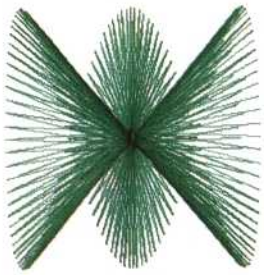
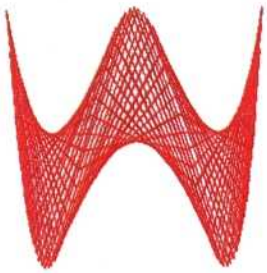
LISSAJOUS FIGURES





One Step Over The Line

The Drunkard's Walk theorem states that after N steps in completely random directions the probability is better than 0.5 that the drunkard's distance from the starting place will be less than SQR(N) steps. This is a statistical prediction based on a large number of steps. LOGO lets you test it for yourself:

```
TO DRUNKWALK :STEPNO
   :STEP
CS REPEAT :STEPNO [RT
   (RANDOM 361) FD
   :STEP]
END
```

The operation NUMBER? outputs TRUE if the input is a number, otherwise FALSE is returned. We use this in the procedure PRIME?, which outputs TRUE if its input is a prime, and FALSE otherwise. It begins by checking that the input is indeed a number, and that it is greater than two. PRIME.TEST then checks to see if any integer between the square root of the number and two will divide into it exactly, leaving no remainder.

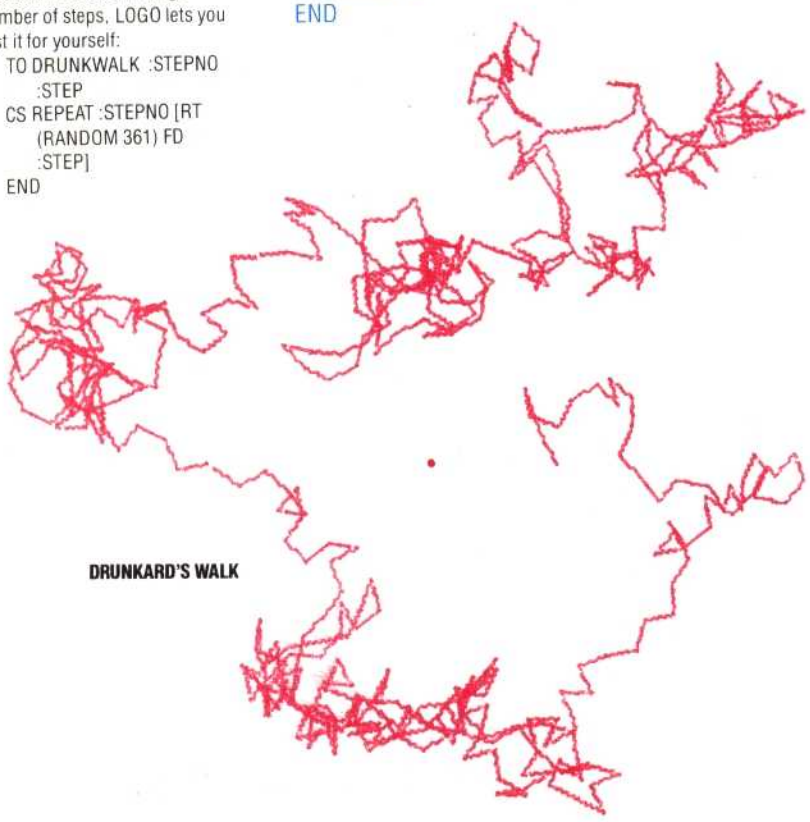
```
TO PRIME? :NO
   IF NOT NUMBER? :NO THEN PRINT [NOT A
      NUMBER DUMMY] STOP
   IF :NO < 2 THEN OUTPUT "FALSE
   OUTPUT PRIME.TEST :NO INTEGER SQRT :NO
END

TO PRIME.TEST :NO :FACT
   IF :FACT = 1 THEN OUTPUT "TRUE
   IF ( REMAINDER :NO :FACT ) = 0 THEN OUTPUT
      "FALSE
   OUTPUT PRIME.TEST :NO :FACT — 1
END
```

RANDOM NUMBERS

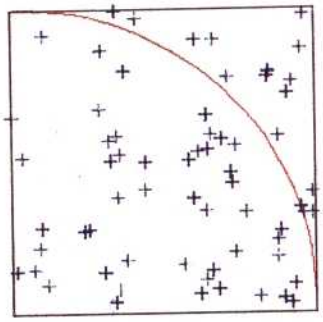
RANDOM n outputs a random integer between 0 and n–1. The procedure DRUNK makes the turtle stagger across the screen, turning a random angle at each step. The input A gives the maximum size of the turn that can be made at any time. If you run this procedure you will find that the turtle turns in vague circles, moving to the left or to the right depending on the value assigned to A.

```
TO DRUNK :A
   FORWARD 1
   RIGHT ( — :A/2 + RANDOM :A)
   DRUNK :A
END
```



DRUNKARD'S WALK

STEVE MALONE



PI COMES TO MONTE CARLO

The so-called 'Monte Carlo method' is a technique for solving mathematical problems through the use of random numbers.

We'll demonstrate by finding an approximation to pi by using this method. Our illustration shows a quarter-circle drawn within a square. The area of the square is 100 × 100 square units, and the area of the quarter-circle is (1÷4) × pi × 100 × 100 square units. The ratio of the areas circle ÷ square is equal to pi ÷ 4. Now drop a pin at random on the square 1,000 times and count how many times the pin falls within the quarter-circle; call this number IN. The value of IN/1000 should be approximately the same as the result of: circle ÷ square — i.e. pi ÷ 4. So if we do the experiment, multiply IN by four and divide by 1,000, then the result should be an approximation to pi. That is precisely what the following procedures do:

```
TO MC
   DRAW
   PU
   MAKE "IN 0
   MC1 1000 100 100
   ( PRINT [VALUE OF PI IS] 0.004 * ( :IN ) )
END

TO MC1 :NO :XNO :YNO
   IF :NO = 0 THEN STOP
   RANDOM.POINT :XNO :YNO
   IF INSIDE? THEN MAKE "IN :IN + 1
   MC1 :NO — 1 :XNO :YNO
END
```

The procedure MC simply sets the conditions, calls MC1 and prints the results. MC1 does most of the work, calls RANDOM.POINT to position the turtle, and then increments IN if the point is inside the circle. This continues until the procedure has been carried out the correct number of times.

```
TO RANDOM.POINT :XNO :YNO
   SETXY RANDOM :XNO RANDOM :YNO
END

TO INSIDE?
   IF ( XCOR * XCOR + YCOR * YCOR ) < 10000
      THEN OUTPUT "TRUE
   OUTPUT "FALSE
END
```

RANDOM.POINT sets the turtle at a random position within the square, while INSIDE? checks to see if the turtle lies within the circle. It will take some time to run this, but eventually a value for pi of 3.15999 will be obtained.