Algorithms and Networking for Computer Games

Chapter 2: Random Numbers

 http://www.wiley.com/go/smed

What are random numbers good for (according to D.E. Knuth)

- simulation
- sampling
- numerical analysis
- computer programming
- decision-making
- aesthetics
- recreation

Random numbers?

- there is no such thing as a ‘random number’
  - is 42 a random number?
- definition: a sequence of statistically *independent* random numbers with a uniform *distribution*
  - numbers are obtained by chance
  - they have nothing to do with the other numbers in the sequence
- uniform distribution: each possible number is equally probable

Methods

- random selection
  - drawing balls out of a ‘well-stirred urn’
- tables of random digits
  - decimals from π
- generating data
  - white noise generators
  - cosmic background radiation
- computer programs?

Generating random numbers with arithmetic operations

- von Neumann (ca. 1946): middle square method
  - take the square of previous number and extract the middle digits
- example: four-digit numbers
  - $r_i = 8269$
  - $r_{i+1} = 3763$ ($r_i^2 = 68\underline{3763}61$)
  - $r_{i+2} = 1601$ ($r_{i+1}^{\ 2} = 14\underline{1601}69$)
  - $r_{i+3} = 5632$ ($r_{i+2}^{\ 2} = 2\underline{5632}01$)

Truly random numbers?

- each number is completely determined by its predecessor!
- sequence is not random but *appears to be*
- → pseudo-random numbers
- all random generators based arithmetic operation have their own in-built characteristic regularities
- hence, testing and analysis is required

Middle square (revisited)

- another example:
  - $r_i = 6100$
  - $r_{i+1} = 2100$ ($r_i^2 = 37\underline{2100}00$)
  - $r_{i+2} = 4100$ ($r_{i+1}{}^2 = 4\underline{4100}00$)
  - $r_{i+3} = 8100$ ($r_{i+2}{}^2 = 16\underline{8100}00$)
  - $r_{i+4} = 6100 = r_i$ ($r_{i+3}{}^2 = 65\underline{6100}00$)
- how to counteract?

Words of the wise

- ‘random numbers should not be generated with a method chosen at random’
  — D. E. Knuth
- ‘Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin.’
  — J. von Neumann

Words of the more (or less) wise

- ‘We guarantee that each number is random individually, but we don’t guarantee that more than one of them is random.’

  — anonymous computer centre’s programming consultant (quoted in *Numerical Recipes in C*)

Other concerns

- speed of the algorithm
- ease of implementation
- parallelization techniques
- portable implementations

Linear congruential method

- D. H. Lehmer (1949)
- choose four integers
  - modulus: m $(0 < m)$
  - multiplier: a $(0 \leq a < m)$
  - increment: c $(0 \leq c < m)$
  - starting value (or seed): X_0 $(0 \leq X_0 < m)$
- obtain a sequence $\langle X_n \rangle$ by setting $X_{n+1} = (aX_n + c) \bmod m \quad (n \geq 0)$

Linear congruential method (cont'd)

- let $b = a - 1$
- generalization:
  $X_{n+k} = (a^k X_n + (a^k - 1)\, c/b) \bmod m$
  $(k \geq 0,\ n \geq 0)$
- random floating point numbers $U_n \in [0, 1)$:
  $U_n = X_n / m$

Random integers from a given interval

- Monte Carlo methods
  - approximate solution
  - accuracy can be improved at the cost of running time
- Las Vegas methods
  - exact solution
  - termination is not guaranteed
- Sherwood methods
  - exact solution, termination guaranteed
  - reduce the difference between good and bad inputs

Choice of modulus *m*

- sequence of random numbers is finite → period (repeating cycle)
- period has at most *m* elements → modulus should be large
- recommendation: *m* is a prime
- reducing modulo: *m* is a power of 2
  - $m = 2^i : x \bmod m = x \sqcap (2^i - 1)$

