Spectral test

- good generators will pass it
- bad generators are likely to fail it
- idea:
  - let the length of the period be m
  - take t consecutive numbers
  - construct a set of t-dimensional points: $\{ (X_n, X_{n+1}, \ldots, X_{n+t-1}) \mid 0 \le n < m \}$
- when t increases the periodic accuracy decreases
  - a truly random sequence would retain the accuracy

Java: Class `Random`

```
private AtomicLong seed;
private final static long multiplier = 0x5DEECE66DL;
private final static long addend = 0xBL;
private final static long mask = (1L << 48) - 1;

protected int next(int bits) {
    long oldseed, nextseed;
    do {
        oldseed = seed.get();
        nextseed = (oldseed * multiplier + addend) & mask;
    } while (!seed.attemptUpdate(oldseed, nextseed));
    return (int)(nextseed >>> (48 - bits));
}

public int nextInt() {  return next(32); }
```

Random shuffling

- generate random permutation, where all permutations have a uniform random distribution
- shuffling ≈ inverse sorting (!)
- ordered set $S = \langle s_1, \ldots, s_n \rangle$ to be shuffled
- naïve solution
  - enumerate all possible $n!$ permutations
  - generate a random integer $[1, n!]$ and select the corresponding permutation
  - practical only when n is small

Random sampling without replacement

- guarantees that the distribution of permutations is uniform
  - every element has a probability $1/n$ to become selected in the first position
  - subsequent position are filled with the remaining $n - 1$ elements
  - because selections are independent, the probability of any generated ordered set is $1/n \cdot 1/(n-1) \cdot 1/(n-2) \cdot \ldots \cdot 1/1 = 1/n!$
  - there are exactly $n!$ possible permutations → generated ordered sets have a uniform distribution

Random numbers in games

- obvious uses
  - terrain generation
  - events
  - character creation
  - decision-making
- not-so-obvious uses
  - game world compression
  - synchronized simulation

Game world compression

- used in *Elite* (1984)
- finite and discrete galaxy
- enumerate the positions
- set the seed value
- generate a random value for each position
  - if smaller than a given density, create a star
  - otherwise, space is void
- each star is associated with a randomly generated number, which used as a seed when creating the star system details (name, composition, planets)
- can be hierarchically extended

Synchronized simulation

- used in *Age of Empires* (1997)
- command categories:
  - deterministic: computer
  - indeterministic: human
- distribute the indeterministic commands only
- deterministic commands are derived from pseudo-random numbers
  → distribute the seed value only
- consistency checks and recovery mechanisms

Recapitulation

- pseudo-random numbers
  - not truly random but appear to be
  - generated with arithmetic operations
  - characteristic regularities
- linear congruential method
  - simple implementation, well studied
  - problem: choice of parameters
- random shuffling
  - random sampling without replacement