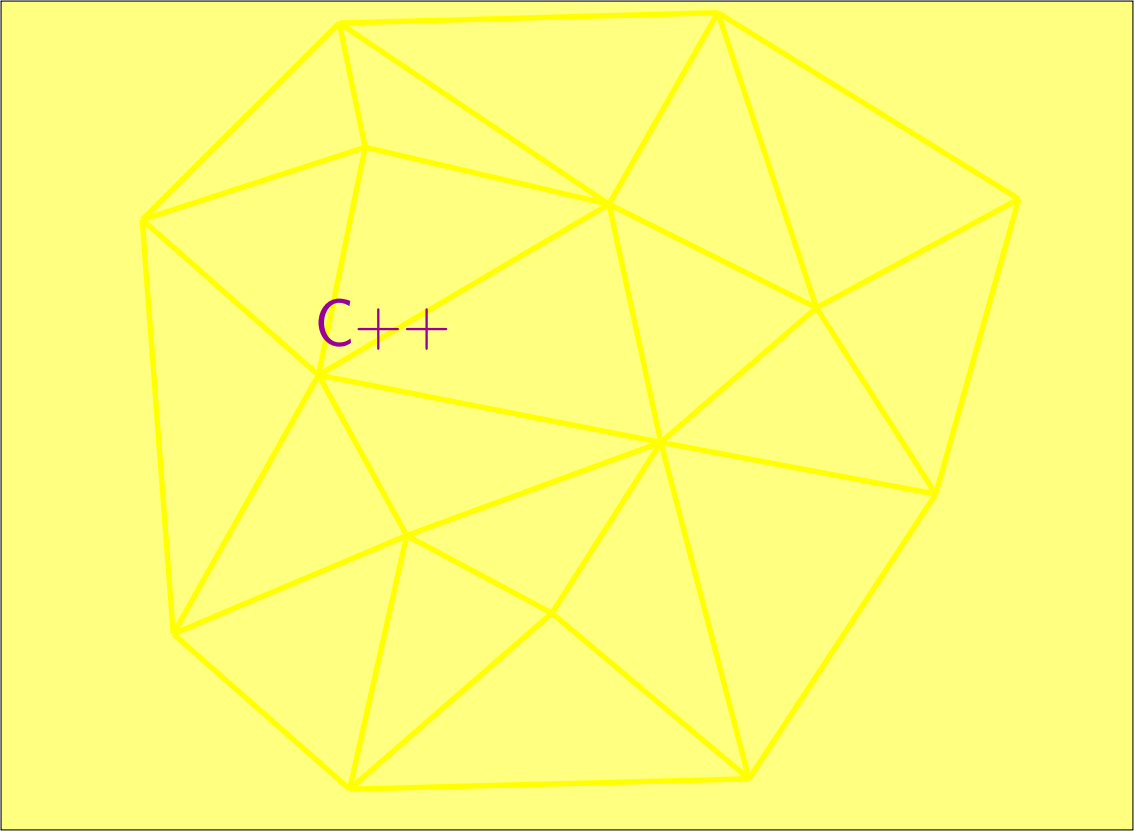
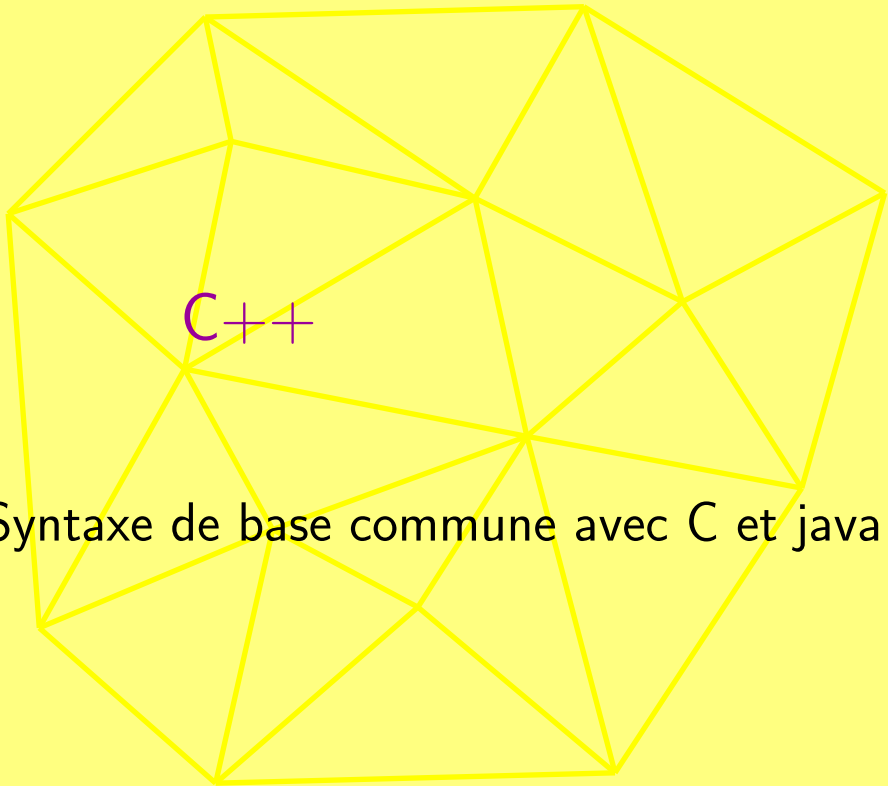
De la géométrie algorithmique au calcul géométrique

