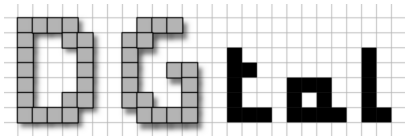


DGtal: Digital Geometry Tools and Algorithms Library

module Géométrie 2D

Module géométrie 2D

Développement

- courbes : 4-connexes
- primitives : segment de droite discrète (DSS)
- décompositions : segmentation gloutonne

Participants

- (David Coeurjolly)
- (Jacques-Olivier Lachaud)
- Bertrand Kerautret
- Isabelle Sivignon
- Tristan Roussillon

Intérêts

Intérêt collectif

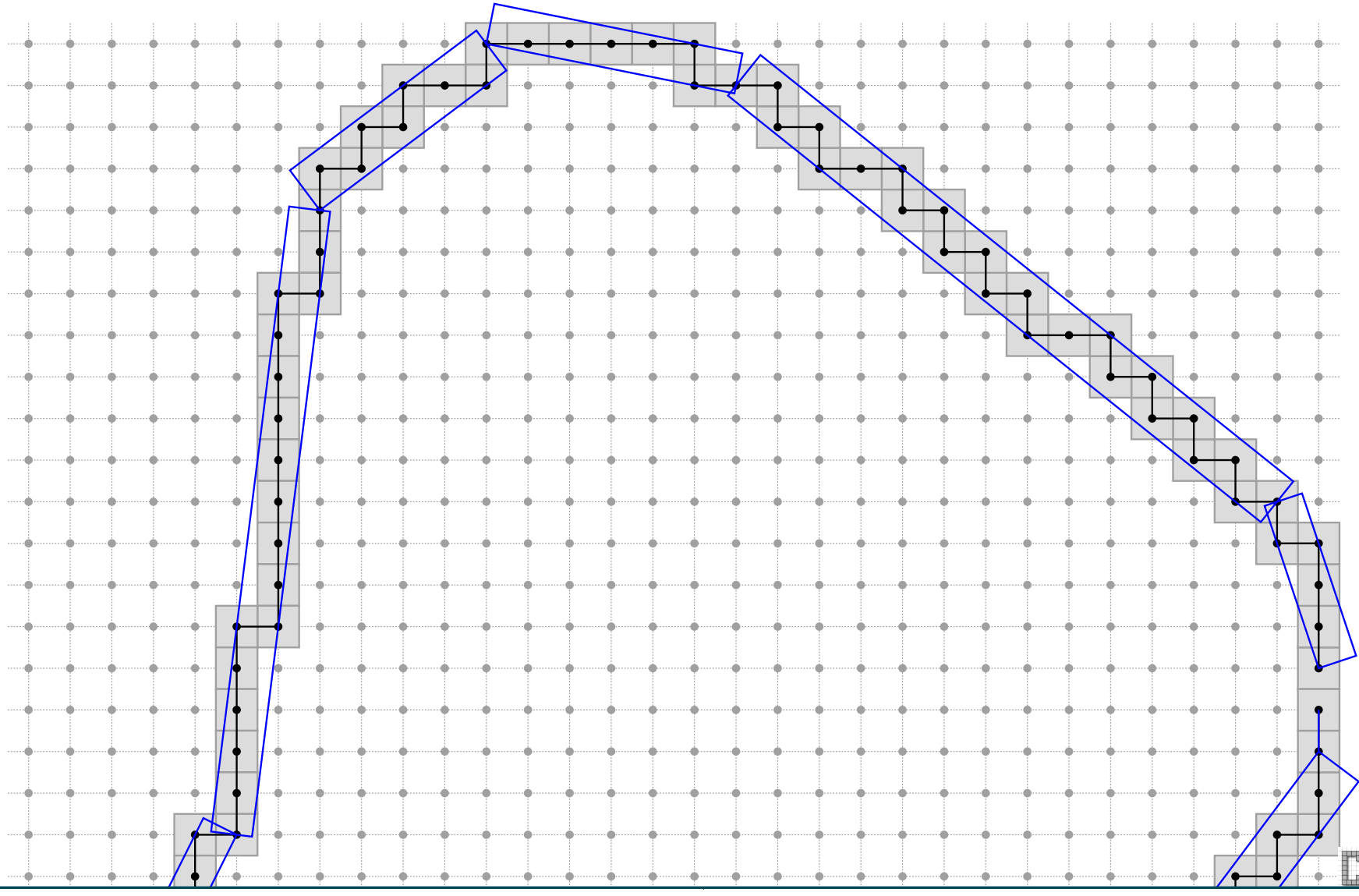
Fournir une première ébauche de code sur un cas d'école (segmentation d'une courbe en DSS)

Intérêt personnel

Me familiariser avec la bibliothèque et la programmation générique en implémentant un algorithme assez simple.

Exemple (greedy-dss-decomposition.cpp)

http://liris.cnrs.fr/dgtal/doc/nightly/



Code

greedy-dss-decomposition.cpp

```
  trace.beginBlock ( "Example dgtalboard-5-greedy-dss" );

  typedef ArithmeticalDSS<StandardBase<int> > DSS;
  typedef FreemanChain<int> ContourType;
  typedef GreedyDecomposition< ContourType, DSS > Decomposition;

  // A Freeman chain code is a string composed by the coordinates of the first pixel, and
the list of elementary displacements.
  std::stringstream ss(stringstream::in | stringstream::out);
  ss << "31 16
1112121212122121212122121222223223232333233333333233333233033303300303000010001001001000
<< endl;

  // Construct the Freeman chain
  ContourType theContour( ss );
  //Segmentation
  Decomposition theDecomposition( theContour );
```

