Voluntary exercise project

- group work: 2–3 persons
- web page (currently in Finnish only):
  http://staff.cs.utu.fi/kurssit/peliohjelmointi/
- topic: select from the given list or suggest your own
- supervisors
  - Olli Luoma
  - Kai Nikulainen
  - Johannes Tuikkala
- questions and enquiries to **po@it.utu.fi**

Idea and implementation

- implement a simple computer game which either
  - provides a computer-controlled opponent

  or
  - allows multiplaying in a network
- platforms
  - PCs (preferably using Java)
  - mobile phones (using J2ME)

Important dates

- introductory lecture: **October 1**, 4 p.m.
- deadline for enrolments: **October 3**
- deadline for topic selection and preliminary plan submission: **October 17**
- deadline for final plan submission: **October 31**
- deadline for finished project: **January 31, 2004**

Final remarks

- exercise project is voluntary!
  - excercise project does not require participation on this course
  - passing this course does not require participation on the exercise project
- but it is beneficial
- questions & enrolments to **po@it.utu.fi** or to the supervisors (not to me!)

Reminder: Bonus on grades

- find error or suggest improvements on the lecture notes
- first one to send gets point(s); check the existing errata!
- among those who receive *at least* 10 points:
  - student with most points gets 0.5 bonus on the grade
  - the next best three get 0.25 bonus on the grade
- scoring (excerpt)
  - 1 – error in text
  - 2 – error in equation or code
  - 4 – bug in code or improvement on a method
- e-mail to **jouni.smed@cs.utu.fi**, subject prefix ‘a4cg’

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 $(0 \le a < m)$
  - increment: c $(0 \le c < m)$
  - starting value (or seed): X_0 $(0 \le X_0 < m)$
- obtain a sequence $\langle X_n \rangle$ by setting
  $X_{n+1} = (aX_n + c) \bmod m$ $(n \ge 0)$

Linear congruential method (cont'd)

- let $b = a - 1$
- generalization:
  $X_{n+k} = (a^k X_n + (a^k - 1)\, c/b) \bmod m$
  $(k \ge 0,\ n \ge 0)$
- 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
  - 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