l'exemple de la triangulation de Delaunay

La bibliothèque CGAL


C++



Syntaxe de base commune avec C et java

Classe

Classe

```
class A{
  private:
       int i;
  public:
       int get();
       void set(int);
};
```

Classe

```
int A::get(){
      return i;
}

void A::set(int n){
      i = n;
}
```

Classe

```
class A{
  private:
       int i;
  public:
       int get() {return i;}
       void set(int n) {i=n;}
} ;  // fonctions "inline"
```

Constructeur, affectation, destruction

Constructeur, affectation, destruction

```
class A {
    int i; // private
public:
    A();                 // default constructor
    A( const A& a);      // copy constructor
    A( int n);           // user defined
    ~A();                // destructor
};
```

Constructeur, affectation, destruction

```
class A {
    int i; // private
public:
    A();              // default constructor
    A( const A& a);   // copy constructor
    A( int n);        // user defined
    ~A();             // destructor
};
```

```
int main() {
    A a1;  // appelle default c.
    A a2 = a1;  // copy constructor
    A a3(a1);   // copy constructor
    A a4(1);    // user constructor
} // destructor: appel a la sortie.
```

Héritage

Surcharge

Fonctions membres virtuelles

Membres statiques

Templates

Templates

```
template <class T>
void swap( T& a, T& b) {
    T tmp = a;
    a = b;
    b = tmp;
}
```

Templates

```
template <class T>
void swap( T& a, T& b) {
    T tmp = a;
    a = b;
    b = tmp;
}
int main() {
    int i = 5;
    int j = 7;
    swap( i, j);  // utilise "int" pour T.
}
```

Templates

```
template <class T>
void swap( T& a, T& b) {
    T tmp = a;
    a = b;
    b = tmp;
}
int main() {
    A i (5);
    A j ; j.set(7)
    swap( i, j);  // utilise "A" pour T.
}
```

Aller dans une classe quand on y est pas !