Choice of multiplier *a*

- period of maximum length
  - $a = c = 1$: $X_{n+1} = (X_n + 1) \bmod m$
  - hardly random: …, 0, 1, 2, …, $m - 1$, 0, 1, 2, …
- results from Theorem 2.1.1
  - if m is a product of distinct primes, only $a = 1$ produces full period
  - if m is divisible by a high power of some prime, there is latitude when choosing a
- rules of thumb
  - $0.01m < a < 0.99m$
  - no simple, regular bit patterns in the binary representation

Choice of increment c

- no common factor with m
  - $c = 1$
  - $c = a$
- if $c = 0$, addition operation can be eliminated
  - faster processing
  - period length decreases

Choice of starting value X_0

- determines from where in the sequence the numbers are taken
- to guarantee randomness, initialization from a varying source
  - built-in clock of the computer
  - last value from the previous run
- using the same value allows to repeat the sequence

Tests for randomness 1(2)

- Frequency test
- Serial test
- Gap test
- Poker test
- Coupon collector's test

Tests for randomness 2(2)

- Permutation test
- Run test
- Collision test
- Birthday spacings test
- Spectral test

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:
    $\{ (X_n, X_{n+1}, \ldots, X_{n+t-1}) \mid 0 \leq n < m \}$
- when *t* increases the periodic accuracy decreases
  - a truly random sequence would retain the accuracy

