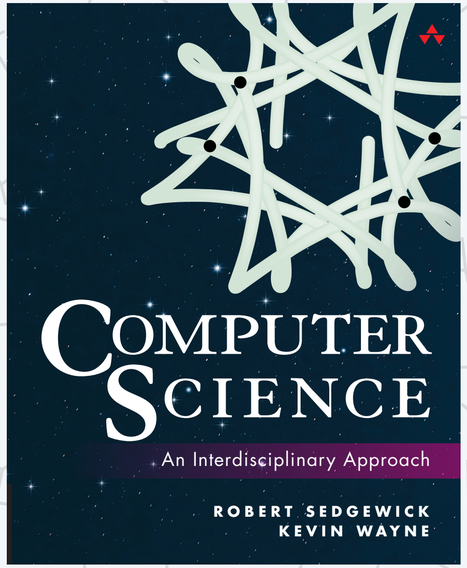
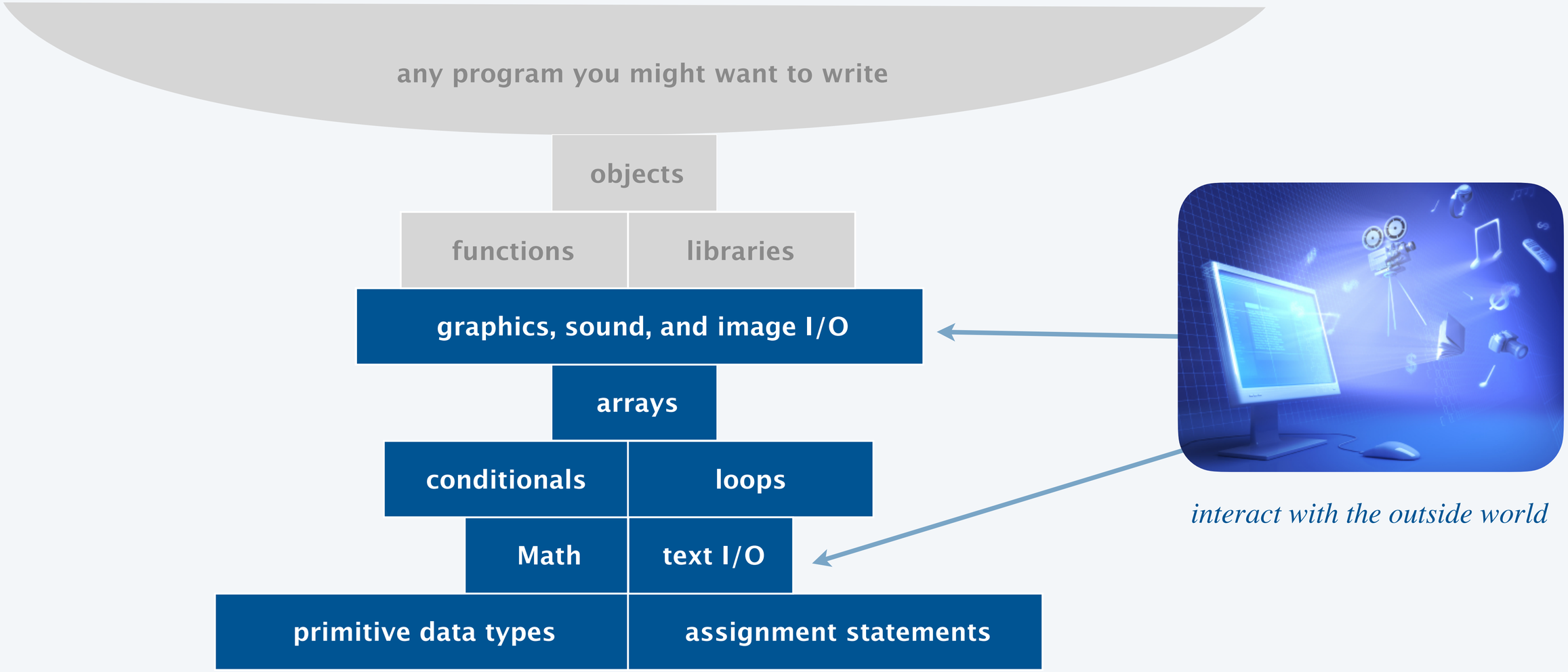


https://introcs.cs.princeton.edu

1.5 Input and Output

- *standard input and output*
- *redirection and piping*
- *standard drawing*
- *animation*

Basic building blocks for programming



Input and output

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

Input devices.

keyboard

trackpad

storage

network

webcam

microphone

Output devices.

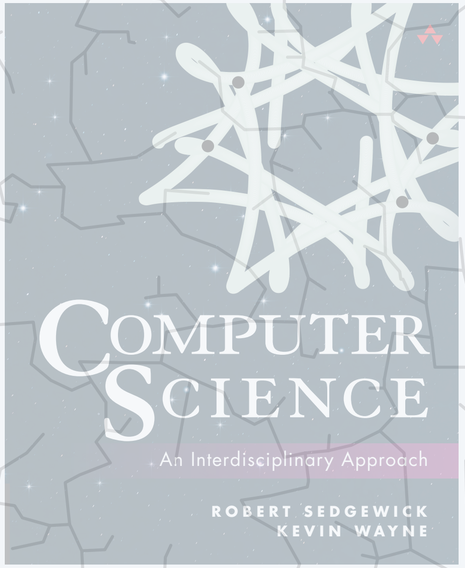
video display

earbuds

storage

network

braille display



https://introcs.cs.princeton.edu

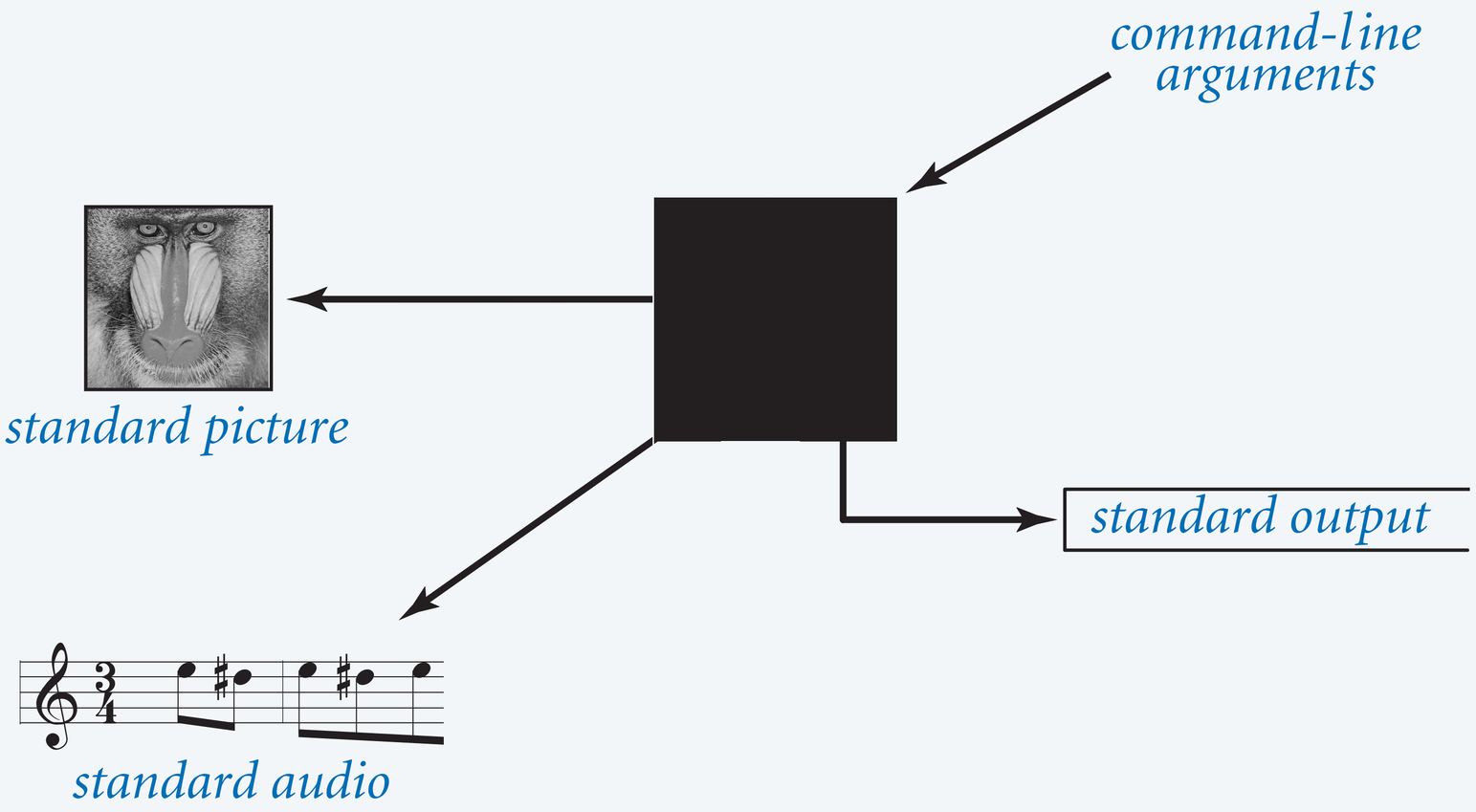
1.5 Input and Output

- *standard input and output*
- *redirection and piping*
- *standard drawing*
- *animation*

Input–output abstractions (so far)

Our approach.

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



Review: terminal

Terminal. A text-based interface for interacting with programs, files, and devices.

Hello, World – HelloGoodbye.java

Project

hello [Computer Science] sources ro
- HelloGoodbye
- logo.png
- External Libraries
- Scratches and Consoles

HelloGoodbye.java

```
1  public class HelloGoodbye {
2      public static void main(String[] args) {
3          System.out.println("Hello " + args[0] + ".");
4          System.out.println("Goodbye " + args[1] + ".");
5      }
6  }
7
```

Terminal: Terminal

```
~/Desktop/hello> java HelloGoodbye Kevin Alan
Hello Kevin.
Goodbye Alan.
~/Desktop/hello> java HelloGoodbye Arya Zahara
Hello Arya.
Goodbye Zahara.
~/Desktop/hello>
```

output to standard output

input from command line

Terminal

Build completed successfully in 1 sec, 310 ms (3 minutes ago) 7:1 LF UTF-8 4 spaces

VT-100 terminal emulator

Review: command-line arguments

Command-line arguments. Provide text input to a program.