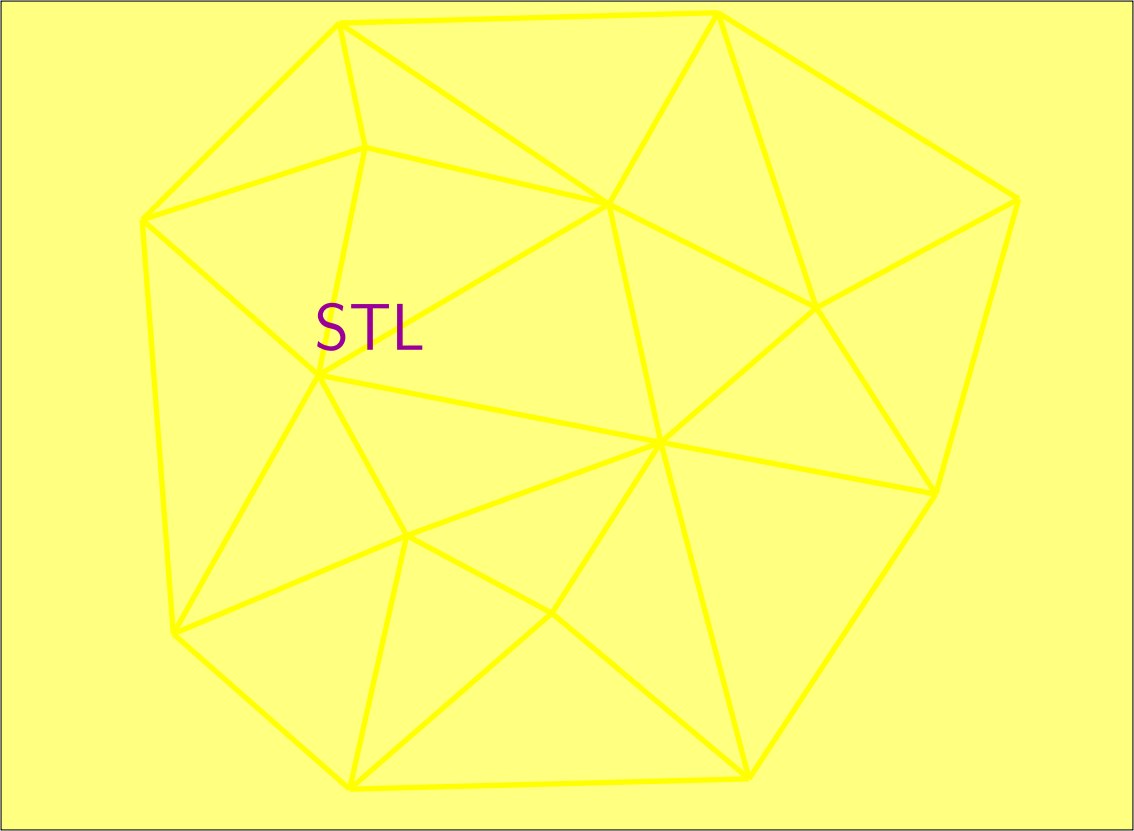
Aller dans une classe quand on y est pas !

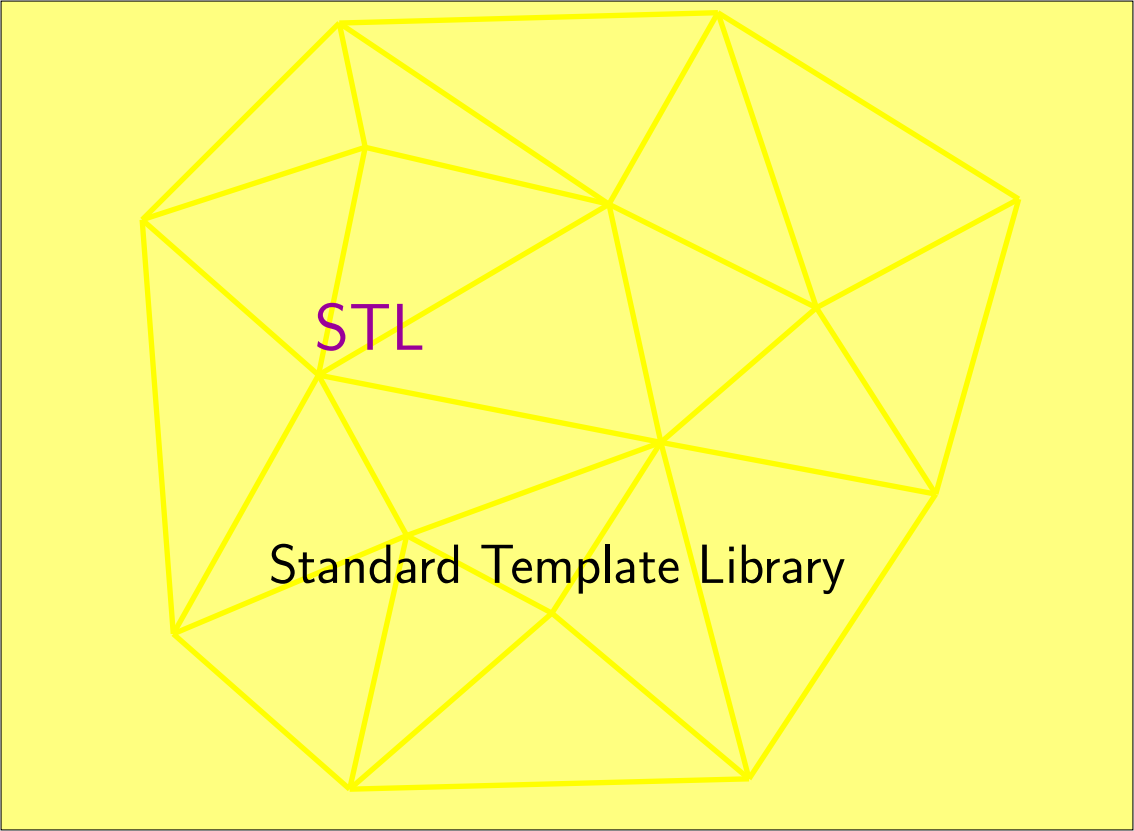
```
class A {
    static int i=0;
};

int main(){
     int j = A::i;
}
```