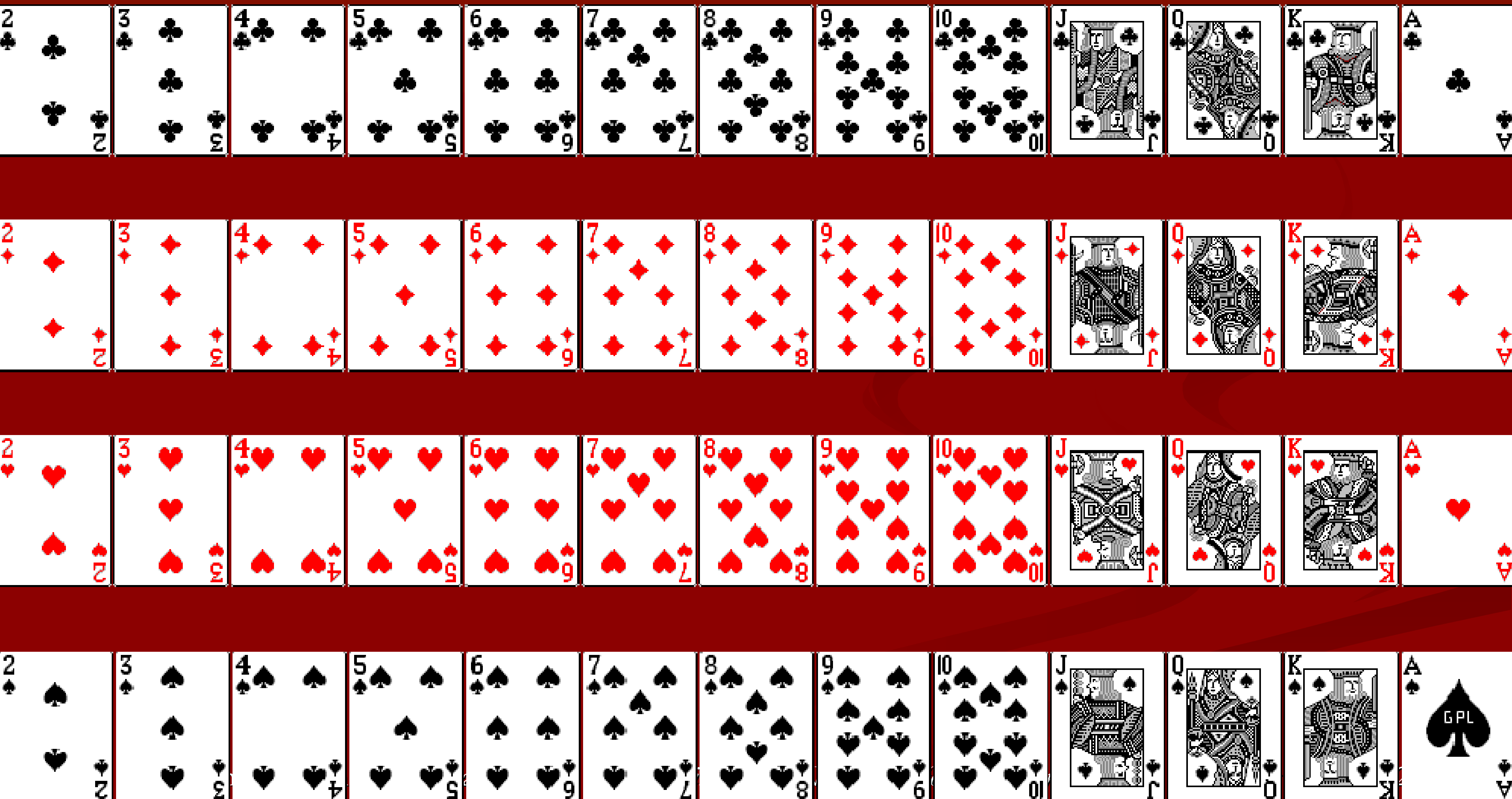
Random shuffling

- 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

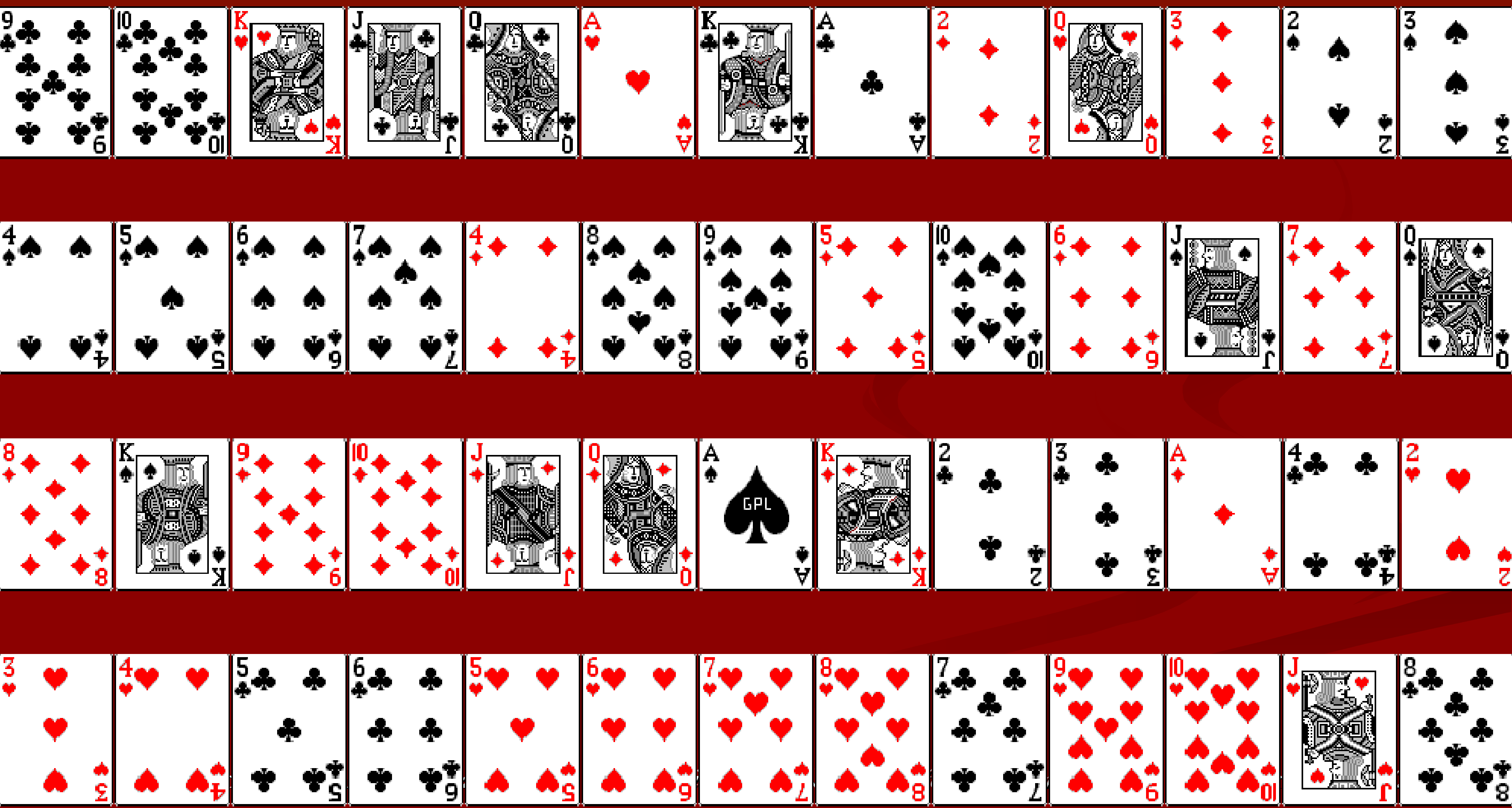
Random sampling without replacement

- 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 $\rightarrow$ generated ordered sets have a uniform distribution

