Random Numbers

- Random Numbers Library
- Generating Values
- Specifying Distributions

C++11: Random

Compared to C style pseudo random number generation with `std::rand`, C++11 has adopted a facility for generating random numbers with given distributions, but at a price:

The code to role a simple dice isn't any more as easy* as

```
int throw_dice() { return 1 + std::rand() % 6; }
```

but requires at least something along the following lines:

```
int throw_dice() {
    static std::random_device rd;
    static std::mt19937 gen(rd());
    static std::uniform_int_distribution<> dis(1, 6);
    return dis(gen);
}
```

*: ... and wrong or at least flawed for the following reasons: (1) Some C implementations start to repeat the "random" sequence as early as after 65534 repetitions and (2) if the range of pseudo-random numbers (starting with zero) is not evenly divisible by six the chosen way to map the numbers to 1..6 will slightly favour the lower values.

C++11: Random Number Generators

All random number engines generate the next value by applying the function call operator (without arguments).*

Some random number generators may be used without distributions if a uniform distribution over the range of generated values is required, e.g.:

- `std::mt19937` – uniformly distributed values over a 32 bit range
- `std::mt19937_64` – uniformly distributed values over a 64 bit range

The C++11 standard also defines an `std::random_device` which may either be mapped to a non-deterministic random source or (if not available) to one of the existing (pseudo-random) sources.

For more informations on the available random number generators see subsection Random Number Engines in: http://en.cppreference.com/w/cpp/numeric/random

*: The prefix `mt` abbreviates the algorithm (Mersenne Twister) and `19937` is the period, after which the generated numbers start to repeat identically: which is 2^{19937}. Or in other words: if such a generator existed since the big bang and random numbers were extracted with a GHz clock since, until today it would have generated only numbers from a tiny weeny fraction of its non-repeating range.

C++11: Random Number Distributions

All distributions produce the next value by applying the function call operator with a generator (engine) as parameter.

There are many distributions available – separately from the generator – of which the *Uniform Distribution* is probably the one most often used. It is available as:*

- `std::uniform_int_distribution`
  generating integral values in a given range
- `std::uniform_real_distribution`
  generating floating point values in a given range
- `uniform_canonical`
  generating values between 0..1 with a given precision

For more information on the above and other distributions see subsection Random Number Distributions in:
http://en.cppreference.com/w/cpp/numeric/random

*: Note that contrary to the otherwise asymmetric limits, as commonly used in C and the C++-STL, the limits specified for distributions are inclusive and symmetric.

Boost: Random

Boost.Random implements the C++11 conformant Random classes with some additions.