Namespace

Namespace

```
int main(){
    int i = CGAL::max(3,4);
}
```


STL


STL
Standard Template Library

Bibliothèque

Bibliothèque

Structures de données:
`list, vector, set, map`

Algorithmes:
`find, sort`

Iterator

Iterator

Parcourir une structure (`container`)

Iterator

Parcourir une structure (`container`)

ressemble à un pointeur

```
operator*()
operator->()
operator++()
operator--()
...
```

exemple: liste

exemple: liste

```
list<int> L;
L.push_back(0);
L.push_front(1);
L.insert(++L.begin(), 2);
copy(L.begin(), L.end(),
          ostream_iterator<int>(cout, " "));
// The values that are printed are 1 2 0
```


CGAL, vue d'ensemble

Computational Geometry Algorithms Library

Computational Geometry Algorithms Library

Kernel

objets élémentaires (points, droites...)

Computational Geometry Algorithms Library

Kernel

objets élémentaires (points, droites...)

Basic library

Algorithmes et structures de données

Computational Geometry Algorithms Library

Kernel

objets élémentaires (points, droites...)

Basic library

Algorithmes et structures de données

Support library

Extension STL (circulator)
Types numériques
Entrées sorties


CGAL Support library

Type de nombre

Type de nombre

```
int main(){
  int n1=0;
  float n2 = 3.1415927;
  double n2bis=5/11;
  using namespace CGAL;
  Quotient<int> n3(5,11);
  MP_Float n4=3.14159265358979323846264338327;
  Lazy_exact_nt<Quotient<MP_Float> > n5;
// Core, GMP, Leda
}
```

Handles

Handles

Encapsulation du pointeur.

juste `*` et `->`

Circulator

Circulator

Variation de l'iterator de STL

```
template <class Circulator, class T>
bool contains( Circulator c,
               Circulator d, const T& value) {
    if (c != 0) {
        do {
            if (*c == value)
                return true;
        } while (++c != d);
    }
    return false;
}
```

