      }

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

Standard Input

Keyboard input.

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

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

Ctrl-d ⇐ Unix EOF

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.

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

"Standard Output" for Graphics

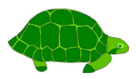
We want analog of standard output for pictures.

- Java support graphics.
- We define our own abstractions to simplify things.
  - output to display
  - output to `stdout` in JPEG format
  - output to `stdout` in PNG format

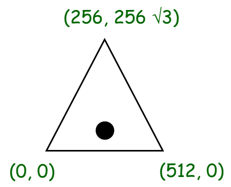
Turtle Graphics

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.



```
Turtle.forward(512);   // forward 512
Turtle.rotate(120);    // rotate 120°
Turtle.forward(512);   // forward 512
Turtle.rotate(120);    // rotate 120°
Turtle.forward(512);   // forward 512
Turtle.rotate(120);    // rotate 120°
```



- Or to fly to absolute coordinates and drop colored spots below.

```
Turtle.fly(256, 200);  // go to (256, 200)
Turtle.spot(80);       // drop spot of diameter 80
```

Data Analysis

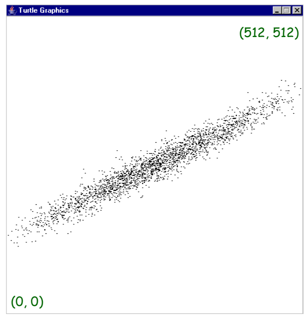
Plotting points.

- Read in a sequence of (x, y) coordinates.
- Plot using Turtle graphics.

2,500 pairs of real numbers

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

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

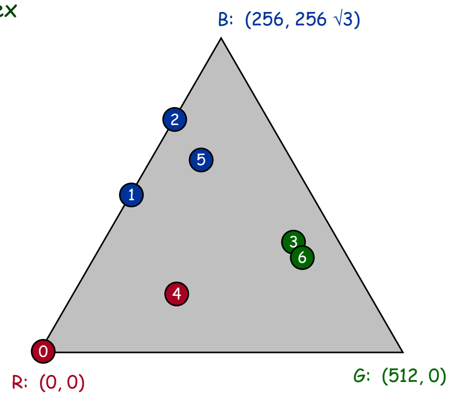


Chaos Game

Game played on equilateral triangle, with vertices R, G, B.

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

Q. What picture emerges?



Chaos Game

```
public class Chaos {
   public static void main(String args[]) {
      int N = Integer.parseInt(args[0]);
      double size = Double.parseDouble(args[1]);
      double x = 0.0, y = 0.0;
      double x0, y0;
      Turtle.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                ↖ 256 √3
         y = (y0 + y) / 2;
         Turtle.fly(x, y);        (usually best to avoid "hardwired" constants)
         Turtle.spot(size);
         Turtle.pause(10);
      }

      Turtle.destroy();
   }
}
```

Saving Turtle Graphics to a File

To produce a portable network graphics (PNG) image file:

- Compile `TurtlePNG.java` to replace `Turtle.class`.
- Output goes to `stdout` instead of display.
- Use redirection to save in a file.

```
% javac TurtlePNG.java

% java Chaos 10000 5 > chaos.png
```

Other implementations of `Turtle`:

- Use `TurtleJPEG.java` to produce JPEG files.
- Use `TurtleEPS.java` to produce PostScript.

Note: client must call `Turtle.destroy` when done, or no output.

Animation

Animation loop.

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

Example: bouncing ball.

- Ball has position (`px`, `py`) and velocity (`vx`, `vy`).
- Detect collision with wall and reverse velocity.

Bouncing Ball

```
import java.awt.Color;   ← needed for Color.black

public class BouncingBall {
   public static void main(String[] args) {
      double px = 48.0, py = 120.0;   ← initial position
      double vx =  7.0, vy =   3.7;   ← velocity
      Turtle.create(512, 512);

      while(true) {
         if ((px + vx > 512.0) || (px + vx < 0.0)) vx = -vx;
         if ((py + vy > 512.0) || (py + vy < 0.0)) vy = -vy;
                                                     ↑ bounce
         px += vx;   ← update position
         py += vy;

         Turtle.clear(Color.black);   ← clear background
         Turtle.fly(px, py);
         Turtle.spot(10);             ← draw image
         Turtle.pause(50);            ← pause for 50ms and display
      }
   }
}
```

Images and Sound Effects

Images.

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

Sound effects.

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

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

- Replace `Turtle.spot(10)` with:

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

- Add following code when collision detected:

```
Turtle.grunt("laser.wav");
```

Saving Turtle Graphics to a Movie

To produce a multi-image network graphics (MNG) movie file:

- Write the library `TurtleMNG.java`.
- Substitute this implementation for `Turtle`.

Other non-existing implementations of `Turtle`:

- Use `TurtleQT.java` to produce QuickTime movies.
- Use `TurtleMPEG4.java` to produce MPEG4 videos.

Moral.

- Having access to nice libraries is useful.
- Having a flexible interface is useful.

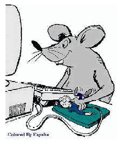
User Interfaces

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.



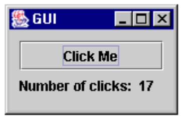
See "In the Beginning was the Command Line" by Neal Stephenson.

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

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 for now.