Basic properties.

- Arguments provided to a program by typing after program name.
- Arguments provided to program *before* execution.
- Java: string arguments available in *main*() as *args*[0], *args*[1], …

```java
public class HelloGoodbye {
   public static void main(String[] args) {
      System.out.print("Hello ");
      System.out.println(args[0] + ".");
      System.out.print("Goodbye ");
      System.out.println(args[1] + ".");
   }
}
```

```
~/cos126/io> java HelloGoodbye Kevin Alan
Hello Kevin.
Goodbye Alan.

~/cos126/io> java HelloGoodbye Arya Zahara
Hello Arya.
Goodbye Zahara.

~/cos126/io> java HelloGoodbye Aðalbjörg "Hua Fei"
Hello Aðalbjörg.
Goodbye Hua Fei.
```

command-line arguments

args[0] *args[1]*

use quotes

Review: standard output

Standard output stream. An abstraction for an output sequence of text.

Basic properties.

- The call *System.out.println()* appends text to the standard output stream.
- By default, the standard output stream is connected to the terminal.
- No limit on amount of output.

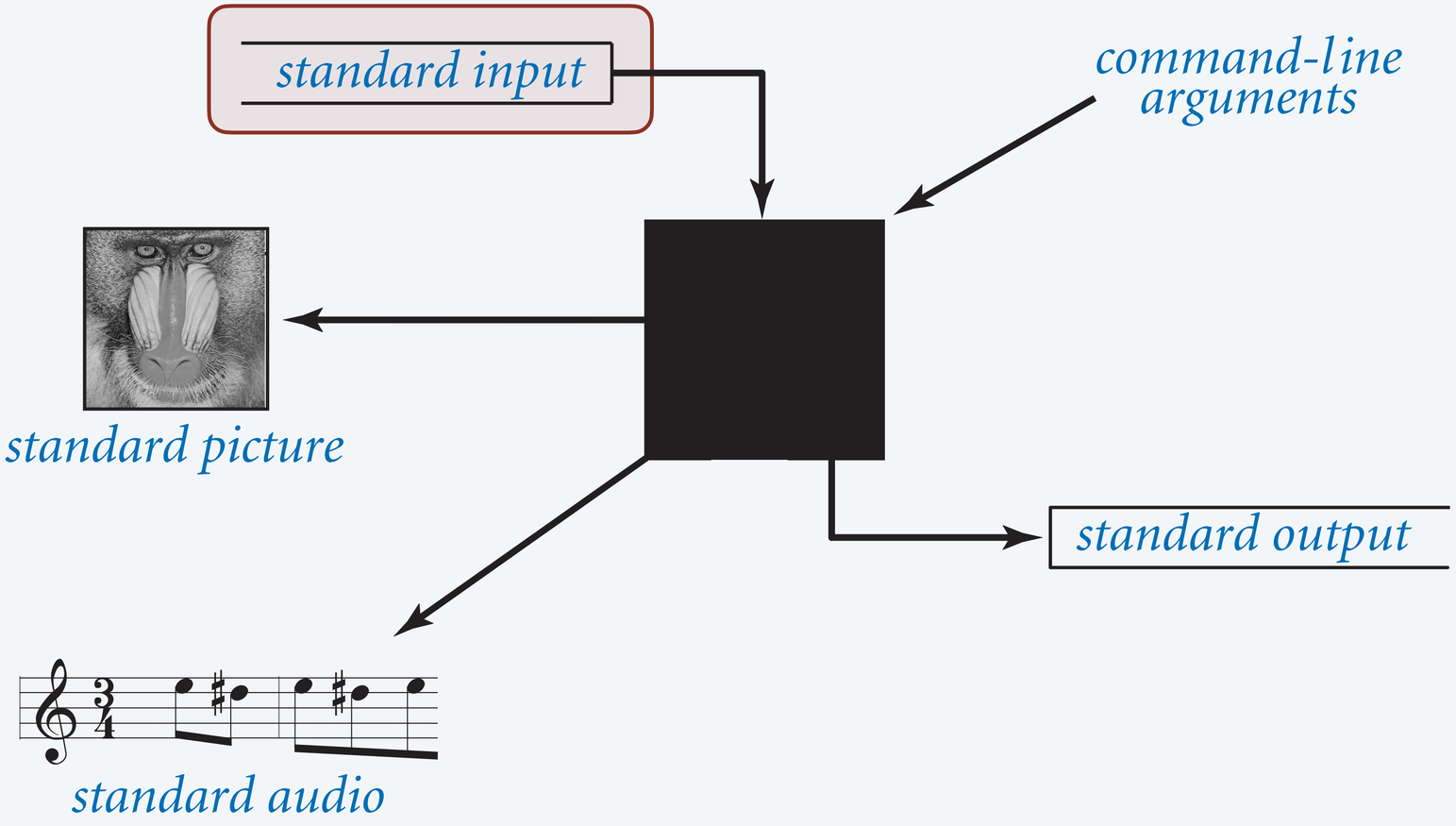
```java
public class RandomUniform {
   public static void main(String[] args) {
      int n = Integer.parseInt(args[0]);
      for (int i = 0; i < n; i++) {
         System.out.println(Math.random());
      }
   }
}
```

```
~/cos126/io> java RandomUniform 4
0.9320744627218469
0.4279508713950715
0.08994615071160994
0.6579792663546435

~/cos126/io> java RandomUniform 1000000
0.09474882292442943
0.2832974030384712
0.1833964252856476
0.2952177517730442
0.8035985765979008
...
```

produces lots of output

Input–output abstractions (standard input)

Next step. Add a text input stream.

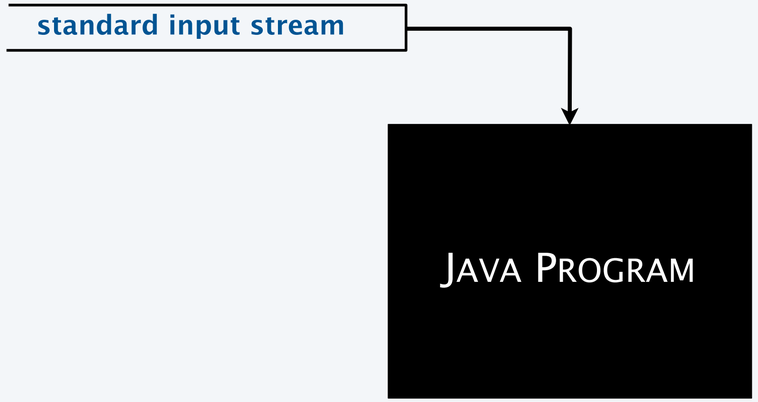


Standard input

Standard input stream. An abstraction for an input sequence of text.

Advantages over command-line arguments:

- No limit on the amount of input.
- Conversion to primitive types is explicitly handled.
- Can provide input interactively, *while* the program is executing.



Standard input library

StdIn. Our library for reading strings and numbers from standard input.

available with `javac-introcs` *and* `java-introcs` *commands*

`public class StdIn`		**description**
`static boolean`	`isEmpty()`	`true` *if no more values,* `false` *otherwise*
`static int`	`readInt()`	*read a value of type* `int`
`static double`	`readDouble()`	*read a value of type* `double`
`static boolean`	`readBoolean()`	*read a value of type* `boolean`
`static String`	`readString()`	*read a value of type* `String`
	⋮	⋮

Standard output library

StdOut. Our library for printing strings and numbers to standard output.

available with `javac-introcs` *and* `java-introcs` *commands*

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

Q. How different from *`System.out.println()`* ?

A. Mostly the same, but output is independent of system and locale.

we'll use `StdOut` *from now on*

Standard input warmup

Interactive user input. User can provide input while the program is running.

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

```
~/cos126/io> java-introcs AddTwoInts
Type the first integer: 1
Type the second integer: 2
Their sum is 3

~/cos126/io> java-introcs AddTwoInts
Type the first integer: 100
Type the second integer: 26
Their sum is 126

~/cos126/io> java-introcs AddTwoInts
Type the first integer: 100
Type the second integer: twenty-six
java.util.InputMismatchException: attempts
to read an ‘int' value from standard input,
but the next token is "twenty-six"
```

Remark 1. By default, standard input stream comes from terminal.

Remark 2. Input and output can be interleaved.

Remark 3. Run-time exception if user enters incompatible input.

Average the numbers on the standard input stream

Goal. Read a stream of numbers (from standard input) and print their average (to standard output).

```java
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);
   }
}
```

```
~/cos126/io> java-introcs Average
1.0
2.0
4.0
2.0
<Ctrl-D>
2.25

~/cos126/io> java-introcs Average
10.0 5.0 6.0 3.0
7.0      32.0
<Ctrl-D>
10.5
```

signifies end of standard input (<Ctrl–Z><Enter> on Windows)

values separated by whitespace

Remark. No limit on amount of input. ← *"streaming algorithm" (avoids storing data)*

Input and output: quiz 1

What does the following program do with the given input?

A. Prints "X", "Y", and "Z".

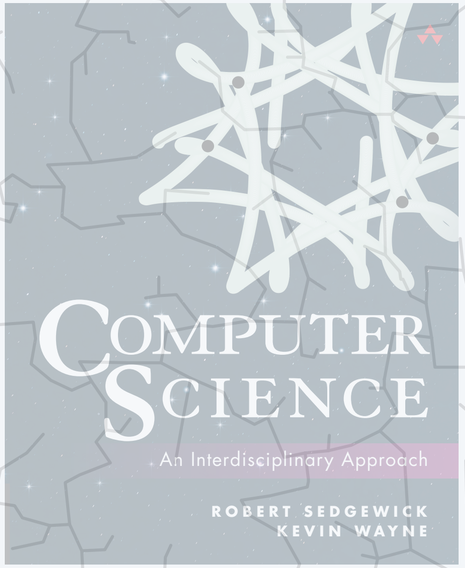
B. Throws an exception.

