Zufallszahlengeneratoren in C++11

David Sander

Einstieg

- „Das, wobei unsere Berechnungen versagen, nennen wir Zufall.“

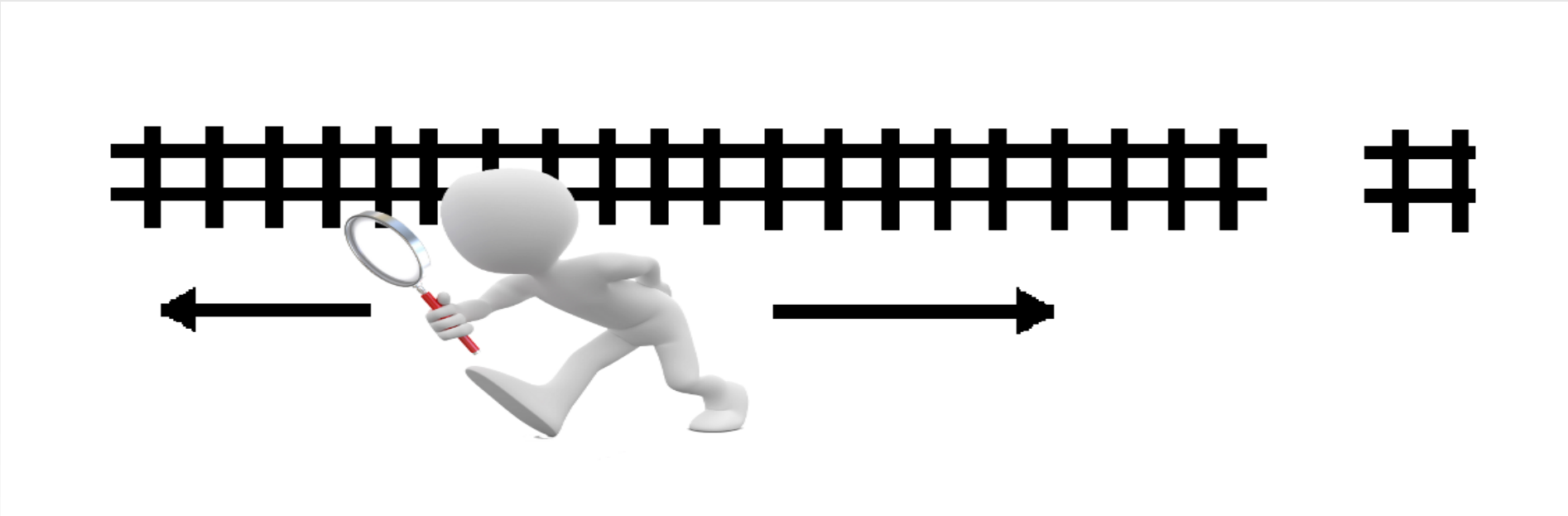
  Albert Einstein

Motivation

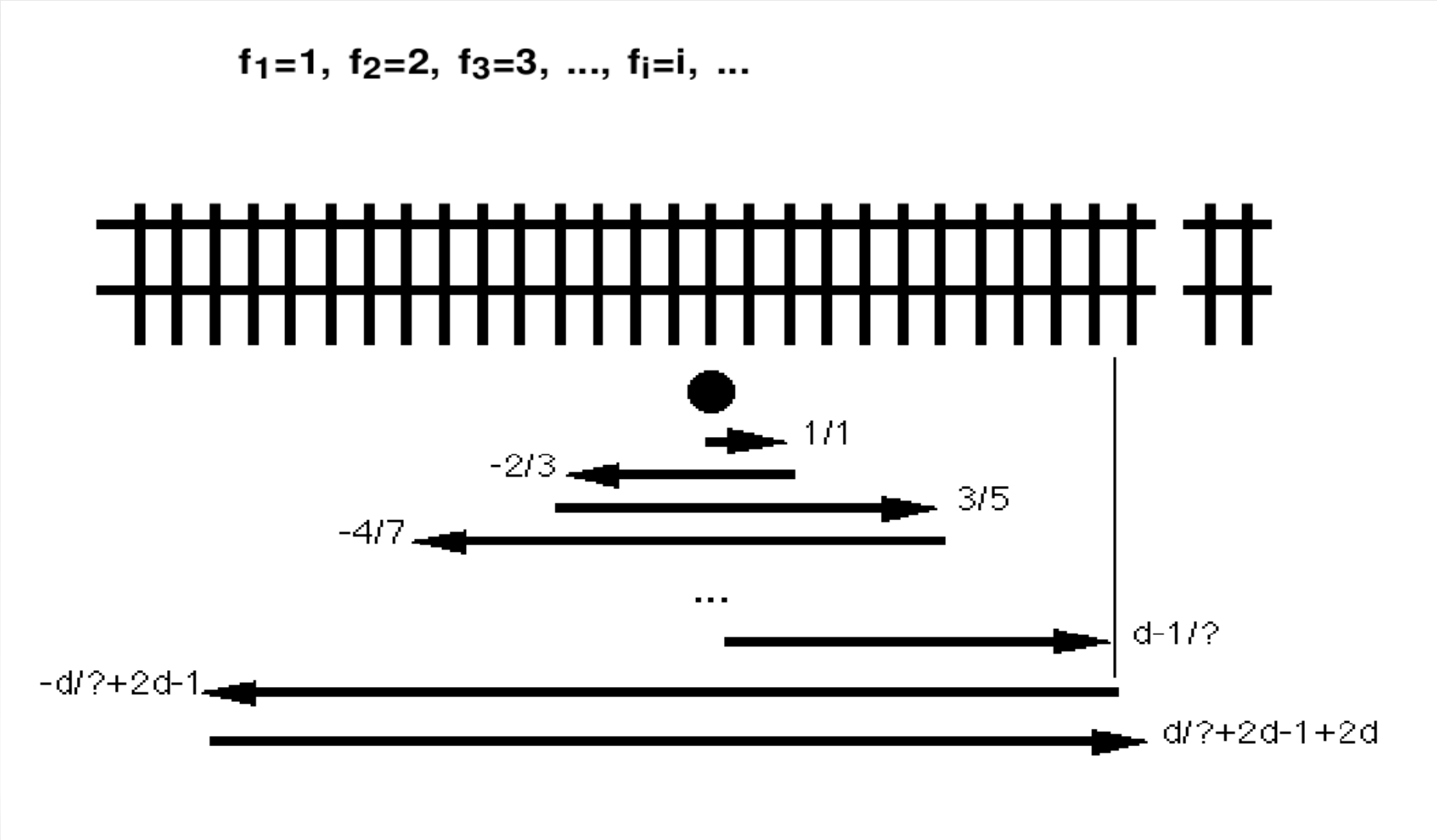
- Der suchende Fahrradfahrer (nach Andreas Schwill)

Motivation

- Im i-ten Schritt suche in Richtung
  i mod 2 (1=rechts, 0=links) bis zur Distanz f_i.



Motivation



Motivation

- durchlaufende Strecke:

$$(1+(1+2)+(2+3)+(3+4)+...((d-1)+d)+2d$$
$$=2\ \Sigma k + 3d = d(d-1)+3d = d^2 +2d=O(d^2)$$

Motivation

- ein einfacher Zufallsalgorithmus:
  - Solange das Fahrrad nicht gefunden ist, tue folgendes:
    - Wähle zufällig eine Richtung R (links/rechts).
    - Gehe einen Schritt in Richtung R.

Überraschend: Der Fahrradfahrer findet das Fahrrad mit Wahrscheinlichkeit 1.

