# De randomness van rand()

## Namen en datum

Alexander Streng groep 17 17-02-2014

## Doel

Bij het maken van een salt & pepper noise generator, had ik een vermoeden dat rand() niet helemaal compleet random is. Dit wordt vaak gezegd op fora etc. Graag wil ik deze bewering dus testen door hem(rand()) te vergelijken met de random generator implementatie van c++11 (gebruik makende van de Mersenne Twister 19937 generator) en kijken welke meer random is.

## Hypothese

Ik denk dat de c++ implementatie meer random is bij gebruik van dezelfde seed. Omdat de rand altijd een nummer tussen de 0 en 32767 genereerd (en deze vervolgens **modulo** doet.). Overigens is dit voor het genererenvan salt & pepper noise niet merkbaar.

## Werkwijze

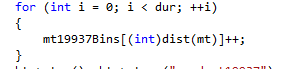
Na eerst een voor-onderzoekje te hebben gedaan kwam ik op de volgende link (is wel een leuk filmpje als je geïnteresseerd bent / tijd hebt):

<http://channel9.msdn.com/Events/GoingNative/2013/rand-Considered-Harmful>

Na het voor-onderzoek zal ik beide methodes implementeren. De c++ 11 implementatie die gebruik maakt van de Mersenne Twister engine is net als rand() een pseudo-random generator van 32 bits.

Ik ga 2 ‘histogrammen’ opstellen van de 2 random number methodes. Dat houd in dat ik een random nummer van 0 – 99 ga genereren. Elke keer als een nummer gegenereerd wordt, increment ik de bin bijbehorend van dat nummer. Op deze manier kan je na een groot aantal keer duidelijk patronen zien in het genereren van nummers.

Als seed zal ik de huidige tijd gebruiken (time(0)). Ik doorloop eerst 1x een loop van 100,000,000 keer als controle - test. Vervolgens ga ik 20 keer een loop van 100, 000, 000, 000 keer doorlopen en random nummers genereren.



De resultaten sla ik op in een .csv om te vergelijken. Op deze manier wordt de afwijking van de gegenereerde nummers duidelijk zichtbaar.

Meer informatie over Mersenne Twister:

<http://www.cplusplus.com/reference/random/mt19937/>

<http://en.wikipedia.org/wiki/Mersenne_twister>

## Resultaten

De resultaten van de controle ( 100,000,000 keer een nummer genereren ) zijn als volgt;

De resultaten van 100,000,000,000 keer zijn iets interessanter. Zie de laatste pagina voor de tabel.

## Verwerking

Ik heb de resultaten simpelweg opgeslagen als een csv bestand. Op deze manier zijn er makkelijk grafieken te maken. De tabellen tonen het aantal keer dat een nummer random is ‘gegenereerd’ door de methode. Ik heb de loop van 100,000,000,000 (100 miljard) ’s avonds laten lopen. In totaal duurde het een paar uur om de resultaten te genereren.

## Conclusie

In de controleproef is al een duidelijke neerwaartse trend zichtbaar rond de 70. Bij de echte test is het echter zeer duidelijk dat alle getallen > 65 – 70 in het geval van rand() minder vaak (de overige getallen dus vaker) worden gegenereerd. Bij de Mersenne Twister is dit niet het geval.

## Evaluatie

Het is vrij logisch dat de getallen > 65 minder vaak worden gegenereerd. Dit heeft te maken met het feit dat en gebruik wordt gemaakt door modulo. Zoals ook wordt uitgelegd in het channel – 9 filmpje, rand() genereerd een 32 bits (32767) random nummer. Dit betekend dus dat als rand() zonder modulo wordt aangeroepen een getal tussen 0 – 32767 wordt gegenereerd. Als er gebruik wordt gemaakt van modulo, dan worden de getallenreeksen als het ware gemapt:

0, 99 -> 0, 99

100, 199 -> 0, 99

Maar wat gebeurt er met de laatste reeks?

32700, 32767 -> 0, 67

Zoals je wel kunt begrijpen is de kans dus kleiner dat getallen groter dan 67 gegenereerd worden. Deze manier van testen heeft echter wel wat onzekerheden. Als er geen gebruik wordt gemaakt van modulo dan zijn de resultaten waarschijnlijk verschillend. Ook in performance is rand() aanzienlijk sneller dan de Mersenne Twister.

Het zal voor de meeste gevallen niet uitmaken welke PRNG er wordt gebruikt. Hoewel het wel belangrijk is je bewust te zijn van deze tekortkoming van rand(). Voor het genereren van salt & pepper noise is het gebruik dus gewoon mogelijk, maar voor PRNG bij bijvoorbeeld beveiliging is het niet aan te raden.