C. Both A and B.

D. Neither A nor B.

```java
public class Mystery {
   public static void main(String[] args) {
      int n = args.length;
      for (int i = 0; i < n; i++) {
         String s = StdIn.readString();
         StdOut.println(s);
      }
   }
}
```

```
~/cos126/io> java-introcs Mystery A B C D E
X Y Z
<Ctrl-D>
```



https://introcs.cs.princeton.edu

1.5 Input and Output

- *standard input and output*
- *redirection and piping*
- *standard drawing*
- *animation*

Redirecting standard output

Terminal. By default, standard output is connected to the terminal.

Redirecting standard output. Send standard output to a file (instead of the terminal).


```
~/cos126/io> java-introcs RandomUniform 1000000 > data.txt
[no output]

~/cos126/io> more data.txt
0.09474882292442943
0.2832974030384712
0.1833964252856476
0.2952177517730442
0.8035985765979008
...
```

redirect standard output

filename

display content of a file

Redirecting standard input

Terminal. By default, standard input is connected to the terminal.

Redirecting standard input. Read standard input from a file (instead of the terminal).

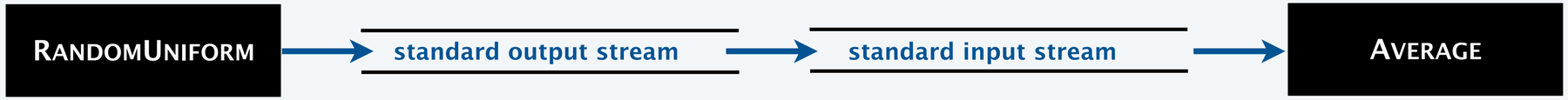

```
~/cos126/io> more data.txt
0.09474882292442943
0.2832974030384712
0.1833964252856476
0.2952177517730442
0.8035985765979008
...

~/cos126/io> java-introcs Average < data.txt
0.4947655567740991
```

redirect standard input

filename

Piping

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




```
~/cos126/io> java-introcs RandomUniform | java-introcs Average
0.4997970473016028

~/cos126/io> java-introcs RandomUniform | java-introcs Average
0.5002071875644842
```

pipe operator

Remark. No limit within programs on amount of data to process.

Input and output: quiz 2

The OS X command say reads text from standard input and synthesizes it as audible speech. Which of the following commands will speak "Hello, World" ?

A.

```
~/cos126/io> say
HelloWorld.java
<Ctrl-D>
```

B.

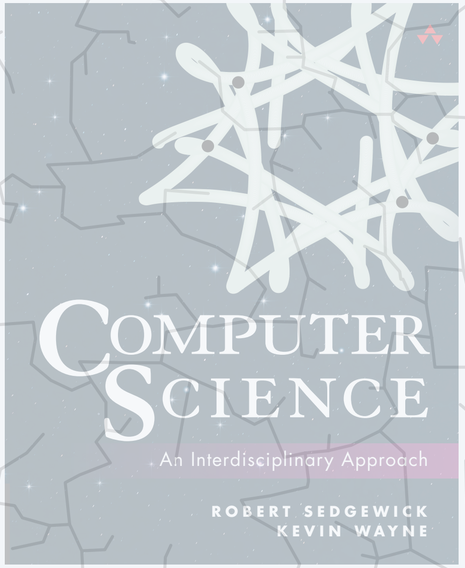
```
~/cos126/io> say > HelloWorld.java
```

C.

```
~/cos126/io> java HelloWorld | say
```

D.

```
~/cos126/io> say < HelloWorld.java
```



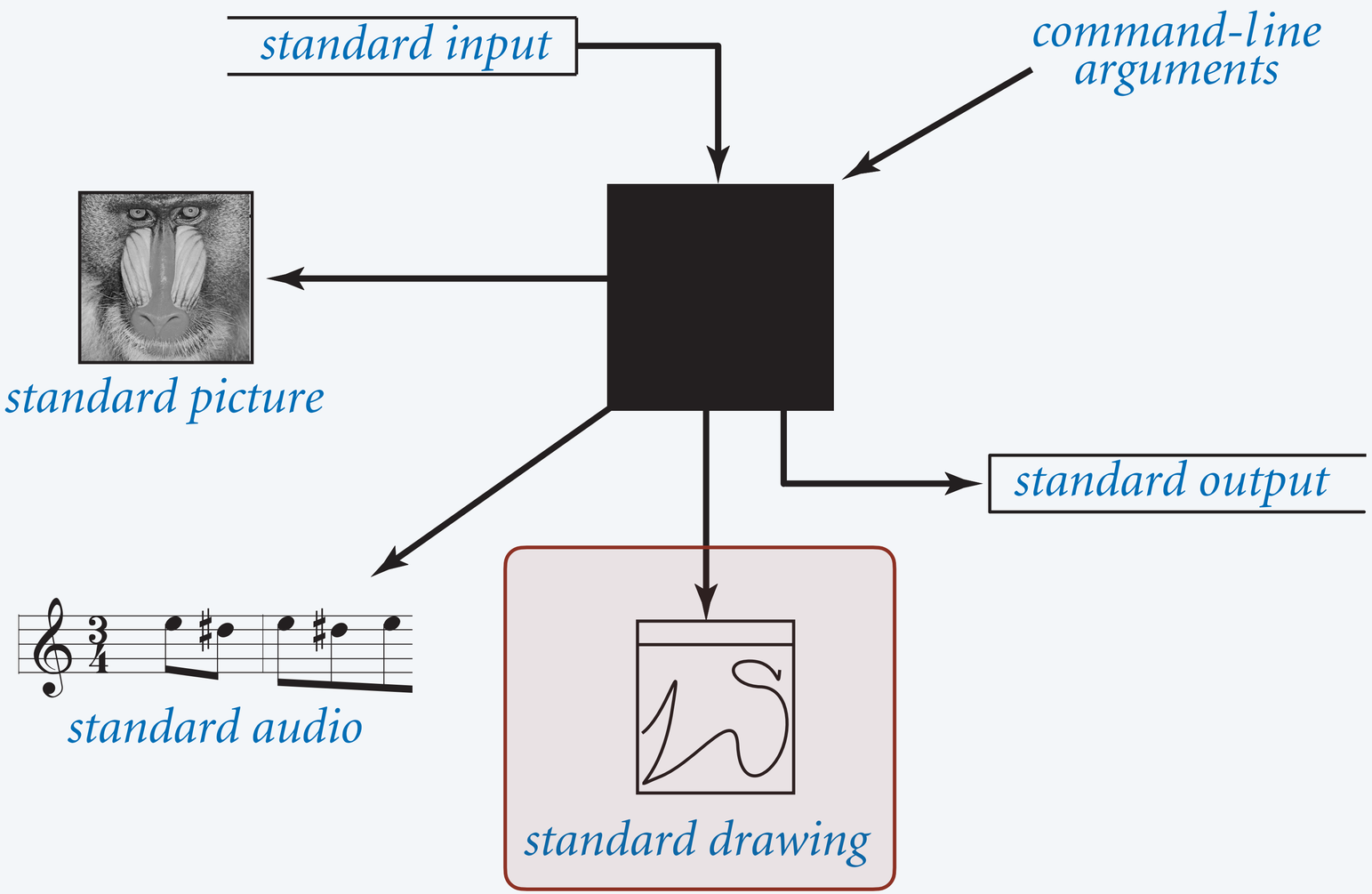
https://introcs.cs.princeton.edu

1.5 Input and Output

- *standard input and output*
- *redirection and piping*
- *standard drawing*
- *animation*

Input–output abstractions (standard drawing)

Next step. Add the ability to create a drawing.



Standard drawing library: drawing methods

StdDraw. Our library for drawing and animating geometric shapes in a graphical window.

to manipulate images, use `StdPicture` *library*

available with `javac-introcs` *and* `java-introcs` *commands*

`public class StdDraw`	**description**
`static void line(double x0, double y0, double x1, double y1)`	*draw line segment between* (x_0, y_0) *and* (x_1, y_1)
`static void point(double x, double y)`	*draw point* (x, y)
`static void circle(double x, double y, double r)`	*draw circle of radius r centered at* (x, y)
`static void square(double x, double y, double r)`	*draw square of half-width r centered at* (x, y)
`static void polygon(double[] x, double[] y)`	*draw polygon connecting points* (x_i, y_i)
`static void text(double x, double y, String text)`	*draw text, centered at* (x, y)
`static void picture(double x, double y, String filename)`	*draw GIF, JPG or PNG image, centered at* (x, y)
⋮	⋮

Standard drawing library: control methods

StdDraw. Our library for drawing and animating geometric shapes in a graphical window.