Générateurs

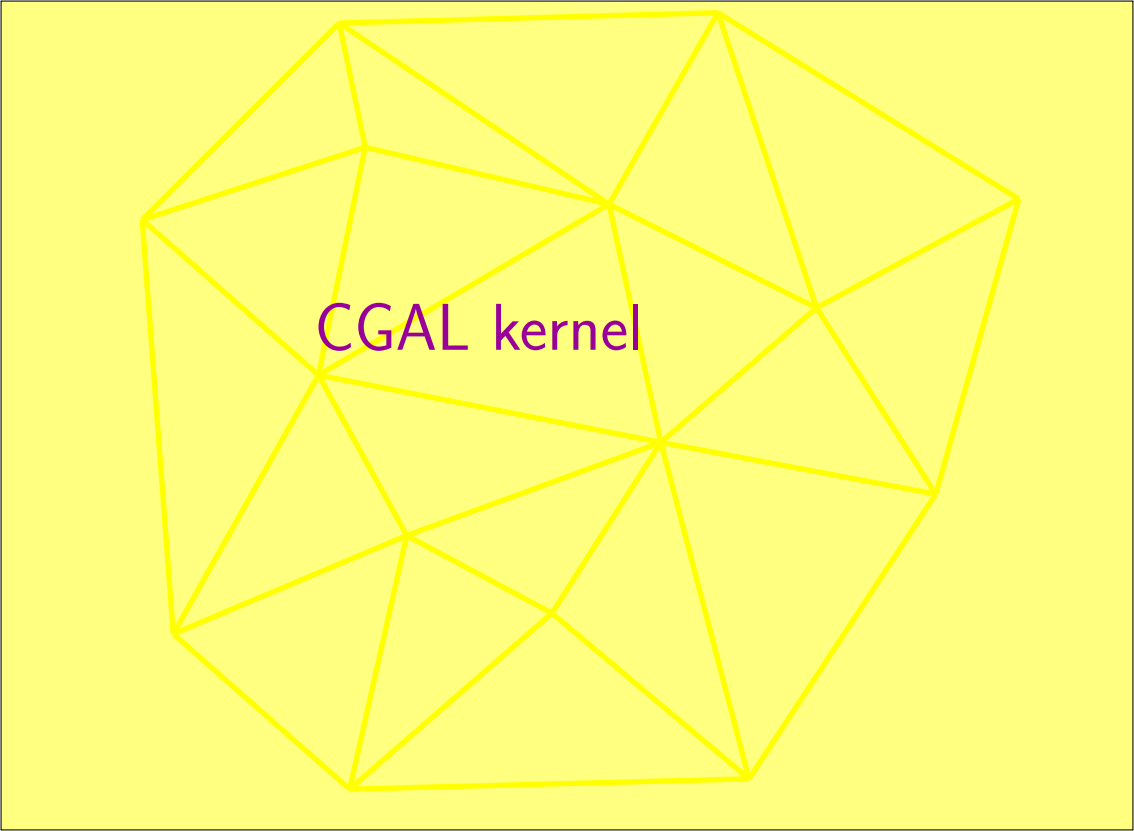
Générateurs

de points aléatoires sur un cercle, dans un disque, sur une grille. . .

Divers

Divers

Timers
Structures de données (hashmap)
IO Streams
Couleurs
postscript
Qt
Geomview


CGAL kernel

Kernel

Kernel

```
Simple_cartesian<double>
```

Kernel

`Simple_cartesian<double>`

Objets

- `Kernel::Point_2`
- `Kernel::Vector_2`
- `Kernel::Direction_2`
- `Kernel::Line_2`
- `Kernel::Ray_2`
- `Kernel::Segment_2`
- `Kernel::Triangle_2`
- `Kernel::Iso_rectangle_2`
- `Kernel::Circle_2`
- `Kernel::Object_2`

Kernel

`Simple_cartesian<double>`

Objets

- `Compare_x_2`
- `Compare_distance_2`
- `Has_on_2`
- `Side_of_oriented_circle_2`

Kernel

`Simple_cartesian<double>`

Objets

- `Compare_x_2`
- `Compare_distance_2`
- `Has_on_2`
- `Side_of_oriented_circle_2`

Constructions


CGAL Basic library

Basic library

Basic library

Enveloppe convexe
Polygones Polyhèdres
Carte planaire, arrangement
Triangulation
Optimisation
Structures de recherche