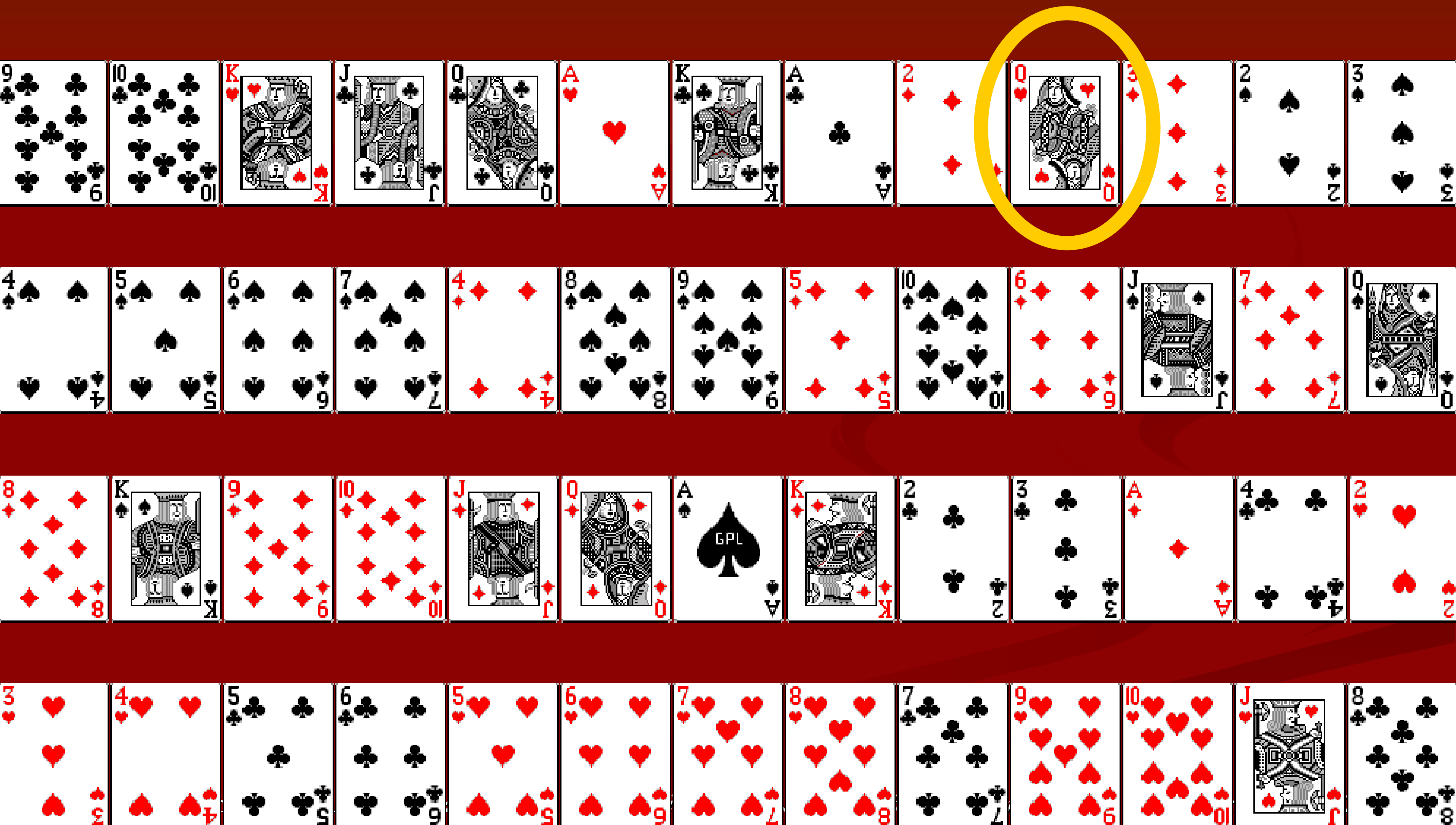
Premo: Standard order



Premo: After a riffle shuffle and card insertion



Premo: The inserted card



Random numbers in games

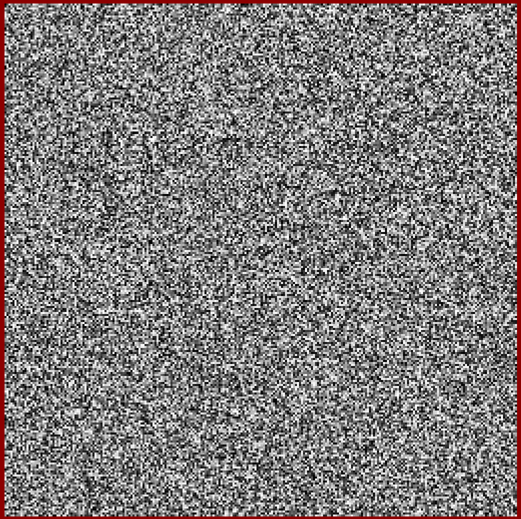
- terrain generation
- events
- character creation
- decision-making
- game world compression
- synchronized simulation

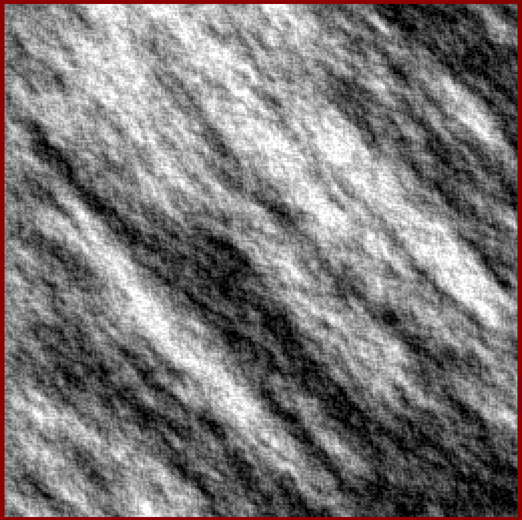
Game world compression

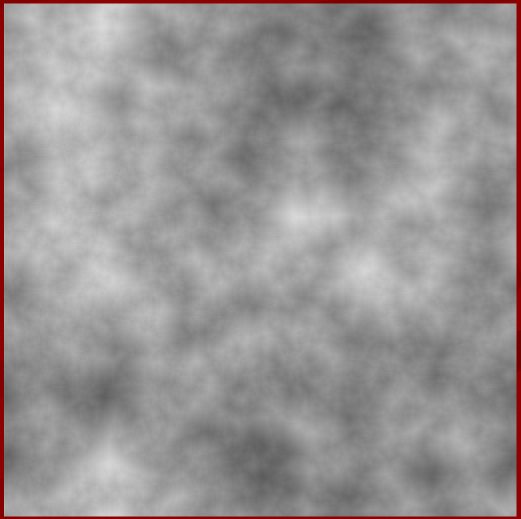
- used in *Elite* (1984)
- finite and discrete galaxy
- enumerate the positions
- set the seed value
- generate a random value for each position
  - if smaller than a given density, create a star
  - otherwise, space is void
- each star is associated with a randomly generated number, which used as a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

Terrain generation 1(2)

- simple random
- limited random
- particle deposition







Terrain generation 2(2)

- fault line
- circle hill
- midpoint displacement