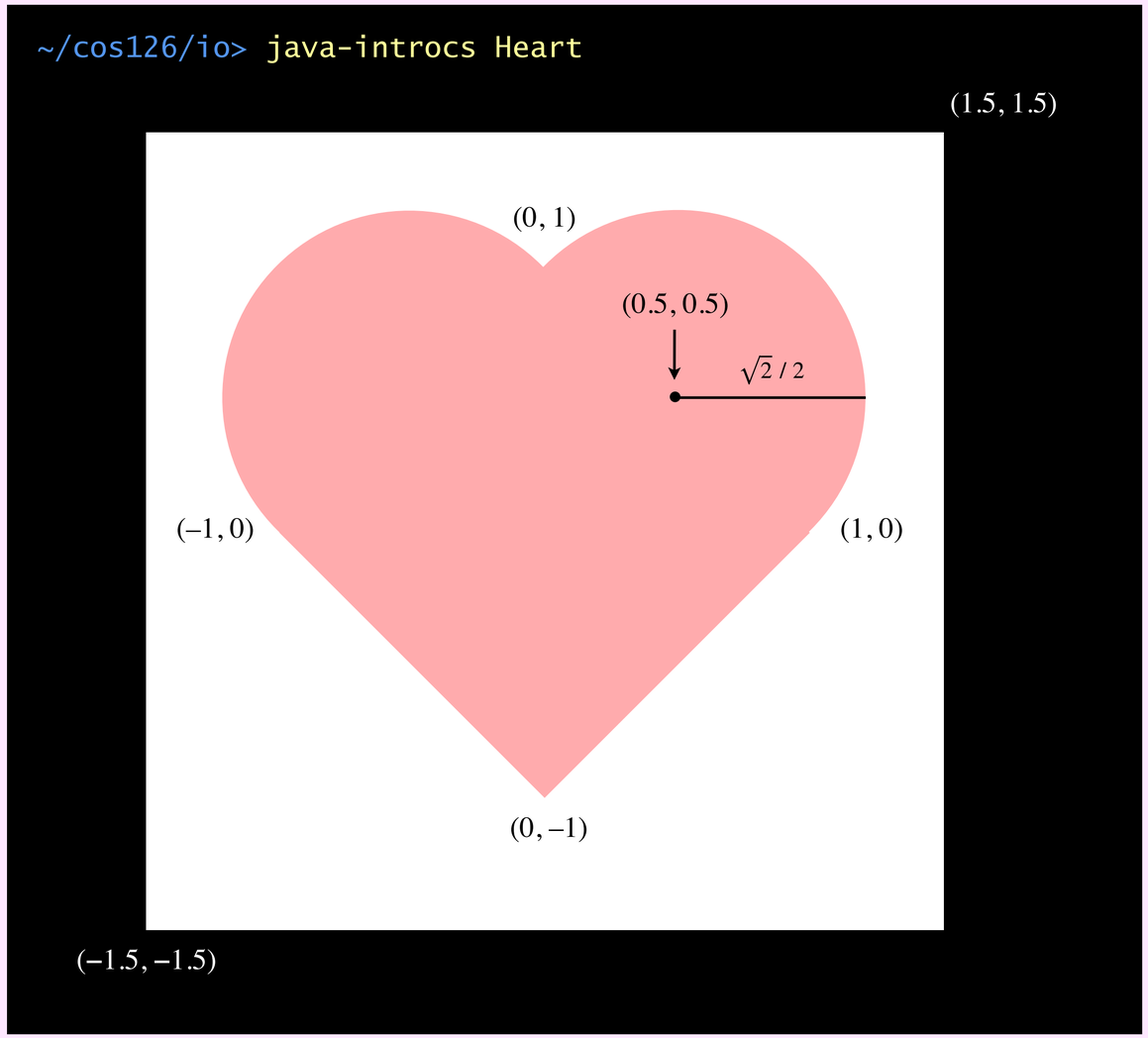
public class StdDraw	description	default value
static void setCanvasSize(int width, int height)	*set the canvas size to* width-*by*-height	512-*by*-512
static void setXscale(double x0, double x1)	*set x-range to* $[x_0, x_1]$	$[0, 1]$
static void setYscale(double y0, double y1)	*set y-range to* $[y_0, y_1]$	$[0, 1]$
static void setPenRadius(double radius)	*set the pen radius to* radius	0.002
static void setPenColor(Color color)	*set the pen color to* color	*black*
⋮	⋮	

StdDraw.BLACK, StdDraw.WHITE,
StdDraw.GRAY, StdDraw.RED,
StdDraw.GREEN, StdDraw.BLUE,
StdDraw.PRINCETON_ORANGE, ...

Your first drawing

Goal. Draw filled diamond and two filled circles.

```java
public class Heart {
    public static void main(String[] args) {
        StdDraw.setXscale(-1.5, +1.5);
        StdDraw.setYscale(-1.5, +1.5);
        StdDraw.setPenColor(StdDraw.PINK);

        // draw filled diamond
        double[] xs = { -1,  0, 1, 0 };
        double[] ys = {  0, -1, 0, 1 };
        StdDraw.filledPolygon(xs, ys);

        // draw two filled circles
        double radius = Math.sqrt(2) / 2;
        StdDraw.filledCircle(+0.5, 0.5, radius);
        StdDraw.filledCircle(-0.5, 0.5, radius);
    }

}
```



trace of drawing

Data visualization

Goal. Read points (from standard input) and plot.

```java
public class PlotPoints {
   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);
      }
   }
}
```

bounding box

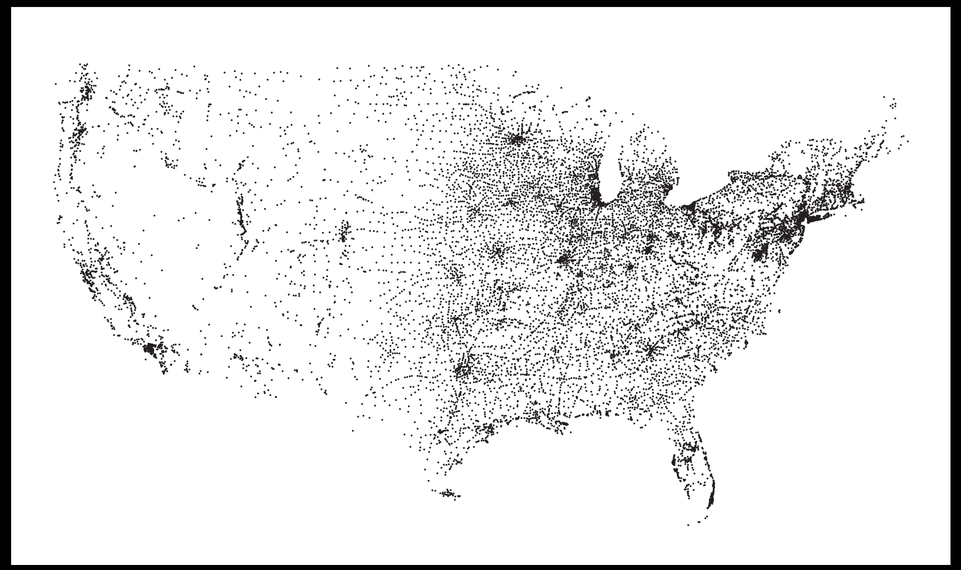
rescale

read points and plot

```
~/cos126/io> more USA.txt
669905.0 247205.0 1244962.0 700000.0
 1097038.8890  245552.7780
 1103961.1110  247133.3330
 1104677.7780  247205.5560
 ...

~/cos126/io> java-introcs PlotPoints < USA.txt
```

bounding box

sequence of points
(13, 509 USA cities)



(699905, 247205)

Data visualization

Goal. Read points (from standard input) and plot.

```java
public class PlotPoints {
   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);
      }
   }
}
```

bounding box

rescale

read points and plot

```
~/cos126/io> more StarryNight.txt
669905.0 247205.0 1244962.0 700000.0
 1097038.8890  245552.7780
 1103961.1110  247133.3330
 1104677.7780  247205.5560
 ...

~/cos126/io> java-introcs PlotPoints < StarryNight.txt
```

bounding box

sequence of points (223,534 dots)

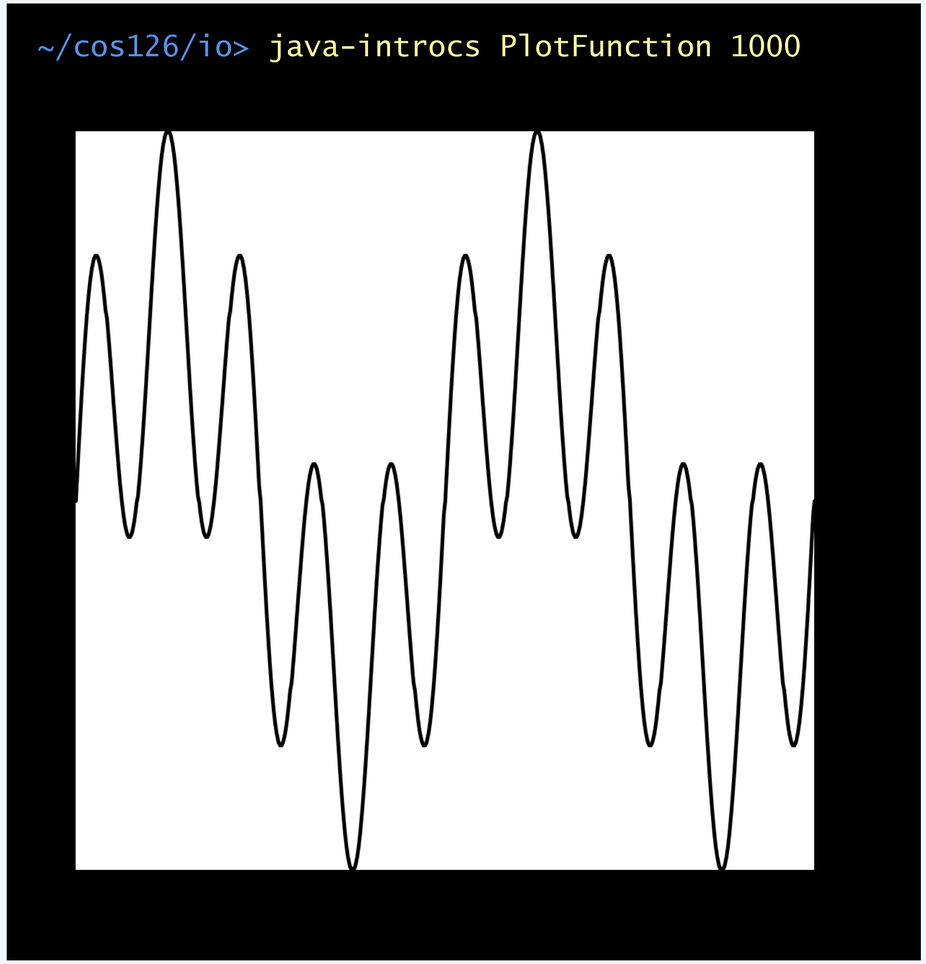
Plotting a function

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

Method. Take $n + 1$ samples, evenly spaced in interval.

how many samples is enough?

```java
public class PlotFunction {
   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]);

   }
}
```