Einsatzgebiete

- Simulation
  - Generierung zufälliger Ereignisse entsprechend statistischer Vorgaben (OdemX)
- Kryptographie
- Tests (Generierung von Testsätzen)
- probabilistische Algorithmen
  - Monte-Carlo- und Las-Vegas-Methoden

Physikalischer Zufallszahlengenerator

- zur Erzeugung von Zufallszahlen werden physikalische Prozesse benutzt:
  - thermisches Rauschen eines Widerstands
  - radioaktive Zerfallsvorgänge
  - atmosphärisches Rauschen
  - …..
- Problem: nicht reproduzierbare Ergebnisse

Pseudozufallsgeneratoren

- Pseudozufallsgeneratoren generieren eine Zahlenfolge, die zwar zufällig aussieht, es aber nicht ist, da sie durch einen deterministischen Algorithmus berechnet wird

- Problem/Vorteil: reproduzierbare Ergebnisse

Zufallszahlen in C++98

- ZZ = (a * ZZ + c) % m
- BSD-Unix verwendet hier zum Beispiel:
  - a = 1103515245, c = 12345 und m = 2^{31}
- srand(x) //initialer Seed
- ZZ = rand() //generiert eine Zufallszahl

Zufallszahlen in C++98

Beispiel 01/02

Zufallszahlen in C++11

- es gibt zwei Bestandteile für die Erzeugung von Zufallszahlen
  - Generatoren
    - Engines
  - Verteilungen

Engines

- Datenquelle mit gleich verteilten Zahlen
- mit folgenden Befehlen:
  - **min()**: minimaler Wert der durch den Befehl **operator()** zurückgegeben wird
  - **max()**: maximaler Wert der durch den Befehl **operator()** zurückgegeben wird
  - **operator()**: Ausgabe von gleich verteilten Werten zwischen **min()** und **max()**
  - **seed()**: initialen Seed einstellen

Engines

- ***linear_congruential***:
  x(i) = (A * x(i-1) + C) mod M
- **mersenne_twister**
- **subtract_with_carry**
  x(i) = (x(i - R) - x(i - S) - cy(i - 1)) mod M mit
  cy(i) = x(i - S) - x(i - R) - cy(i - 1) < 0 ? 1 : 0
- **random_device**: Generates random numbers by using an external device.

Engine Template

- http://msdn.microsoft.com/en-us/library/bb982659.aspx

Verteilungen

- Klasse die einen Stream von gleich verteilten Zahlen in einen Stream mit einer bestimmten Verteilung verwandelt

- **reset()**: löscht alle bisherigen Werte die eine **Engine** geliefert hat
- **operator()**: gibt einen Wert (einen zu einem Zeitpunkt) mit einer bestimmten Verteilung zurück

Verteilungen

- bernoulli_distribution
- binomial_distribution
- exponential_distribution
- gamma_distribution
- geometric_distribution
- normal_distribution
- poisson_distribution
- uniform_int_distribution
- uniform_real_distribution

Benutzung von Engines und Verteilungen

Beispiel 04/05/06

Probleme

- Generatoren haben nicht die gleiche Basis → die Weitergabe an eine generische Funktion ist schwierig (http://stackoverflow.com/questions/2241327/how-do-i-pass-a-c11-random-number-generator-to-a-function)
- Thread-Safety ? Zufallszahlengeneratoren bieten die gleiche Thread-Safety wie Container (http://stackoverflow.com/questions/11214394/random-number-generators-and-thread-safety)

Quellen

- http://cplus.kompf.de/artikel/random.html
- http://www.codeguru.com/cpp/cpp/cpp_mfc/stl/article.php/c15319/A-TR1-Tutorial-Generating-Random-Numbers.htm
- T. Hinze, Prinzipien zur Erzeugung von Zufallszahlen in der Informatik, TU Dresden
- Andreas Schwill, Der kontrollierte Zufall in der Informatik, Institut für Informatik Universität Potsdam, 2002
- http://en.wikipedia.org/wiki/C%2B%2B11#Extensible_random_number_facility