Traits class

Traits class

Paramétrer un algorithme/structure de données

```
typedef CGAL::Cartesian<double>                Rep;
typedef CGAL::Point_2<Rep>                     Point;
typedef CGAL::Delaunay_triangulation_2<Rep>    Delaunay;
```

```
// K_ is the new kernel, and K_Base is the old kernel
template < typename K_, typename K_Base >
class MyCartesian_base
  : public K_Base::template Base<K_>::Type
{
  typedef typename K_Base::template Base<K_>::Type   OldK;
public:
  typedef K_                                        Kernel;
  typedef MyPointC2                                 Point_2;
}

template < typename FT_ >
struct MyKernel
  : public MyCartesian_base<MyKernel<FT_>, CGAL::Cartesian<FT_> >
{};

typedef MyKernel<double>      MK;
typedef CGAL::Filtered_kernel_without_type_equality<MK> K;
typedef CGAL::Delaunay_triangulation_2<K> Delaunay;
```

Exemple

Exemple

TD2/initial.C

/demo/Qt_widget/basic/tutorial3/tutorial3.C

```
#ifndef CGAL_USE_QT
#include <iostream>
int main(int, char*){
  std::cout << "Sorry, this demo needs QT..." << std::endl; return 0;}
#else
#include <CGAL/Cartesian.h>
#include <CGAL/Point_2.h>
#include <CGAL/Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget_Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget.h>
#include <CGAL/IO/Qt_widget_layer.h>
#include <qapplication.h>

typedef CGAL::Cartesian<double>             Rep;
typedef CGAL::Point_2<Rep>                  Point;
typedef CGAL::Delaunay_triangulation_2<Rep> Delaunay;


//////////////////////////////////////////////////////
//  Les objets affiches sont declares ici en variable globales
//  l'affichage est fait dans la methode "draw" ci dessous

Delaunay dt;
Point    the_point=Point(-1,-1);

class My_layer : public CGAL::Qt_widget_layer{
  void draw(){
    *widget << CGAL::RED << dt << CGAL::PointSize(6) << CGAL::BLUE << the_point
;
  }
};

class My_window : public CGAL::Qt_widget {
public:
  My_window(int x, int y)
  {
    resize(x,y);
    attach(&layer);
  };
private:
  //this method is called when the user presses the mouse
  void mousePressEvent(QMouseEvent *e)
  {
    Qt_widget::mousePressEvent(e);

    ////////////////////////////////////////////////////////
    //  C'EST ICI QUE CA SE PASSE QUAND ON CLIQUE !
    //
    if (e->button() == Qt::LeftButton)
      dt.insert(Point(x_real(e->x()), y_real(e->y())));
    if (e->button() == Qt::MidButton)
      std::cout << Point(x_real(e->x()), y_real(e->y())) << std::endl;
    if (e->button() == Qt::RightButton)
      the_point =  Point(x_real(e->x()), y_real(e->y()));
    ////////////////////////////////////////////////////////////////

    redraw();
  }
  My_layer layer;
};

int main( int argc, char **argv )
{
    QApplication app( argc, argv );
    My_window *W = new My_window(400,400);
    app.setMainWidget(W);
    W->show();
    W->set_window(0, 400, 0, 400);
    return app.exec();
}
#endif
```

```
/demo/Qt_widget/basic/tutorial3/tutorial3.C
#ifndef CGAL_USE_QT
#include <iostream>
int main(int, char*){
  std::cout << "Sorry, needs QT..." << std::endl; return 0;}
#else
#include <CGAL/Cartesian.h>
#include <CGAL/Point_2.h>
#include <CGAL/Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget_Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget.h>
#include <CGAL/IO/Qt_widget_layer.h>
#include <qapplication.h>

typedef CGAL::Cartesian<double>              Rep;
typedef CGAL::Point_2<Rep>                   Point;
typedef CGAL::Delaunay_triangulation_2<Rep> Delaunay;

.....................................................

#endif
```

```
/demo/Qt_widget/basic/tutorial3/tutorial3.C
#ifndef CGAL_USE_QT
#include <iostream>
int main(int, char*){
  std::cout << "Sorry, needs QT..." << std::endl; return 0;}
#else
#include <CGAL/Cartesian.h>
#include <CGAL/Point_2.h>
#include <CGAL/Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget_Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget.h>
#include <CGAL/IO/Qt_widget_layer.h>
#include <qapplication.h>

typedef CGAL::Cartesian<double>              Rep;
typedef CGAL::Point_2<Rep>                   Point;
typedef CGAL::Delaunay_triangulation_2<Rep> Delaunay;

.....................................................

#endif
```

```
/demo/Qt_widget/basic/tutorial3/tutorial3.C
#ifndef CGAL_USE_QT
#include <iostream>
int main(int, char*){
  std::cout << "Sorry, needs QT..." << std::endl; return 0;}
#else
#include <CGAL/Cartesian.h>
#include <CGAL/Point_2.h>
#include <CGAL/Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget_Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget.h>
#include <CGAL/IO/Qt_widget_layer.h>
#include <qapplication.h>

typedef CGAL::Cartesian<double>              Rep;
typedef CGAL::Point_2<Rep>                   Point;
typedef CGAL::Delaunay_triangulation_2<Rep> Delaunay;

.....................................................

#endif
```