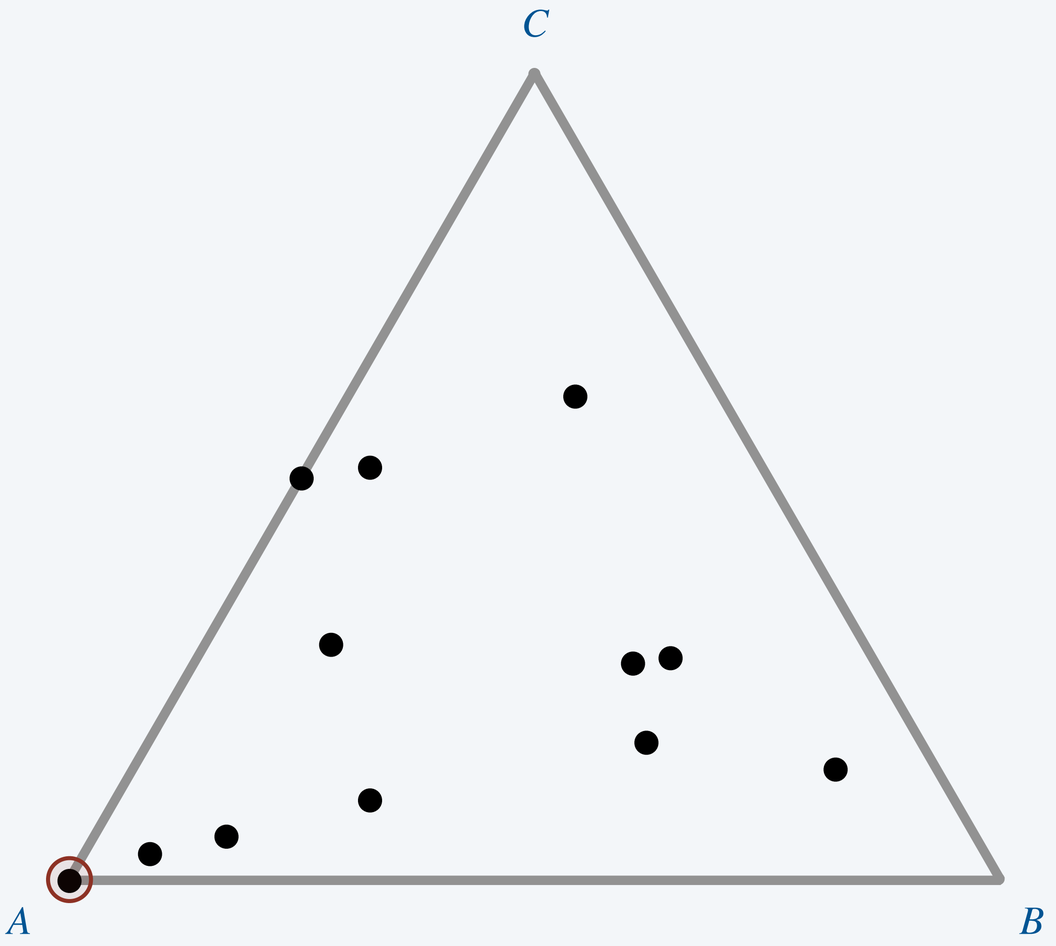


The chaos game

Chaos game. Draw an equilateral triangle; make one vertex the current point.

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

i	vertex
0	C
1	B
2	C
3	A
4	B
5	A
6	A
7	A
8	C
9	B
10	B
…	…



Q. What figure emerges?

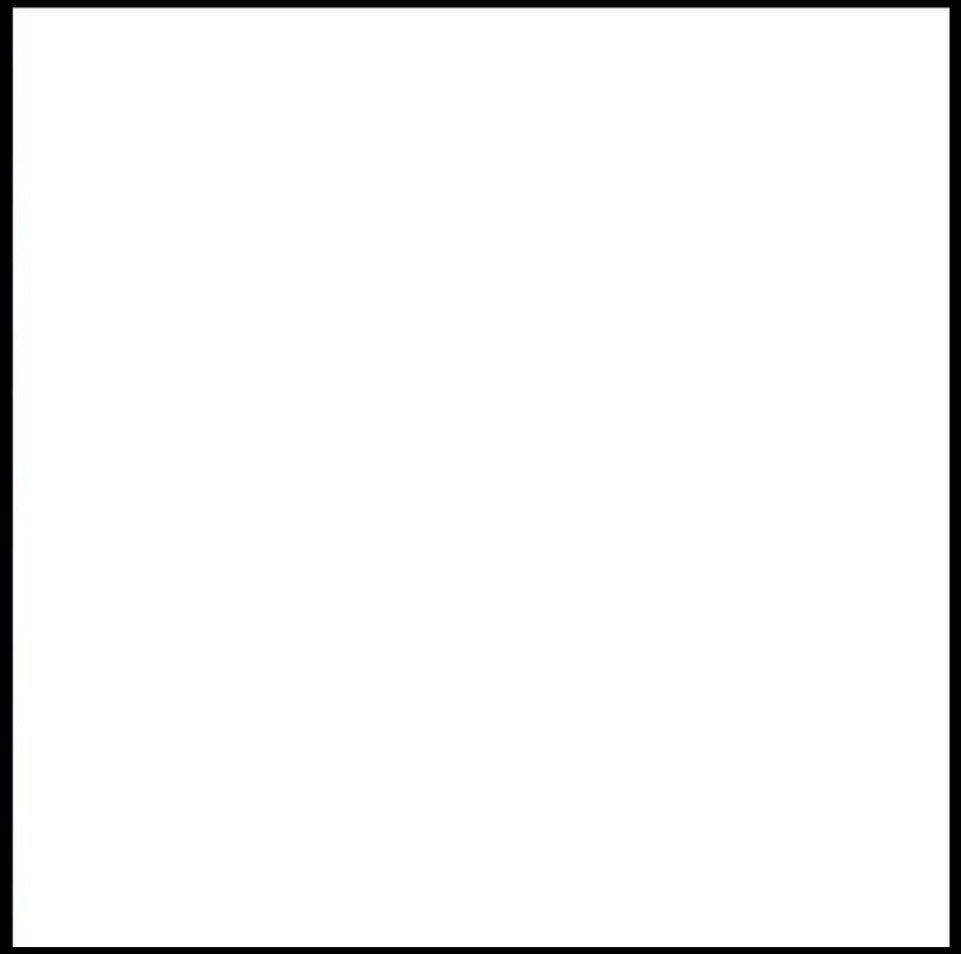
The chaos game: implementation

```
public class ChaosGame {
   public static void main(String[] args) {
      int trials = Integer.parseInt(args[0]);
      double c = Math.sqrt(3) / 2;
      double[] cx = { 0.0, 1.0, 0.5 };
      double[] cy = { 0.0, 0.0, c   };

      StdDraw.setPenRadius(0.01);
      double x = 0.0, y = 0.0;
      for (int t = 1; 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);
      }
   }
}
```

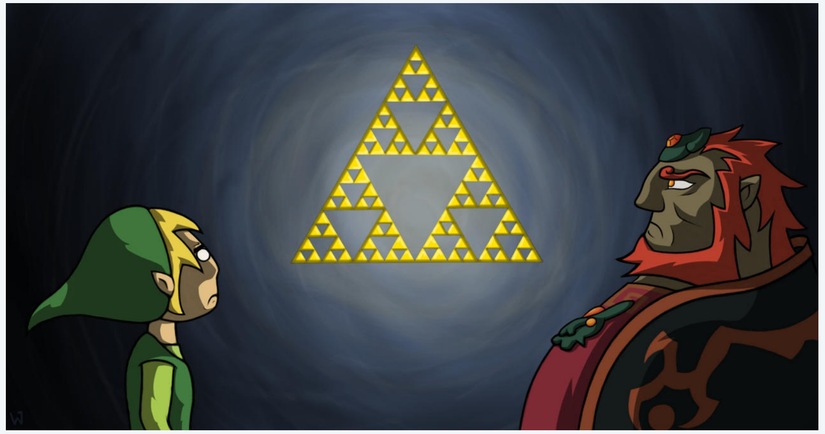
vertices of triangle

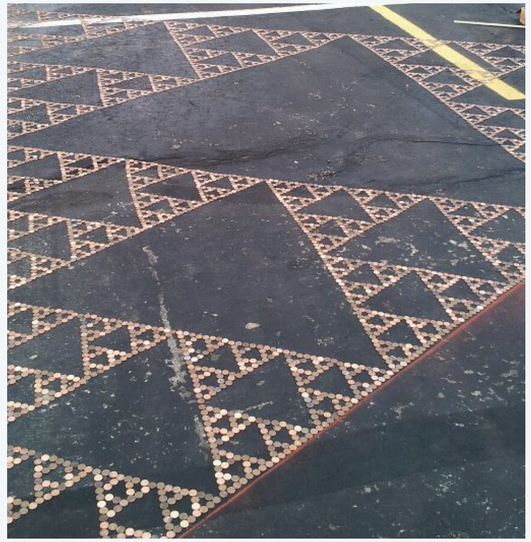
midpoint

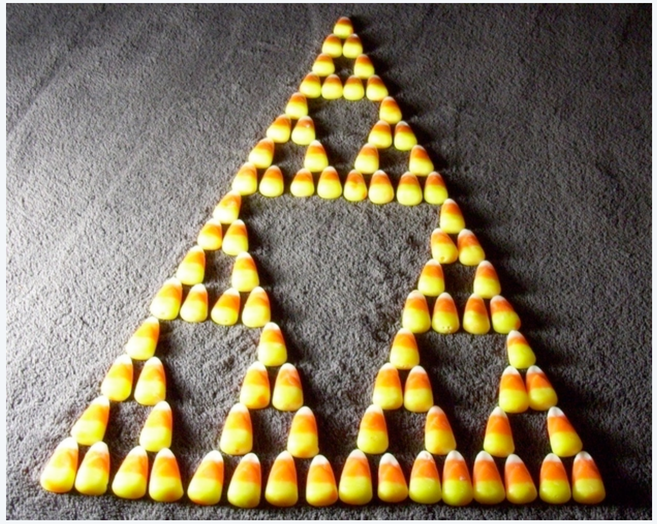
```
~/cos126/io> java-introcs ChaosGame 10000
```