Code

greedy-dss-decomposition.cpp

```
Point p1( 0, 0 );
Point p2( 31, 31 );
Domain domain( p1, p2 );
DGtalBoard aBoard;
aBoard << SetMode( domain.styleName(), "Grid" )
 << domain
 << theContour;
//for each segment
DSS segment;
aBoard << SetMode( segment.styleName(), "BoundingBox" );
string styleName = segment.styleName() + "/BoundingBox";
for ( Decomposition::ConstIterator i = theDecomposition.begin();
i != theDecomposition.end(); ++i )
  {
    segment = *i;
    aBoard << CustomStyle( styleName,
         new CustomPenColor( DGtalBoard::Color::Blue ) )
     << segment; // draw each segment
  }
aBoard.saveSVG("dgtalboard-5-greedy-dss.svg");
aBoard.saveSVG("dgtalboard-5-greedy-dss.eps");

trace.endBlock();
```

Code

GreedyDecomposition.ih

```
myBack = myFront;
++myFront;
myP = Primitive(*myBack,*myFront);

//while my primitive can grow
++myFront;
while ( (myFront != myC->end()) &&
        (myP.addFront(*myFront)) ) {
  ++myFront;
}
//
if (myFront != myC->end()) --myFront;
```

Code

ArithmeticalDSS.ih

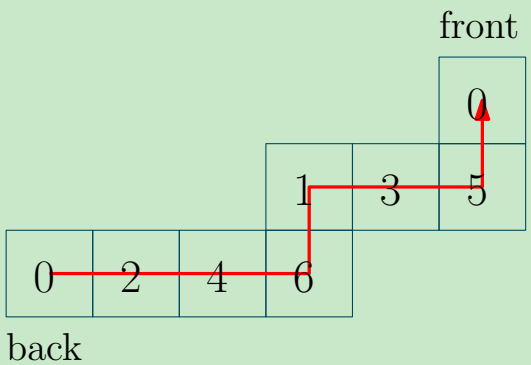
```
//remainder
Integer r = myA*aPoint[0] - myB*aPoint[1];

if ( (r < myMu-1)||(r > myMu+myOmega) )
  return false; //strongly exterior
else {

  //add aPoint to the DSS
  myL = aPoint;
  //leaning points update
  //weakly interior
  if (r == myMu) myUl = aPoint;
  if (r == myMu+myOmega-1) myLl = aPoint;
  //weakly exterior
  if (r == myMu-1) {
    myUl = aPoint;
    myLf = myLl;
    myA = myUl[1] - myUf[1];
    myB = myUl[0] - myUf[0];
    myMu = myA*myUl[0] - myB*myUl[1];
    myOmega = TDSS::norm(myA,myB);
  } else if (r == myMu+myOmega) {
    myLl = aPoint;
    myUf = myUl;
    myA = myLl[1] - myLf[1];
    myB = myLl[0] - myLf[0];
    myMu = myA*myUl[0] - myB*myUl[1];
    myOmega = TDSS::norm(myA,myB);
  }
  return true;

}
```

Choix algorithmiques

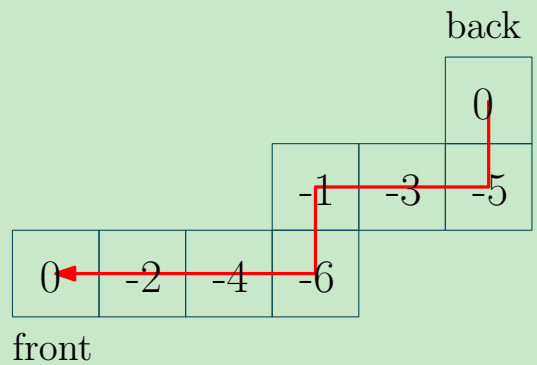
Algorithme de Debled et Reveillès (1995)

- caractéristiques a, b, μ, ω
- 4 points d'appui
- premier et dernier points
- pas de changement d'octant, mais une orientation

(2,5,0,7)



(-2,-5,-6,7)



Choix d'implémentation

Paramètres templates

- Entier (Domaine ?)
- Connexité {naïf, standard}

Méthodes principales

- Initialisation à partir de deux points (un seul point ?)
- ajout d'un point à l'avant
  - *addFront*
  - prends un point comme paramètre en entrée (un vecteur, un caractère ?)
  - retourne un booleen (V si DSS, F sinon)
- retrait d'un point à l'arrière
  - *removeBack*
  - pas de paramètre en entrée
  - retourne un booleen (V s'il reste plus de 2 points, F sinon)

Bilan

Définition des concepts (à discuter)

- courbe : liste (circulaire) de points (ordre)
  - fournit un itérateur sur les points (ou les vecteurs de déplacement entre deux points consécutifs ?)
- primitive : courbe vérifiant une propriété donnée
  - initialisation (un point plutôt que deux pour que la propriété soit toujours vérifiée)
  - ajout à l'avant (un point, un vecteur déplacement ?), retourne V si la propriété reste vraie (et le point est ajouté), F sinon (et le point n'est pas ajouté pour que la propriété reste vraie)
  - ...
- décomposition : liste (circulaire) de primitives recouvrant une courbe (ordre)
  - fournit un itérateur sur les primitives

Perspectives

Regarder ce qu'on a fait

- Améliorations de *FreemanChain*, *ArithmeticalDSS*, *GreedySegmentation* (si besoin)
- Ecriture des concepts de courbe, primitive et décomposition (si on est d'accord).

Aller plus loin...

- D'autres courbes (8-connexes ?)
  - discrétisation de courbes euclidiennes
  - extraction du bord de régions connexes
- D'autres primitives (il y a le choix!)
- D'autres décompositions (couverture)