/demo/Qt_widget/basic/tutorial3/tutorial3.C

```
#ifndef CGAL_USE_QT
#include <iostream>
int main(int, char*){
  std::cout << "Sorry, needs QT..." << std::endl; return 0;}
#else
#include <CGAL/Cartesian.h>
#include <CGAL/Point_2.h>
#include <CGAL/Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget_Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget.h>
#include <CGAL/IO/Qt_widget_layer.h>
#include <qapplication.h>

typedef CGAL::Cartesian<double>                Rep;
typedef CGAL::Point_2<Rep>                     Point;
typedef CGAL::Delaunay_triangulation_2<Rep>    Delaunay;

.....................................................

#endif
```

/demo/Qt_widget/basic/tutorial3/tutorial3.C

```
#ifndef CGAL_USE_QT
#include <iostream>
int main(int, char*){
  std::cout << "Sorry, needs QT..." << std::endl; return 0;}
#else
#include <CGAL/Cartesian.h>
#include <CGAL/Point_2.h>
#include <CGAL/Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget_Delaunay_triangulation_2.h>
#include <CGAL/IO/Qt_widget.h>
#include <CGAL/IO/Qt_widget_layer.h>
#include <qapplication.h>

typedef CGAL::Cartesian<double>                Rep;
typedef CGAL::Point_2<Rep>                     Point;
typedef CGAL::Delaunay_triangulation_2<Rep>    Delaunay;

.....................................................

#endif
```

```
////////////////////////////////////////////////////////
.............

int main( int argc, char **argv )
{
    QApplication app( argc, argv );
    My_window *W = new My_window(400,400);
    app.setMainWidget(W);
    W->show();
    W->set_window(0, 400, 0, 400);
    return app.exec();
}
```

```
////////////////////////////////////////////////////////
.............
```

```
int main( int argc, char **argv )
{
    QApplication app( argc, argv );
    My_window *W = new My_window(400,400);
    app.setMainWidget(W);
    W->show();
    W->set_window(0, 400, 0, 400);
    return app.exec();
}
```