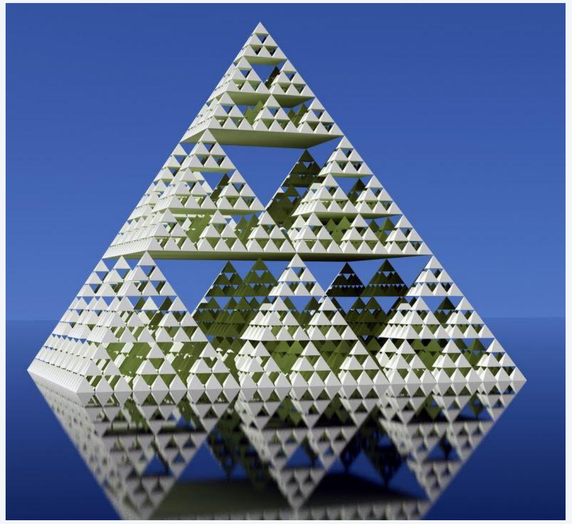


Sierpinski triangles in the wild







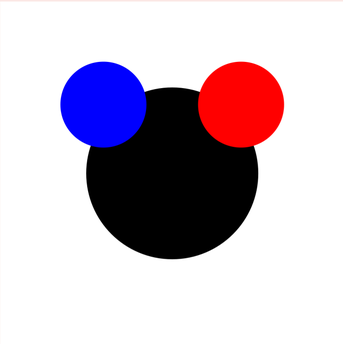


Input and output: quiz 3

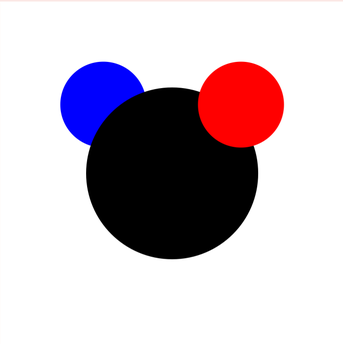
What is the result of executing the following code fragment?

```
// black circle (center)
StdDraw.setPenColor(StdDraw.BLACK);
StdDraw.filledCircle(0.5, 0.5, 0.25);

// small blue circle (upper left)
StdDraw.setPenColor(StdDraw.BLUE);
StdDraw.filledCircle(0.3, 0.7, 0.125);

// small red circle (upper right)
StdDraw.setPenColor(StdDraw.RED);
StdDraw.filledCircle(0.7, 0.7, 0.125);
```

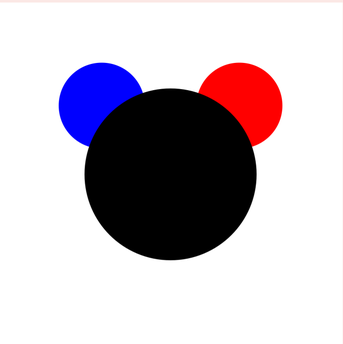
A.

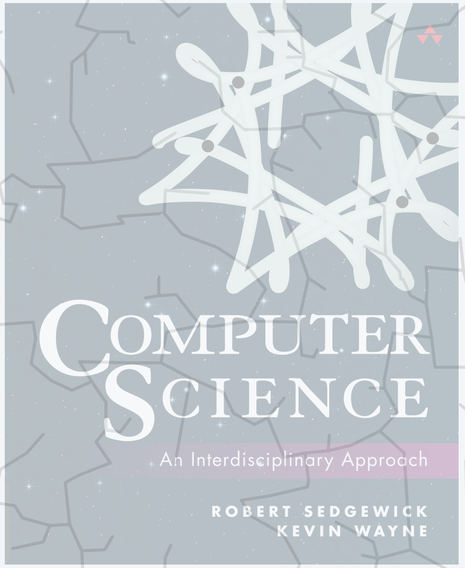


B.



C.





https://introcs.cs.princeton.edu

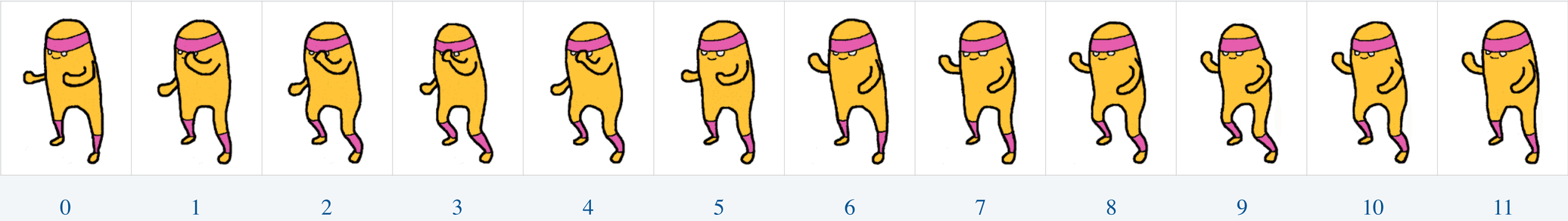
1.5 Input and Output

- *standard input and output*
- *redirection and piping*
- *standard drawing*
- *animation*

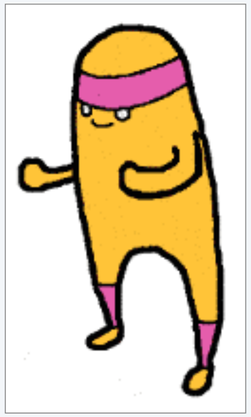
Computer animation

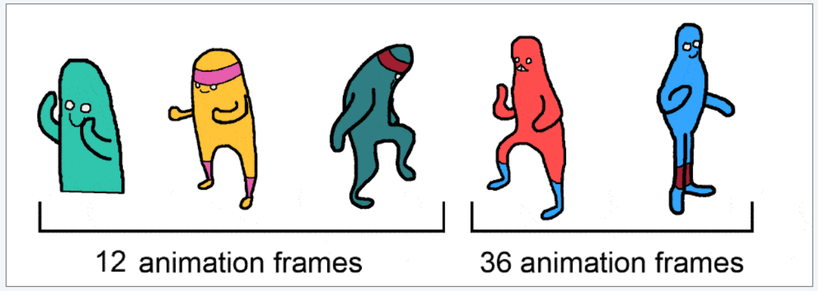
To create an animation, repeat the following:

- Clear the drawing window.
- Draw next animation frame.
- Pause for a short period of time.



12 animation frames





Bottom line. Animation loop produces the illusion of motion.

Animation loop

Goal. Read animation frames from command line and display in an animation loop. ← *"cel" animation*

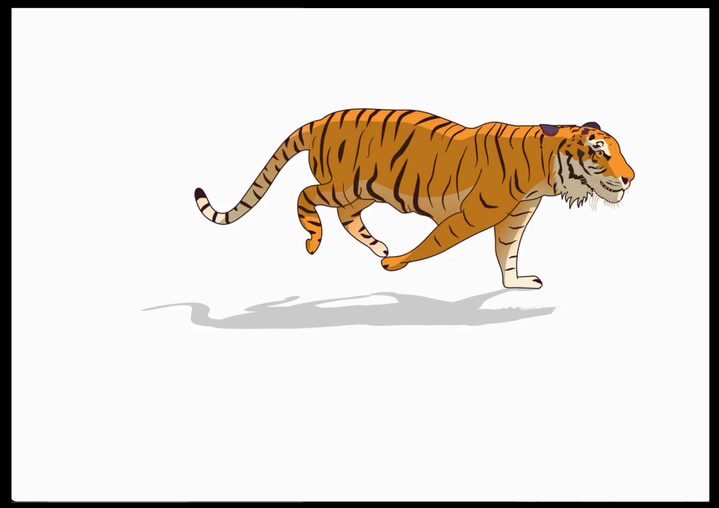
```java
public class AnimationLoop {
   public static void main(String[] args) {
      int n = args.length;

      for (int i = 0; true; i++) {
         String filename = args[i % n];
         StdPicture.read(filename);
         StdPicture.show();
         StdPicture.pause(50);
      }

   }
}
```

animation loop

cycles between 0 and n−1

50ms between frames (20 frames per second)

```
~/cos126/io> ls tiger*.png
tiger00.png  tiger03.png  tiger06.png  tiger09.png
tiger01.png  tiger04.png  tiger07.png  tiger10.png
tiger02.png  tiger05.png  tiger08.png  tiger11.png

~/cos126/io> java-introcs AnimationLoop tiger*.png
```



"wildcard"

Standard drawing library: animation methods

StdDraw. Our library for drawing and animating geometric shapes in a graphical window.

`public class StdDraw`	**description**
`static void enableDoubleBuffering()`	*enable double buffering*
`static void disableDoubleBuffering()`	*disable double buffering*
`static void clear(Color color)`	*clear the background to* `color`
`static void show()`	*show the drawing in a window*
`static void pause(int t)`	*pause for* `t` *milliseconds*
⋮	⋮

Double buffering. Defer drawing shapes on screen until next call to `StdDraw.show()`.

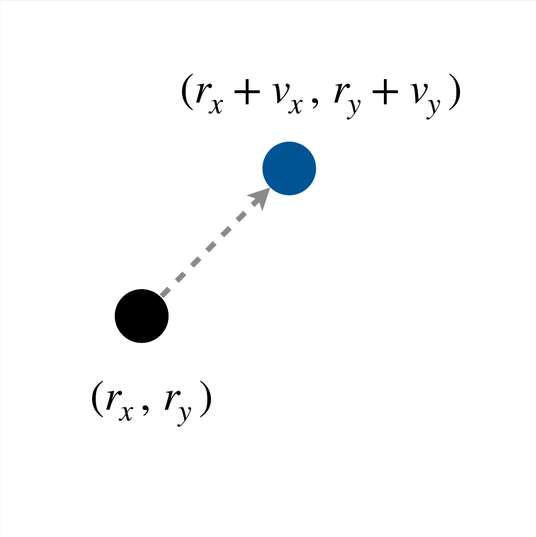
- Smoother animation.
- Faster (when drawing many shapes).

drawing to screen is slow;
typical screen refresh rate = 60 Hz

Moving ball

Moving ball. [with constant velocity]

- Ball has position (r_x, r_y) and velocity (v_x, v_y).
- To move ball, update position to $(r_x + v_x, r_y + v_y)$.



To animate a moving ball, repeat the following:

- Clear the drawing window.
- Move the ball.
- Draw the ball.
- Pause for a short period of time.

