C++11 BLWS (Friday)

A Tour of Boost (cont.)

Short breaks will be inserted as convenient.

Specific Mathematical Areas

Geometry

From the Boost Documentation:

> Boost.Geometry (aka Generic Geometry Library, GGL) defines concepts, primitives and algorithms for solving geometry problems.
> Contained is a dimension-agnostic, coordinate-system-agnostic and scalable kernel, based on concepts, meta-functions and tag dispatching. On top of that kernel, algorithms are built: area, length, perimeter, centroid, convex hull, intersection (clipping), within (point in polygon), distance, envelope (bounding box), simplify, transform, and much more. The library supports high precision arithmetic numbers, such as ttmath.
> Also contained are instantiable geometry classes, but library users can also use their own. Using registration macros or traits classes their geometries can be adapted to fulfil Boost.Geometry concepts.

Polygon

From the Boost Documentation:

> The Boost.Polygon Library provides algorithms focused on manipulating planar polygon geometry data. Specific algorithms provided are the polygon set operations (intersection, union, difference, disjoint-union) and related algorithms such as polygon connectivity graph extraction, offsetting and map-overlay.

Graph Theory

The Boost Graph Library (or BGL in short) helps C++ developers convert practical engineering problems into graph-theory problems.

The library author has published a whole book on it (ISBN: 978-0201729146) but there are also compact introductions like this one:

- http://www.ibm.com/developerworks/aix/library/au-aix-boost-graph/

Differential Equations

From the Boost Documentation:

> Boost.Odeint is a library for solving initial value problems (IVP) of ordinary differential equations (ODEs), which occur nearly everywhere in natural sciences. For example, the whole Newtonian mechanics are described by second order differential equations. They also occur if partial differential equations (PDEs) are discretized. Then, a system of coupled ordinary differential occurs, sometimes also referred as lattices ODEs.

Domain Specific Languages

Parsers and Generators

From the Boost Documentation:

> Boost.Spirit is an object-oriented, recursive-descent parser and output generation library for C++. It allows you to write grammars and format descriptions using a format similar to Extended Backus Naur Form (EBNF) directly in C++. These inline grammar specifications can mix freely with other C++ code and, thanks to the generative power of C++ templates, are immediately executable. In retrospect, conventional compiler-compilers or parser-generators have to perform an additional translation step from the source EBNF code to C or C++ code.

State Machines

From the Boost Documentation:

> Boost.MSM is a library allowing to easily and quickly define state machines of very high performance.

Proto (EDSL Framework)

From the Boost Documentation:

> Boost.Proto is a framework for building Embedded Domain-Specific Languages in C++. It provides tools for constructing, type-checking, transforming and executing expression templates. More specifically, Proto provides:
>
> - An expression tree data structure.
> - A mechanism for giving expressions additional behaviors and members.
> - Operator overloads for building the tree from an expression.
> - Utilities for defining the grammar to which an expression must conform.
> - An extensible mechanism for immediately executing an expression template.
> - An extensible set of tree transformations to apply to expression trees.

Property Map

From the Boost Documentation:

> The Boost.Property_map Library consists mainly of interface specifications in the form of concepts (similar to the iterator concepts in the STL). These interface specifications are intended for use by implementors of generic libraries in communicating requirements on template parameters to their users. In particular, the concepts define a general purpose interface for mapping key objects to corresponding value objects, thereby hiding the details of how the mapping is implemented from algorithms. The implementation of types fulfilling the property map interface is up to the client of the algorithm to provide. The property map requirements are purposefully vague on the type of the key and value objects to allow for the utmost genericity in the function templates of the generic library.

MPL (Meta Programming Library)

From the Boost Documentation:

> The Boost.MPL Library is a general-purpose, high-level C++ template meta-programming framework of compile-time algorithms, sequences and meta-functions. It provides a conceptual foundation and an extensive set of powerful and coherent tools that make doing explict meta-programming in C++ as easy and enjoyable as possible within the current language.

Wave Preprocessor Implementation

From the Boost Documentation:

> The Boost.Wave C++ preprocessor library uses the Spirit parser construction library to implement a C++ lexer with ISO/ANSI Standards conformant preprocessing capabilities. It exposes an iterator interface, which returns the current preprocessed token from the input stream. This preprocessed token is generated on the fly while iterating over the preprocessor iterator sequence (in the terminology of the STL these iterators are forward iterators).