Sujet de stage : Concepts en C++ et CGAL

Titre : Concepts en C++ et CGAL.
Mots clés : C++, programmation générique, concepts, CGAL.
Encadrant : Sylvain Pion (chargé de recherche INRIA).
Chef d'équipe : Jean-Daniel Boissonnat
Lieu du stage : Projet GEOMETRICA, INRIA Sophia Antipolis, France.
Téléphone : 04 92 38 50 25.
Mél : `Sylvain.Pion@sophia.inria.fr`

Contexte `CGAL` [1] (Computational Geometry Algorithms Library) est une importante bibliothèque d'algorithmes géométriques développée dans le projet Geometrica, en collaboration avec d'autres équipes internationales, depuis 10 ans. Elle est écrite en C++, en s'inspirant de manière importante du paradigme de la *programmation générique*. Le leitmotiv de ce paradigme est de maximiser la réutilisabilité du code en exprimant les pré-requis des algorithmes sur leurs entrées sous forme de concepts et en les minimisant. La `STL` [2] (Standard Template Library) est un exemple de bibliothèque suivant également ce paradigme.

`CGAL` utilise ce mécanisme à plusieurs niveaux. D'une part, les algorithmes (comme le calcul d'enveloppes convexes, triangulations de Delaunay...) sont ainsi découplés des objets géométriques de base (points, segments...) qu'ils manipulent. Cette séparation permet d'appliquer les algorithmes dans des contextes multiples, mais aussi de changer la représentation des points pour des raisons de robustesse numérique par exemple... D'autre part, les algorithmes sont séparés des structures de données sur lesquels ils s'executent, afin de pouvoir les utiliser sur des structures provenant d'un contexte externe (d'une application utilisateur typiquement). Les concepts de plus bas niveaux sont ceux des *types de nombres*, qui doivent fournir certaines opérations arithmétiques.

Dans le cadre de la prochaine version du *standard* ISO du langage C++, l'introduction de `concept`s en tant qu'entités du langage à part entière est proposée [3]. Cette fonctionalité permettra d'obtenir des messages d'erreur beaucoup plus clairs de la part du compilateur pour les usages de codes templates en général, et facilitera le développement de codes templates en diagnostiquant les erreurs à la définition des fonctions template, sans devoir les instantier.

Un compilateur C++ supportant cette fonctionalité est disponible[4].

Travail demandé Nous aimerions tester l'impact pratique de ces concepts sur la bibliothèque `CGAL`. Les bénéfices que nous pouvons en tirer sont des messages d'erreurs plus clairs pour les utilisateurs et les développeurs, mais aussi une clarification plus précise de nos concepts pour mieux structurer et documenter le code de `CGAL`.

Le code résultant du stage pourra éventuellement être intégré dans `CGAL` s'il n'est pas trop intrusif.

Outils Le stage contient une importante composante de programmation en C++ faisant appel fortement aux templates. `CGAL` sera utilisé et modifié. Le travail se fera sous Linux.

[1] `http://www.cgal.org/`
[2] `http://www.sgi.com/tech/stl/`
[3] Documents N2081 et suivants à `http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/`
[4] `http://www.generic-programming.org/software/ConceptGCC/`