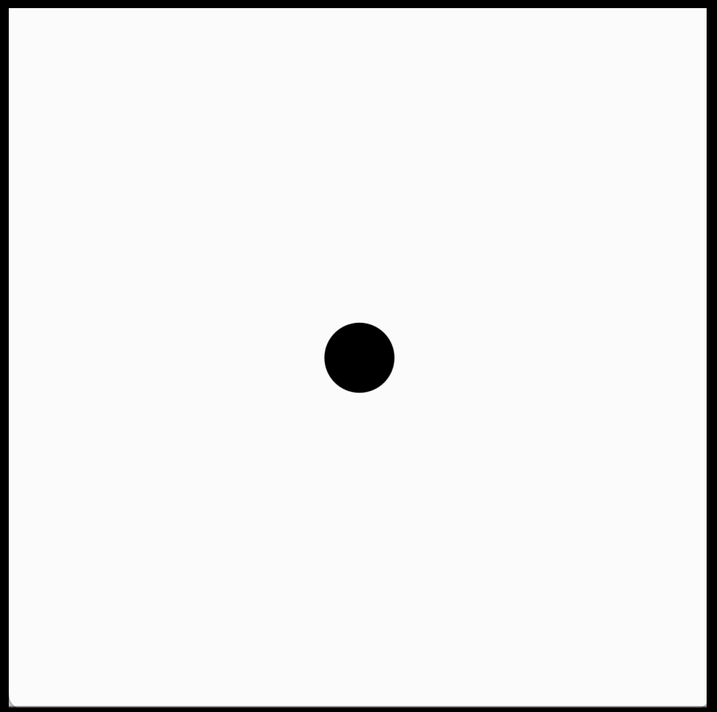
next animation frame

Moving ball

```java
public class MovingBall {
   public static void main(String[] args) {
      double rx = 0.0,   ry = 0.0;
      double vx = 0.001, vy = 0.002;
      double radius = 0.10;

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

      while (true) {
         rx = rx + vx;
         ry = ry + vy;
         StdDraw.clear(StdDraw.WHITE);
         StdDraw.filledCircle(rx, ry, radius);
         StdDraw.show();
         StdDraw.pause(20);
      }

   }
}
```

move the ball

```
~/cos126/io> java-introcs MovingBall
```

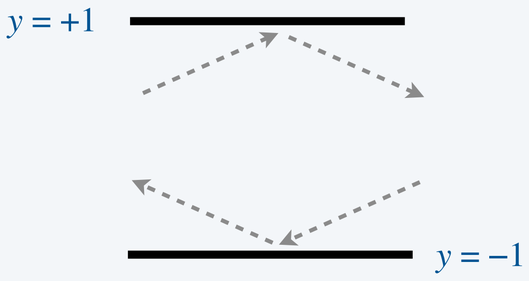


Bouncing ball

To "bounce" the ball off the walls:

- If the ball hits a vertical wall, set v_x to $-v_x$.

- If the ball hits a horizontal wall, set v_y to $-v_y$.

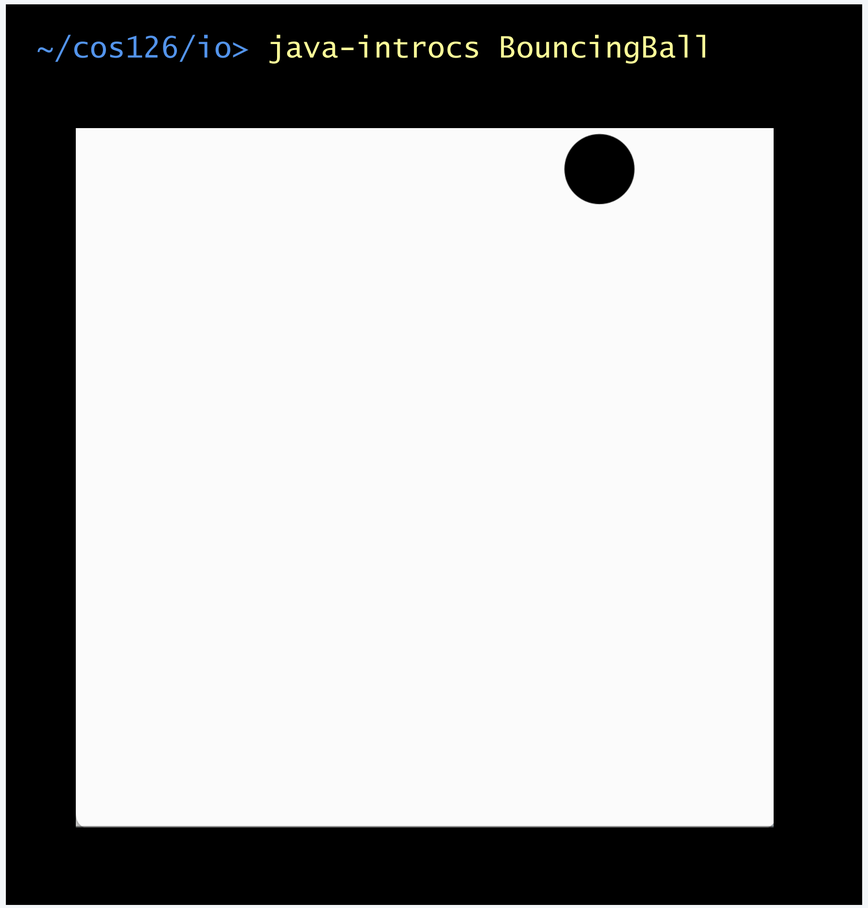


Physics. We're ignoring gravity, spin, friction, inelasticity, air resistance, …

Bouncing ball

```java
public class BouncingBall {
   public static void main(String[] args) {
      double rx = 0.480, ry = 0.860;
      double vx = 0.015, vy = 0.023;
      double radius = 0.1;

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

      while (true) {
         rx = rx + vx;
         ry = ry + vy;
         if (Math.abs(rx) + radius >= 1.0) vx = -vx;
         if (Math.abs(ry) + radius >= 1.0) vy = -vy;
         StdDraw.clear(StdDraw.WHITE);
         StdDraw.filledCircle(rx, ry, radius);
         StdDraw.show();
         StdDraw.pause(20);
      }
   }
}
```

bounce off walls



Standard audio library

StdAudio. Our library for processing digital audio. ← *available with* `javac-introcs` *and* `java-introcs` *commands*

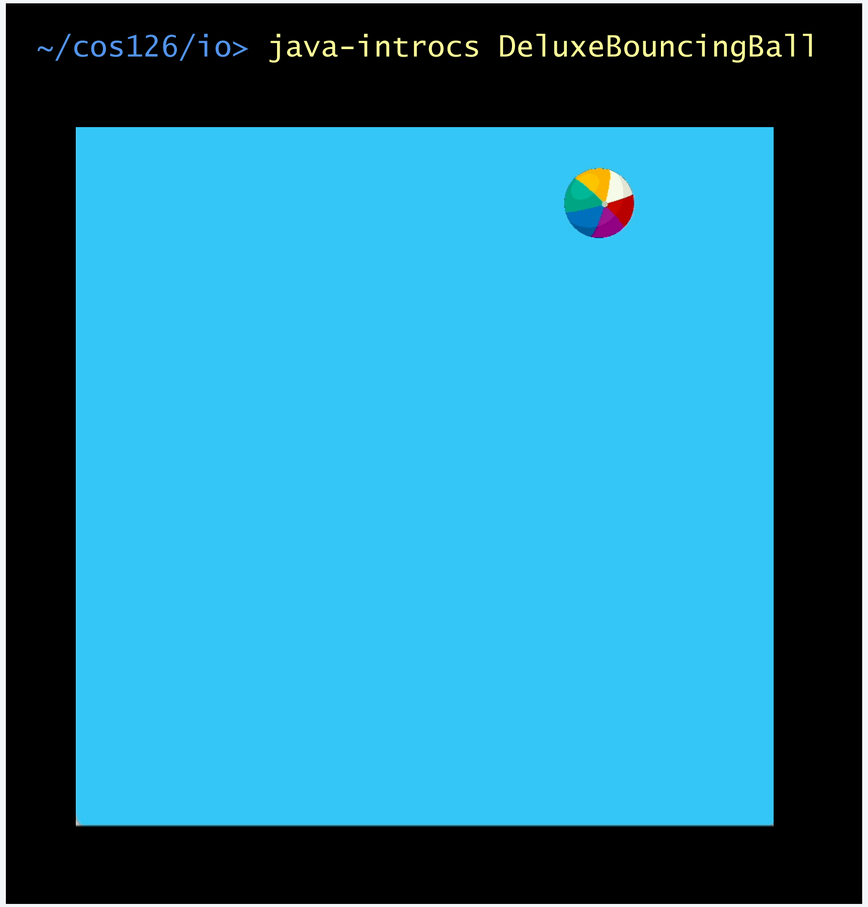
`public class StdAudio`		**description**
`static int`	`SAMPLE_RATE`	44100 (*CD quality audio*)
`static void`	`play(double sample)`	*play the sample*
`static void`	`play(double[] sample)`	*play the samples*
`static void`	`play(String filename)`	*play the audio file* (*do not execute subsequent code until done playing*)
`static void`	`playInBackground(String filename)`	*play the audio file in a background thread* (*execute subsequent code while playing*)
`static double[]`	`read(String filename)`	*read the samples from an audio file*
	⋮	⋮

Deluxe bouncing ball

```
while (true) {
   rx = rx + vx;
   ry = ry + vy;

   // bounce off vertical walls
   if (Math.abs(rx) + radius > 1.0) {
      vx = -vx;
      StdAudio.playInBackground("BallTap.wav");
   }

   // bounce off horizontal walls
   if (Math.abs(ry) + radius > 1.0) {
      vy = -vy;
      StdAudio.playInBackground("BlockHit.wav");
   }

   StdDraw.clear(StdDraw.BOOK_LIGHT_BLUE);
   StdDraw.picture(rx, ry, "ball.png", 2*radius, 2*radius);
   StdDraw.show();
   StdDraw.pause(20);
}
```

plays sound effect

draws picture (resized to specified width and height)



Input and output: quiz 4

What happens if we clear the screen outside the animation loop (instead of inside it)?