Gestion de la fenêtre graphique

```
private:
  //this method is called when the user presses the mouse
  void mousePressEvent(QMouseEvent *e)
  {
    Qt_widget::mousePressEvent(e);

    ////////////////////////////////////////////////////////////
    //  C'EST ICI QUE CA SE PASSE QUAND ON CLIQUE !
    //
    if (e->button() == Qt::LeftButton)
      dt.insert(Point(x_real(e->x()), y_real(e->y())));
    if (e->button() == Qt::MidButton)
      std::cout<<Point(x_real(e->x()),y_real(e->y()))<<std::endl;
    if (e->button() == Qt::RightButton)
      the_point =  Point(x_real(e->x()), y_real(e->y()));
    ////////////////////////////////////////////////////////////

    redraw();
  }
  My_layer layer;
};
```

```
private:
  //this method is called when the user presses the mouse
  void mousePressEvent(QMouseEvent *e)
  {
    Qt_widget::mousePressEvent(e);

    ////////////////////////////////////////////////////////////
    //  C'EST ICI QUE CA SE PASSE QUAND ON CLIQUE !
    //
    if (e->button() == Qt::LeftButton)
      dt.insert(Point(x_real(e->x()), y_real(e->y())));
    if (e->button() == Qt::MidButton)
      std::cout<<Point(x_real(e->x()),y_real(e->y()))<<std::endl;
    if (e->button() == Qt::RightButton)
      the_point =  Point(x_real(e->x()), y_real(e->y()));
    ////////////////////////////////////////////////////////////

    redraw();
  }
  My_layer layer;
};
```

```
private:
  //this method is called when the user presses the mouse
  void mousePressEvent(QMouseEvent *e)
  {
    Qt_widget::mousePressEvent(e);

    ////////////////////////////////////////////////////////////
    //  C'EST ICI QUE CA SE PASSE QUAND ON CLIQUE !
    //
    if (e->button() == Qt::LeftButton)
      dt.insert(Point(x_real(e->x()), y_real(e->y())));
    if (e->button() == Qt::MidButton)
      std::cout<<Point(x_real(e->x()),y_real(e->y()))<<std::endl;
    if (e->button() == Qt::RightButton)
      the_point =  Point(x_real(e->x()), y_real(e->y()));
    ////////////////////////////////////////////////////////////

    redraw();
  }
  My_layer layer;
};
```

```
private:
  //this method is called when the user presses the mouse
  void mousePressEvent(QMouseEvent *e)
  {
    Qt_widget::mousePressEvent(e);

    ////////////////////////////////////////////////////////////
    //  C'EST ICI QUE CA SE PASSE QUAND ON CLIQUE !
    //
    if (e->button() == Qt::LeftButton)
      dt.insert(Point(x_real(e->x()), y_real(e->y())));
    if (e->button() == Qt::MidButton)
      std::cout<<Point(x_real(e->x()),y_real(e->y()))<<std::endl;
    if (e->button() == Qt::RightButton)
      the_point =  Point(x_real(e->x()), y_real(e->y()));
    ////////////////////////////////////////////////////////////

    redraw();
  }
  My_layer layer;
};
```

```
private:
  //this method is called when the user presses the mouse
  void mousePressEvent(QMouseEvent *e)
  {
    Qt_widget::mousePressEvent(e);

    ////////////////////////////////////////////////////////////
    //  C'EST ICI QUE CA SE PASSE QUAND ON CLIQUE !
    //
    if (e->button() == Qt::LeftButton)
      dt.insert(Point(x_real(e->x()), y_real(e->y())));
    if (e->button() == Qt::MidButton)
      std::cout<<Point(x_real(e->x()),y_real(e->y()))<<std::endl;
    if (e->button() == Qt::RightButton)
      the_point =  Point(x_real(e->x()), y_real(e->y()));
    ////////////////////////////////////////////////////////////

    redraw();
  }
  My_layer layer;
};
```

```
private:
  //this method is called when the user presses the mouse
  void mousePressEvent(QMouseEvent *e)
  {
    Qt_widget::mousePressEvent(e);

    ////////////////////////////////////////////////////////////
    //  C'EST ICI QUE CA SE PASSE QUAND ON CLIQUE !
    //
    if (e->button() == Qt::LeftButton)
      dt.insert(Point(x_real(e->x()), y_real(e->y())));
    if (e->button() == Qt::MidButton)
      std::cout<<Point(x_real(e->x()),y_real(e->y()))<<std::endl;
    if (e->button() == Qt::RightButton)
      the_point =  Point(x_real(e->x()), y_real(e->y()));
    ////////////////////////////////////////////////////////////

    redraw();
  }
  My_layer layer;
};
```

```
//////////////////////////////////////////////////////////
//  Les objets affiches sont declares ici en variable globales
//  l'affichage est fait dans la methode "draw" ci dessous

Delaunay dt;
Point    the_point=Point(-1,-1);

class My_layer : public CGAL::Qt_widget_layer{
  void draw(){
    *widget << CGAL::RED << dt << CGAL::PointSize(6)
                << CGAL::BLUE << the_point;
  }
};
```

Démo CGAL

Démo Doc CGAL

Démo premier TD CGAL

Démo ddd


C’est tout pour aujourd’hui