A. White only.

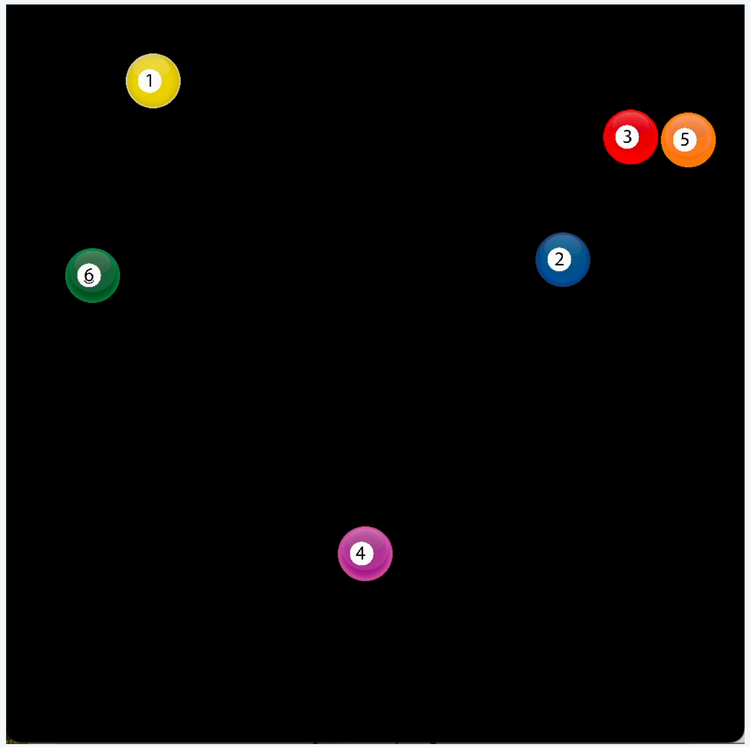
B. Black only.

C. See a trace of the ball's entire path.

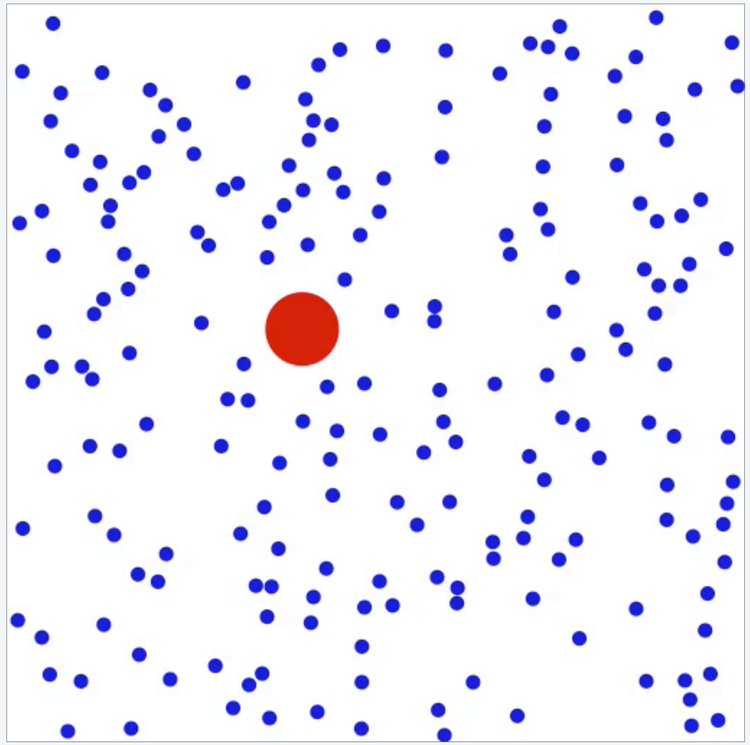
D. Compile-time error.

```
StdDraw.clear(StdDraw.BOOK_LIGHT_BLUE);

while (true) {
   rx = rx + vx;
   ry = ry + vy;
   if (Math.abs(rx) + radius > 1.0) vx = -vx;
   if (Math.abs(ry) + radius > 1.0) vy = -vy;
   StdDraw.clear(StdDraw.BOOK_LIGHT_BLUE);
   StdDraw.picture(rx, ry, "ball.png", 2*radius, 2*radius);
   StdDraw.show();
   StdDraw.pause(20);
}
```

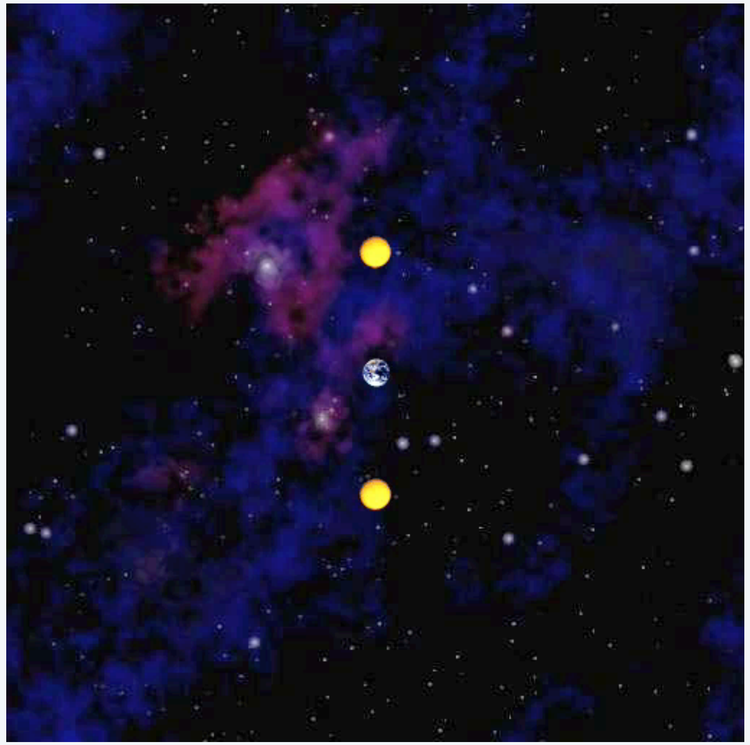
Bouncing ball extensions



multiple balls



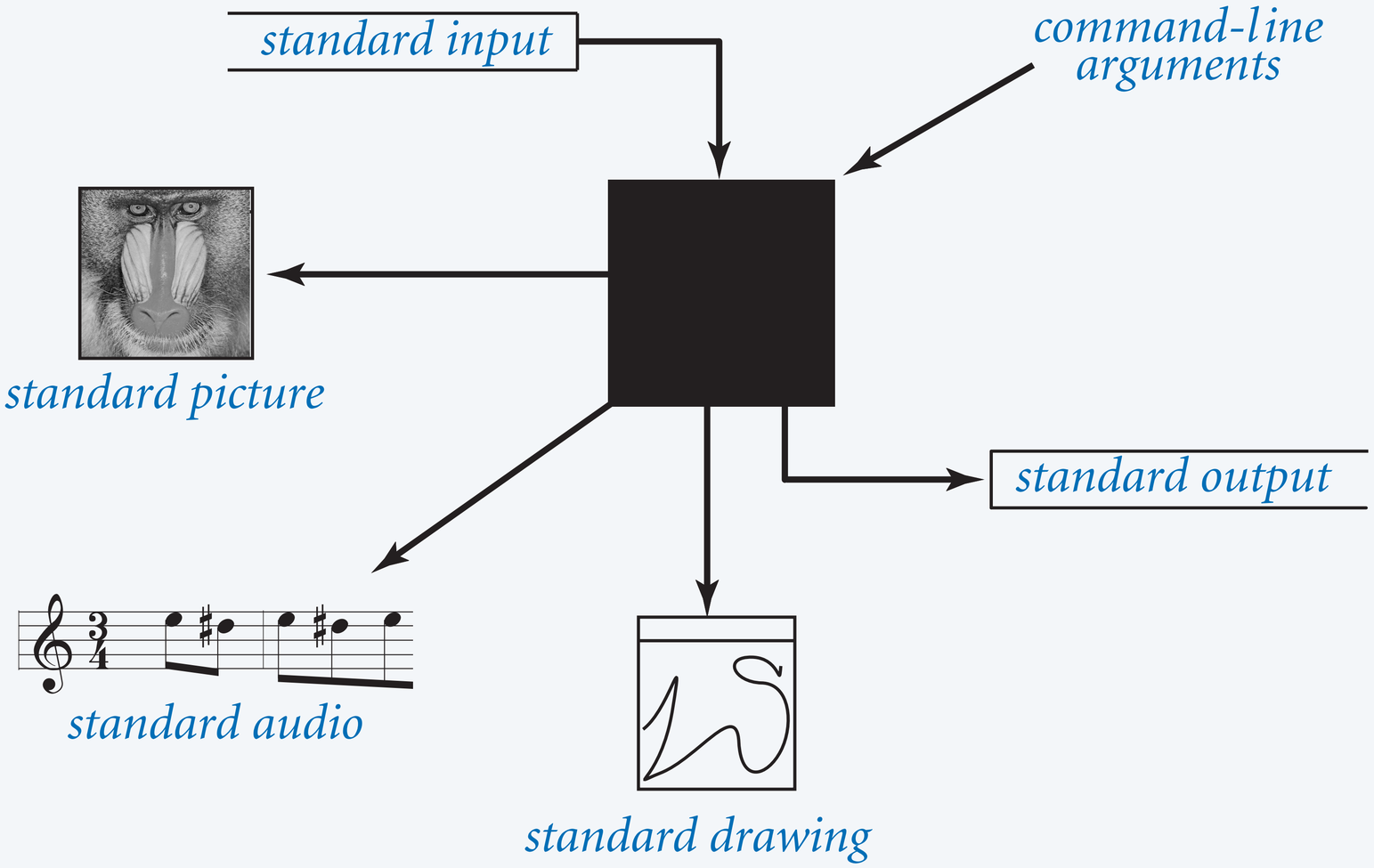
elastic collisions



gravity

Input–output abstractions

Summary. Input and output for text, pictures, drawings, and audio.



Credits

media	source	license
Computer Monitor	iStock	standard license
DEC VT100 Terminal	Wikimedia	CC BY-SA 4.0
Mandrill	USC SIPI Image Database	
Starry Night Stipple	Julia Ying ’26	by author
Sierpinski Coca Cola	Paul Bourke	
The Legend of Sierpinski	Sheilakh	
Sierpinski Pennies	Pinterest	
Sierpinski Candy Corn	Pinterest	
Sierpinski Pyramid	Wikimedia	
Sierpinski Cookie	unknown	

Credits

media	source	license
Dancing Characters	Mark Knight	
Tiger Animation Frames	Adobe Stock	education license
Beach Ball	Open Clip Art	public domain
Sound Effects	Mixkit	Mixkit free license
Pool Balls	Openclipart	public domain