Introduction to DGtal and its Concepts

http://liris.cnrs.fr/dgtal

David Coeurjolly

DGtal: why, who

Objectives

- to make digital geometry easier for the neophyte (student, researcher from another field, . . .)
- to quickly test new ideas, with objective comparison wrt existing works
- to make easier the implementation of demonstrators
- to help spread our research results to other domains

⇒ Federative Project

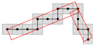
Who ? (for now) . . .

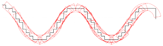
- LIRIS (Lyon)
- Gipsa-lab (Grenoble)
- GREYC (Caen)
- LAMA (Chambéry)
- LORIA (Nancy)
- IRCCyn (Nantes)

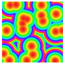
DGtal: what for ?

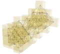
Main features

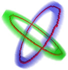
- to define digital objects in arbitrary dimension
- to propose algorithms for topological and geometric analysis
- to provide I/O mechanisms and visualization tools

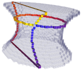

DSS

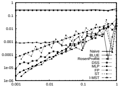

DCA


DT

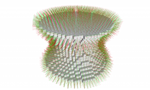

Objects

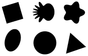

Thinning


Cellular model

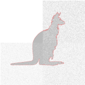


Estimators


normal vectors


Shape DB

polynomial surfaces


Contours

...

DGtal philosophy and structure

- Genericity and efficiency: C++ library, concepts
- LGPL
- Cmake build system (linux/macOS/MSwindows)
- user friendly, not necessarily kernel-developer friendly

Kernel Package

- Digital space
- Point, vectors
- Digital domains and digital sets
- ...

Arithmetic Package

- Fractions
- Irreducible fractions
- DSS Pattern..
- ...

DGtal philosophy and structure

Topology Package

- Digital Topology: connectedness, border, simple points (*a la* Rosenfeld)
- Cartesian Cellular Topology: cells, surfaces and contours (*a la* Herman), tracking algorithms
- Digital Surface concepts and models

Geometry Package

- Primitives (*a.k.a.* SegmentComputers): DSS, DCA,...
- Contour analysis: decomposition, convexity, estimators
- Volumetric analysis: area/volume, distance transforms, reverse distance transforms, Fast-marching methods.
- Implicit/parametric shape generator for multigrid analysis

Math Package

- Representation of polynoms
- ...

DGtal philosophy and structure

Image Package

Image concept and Image containers, e.g.

- Image by STL `vector` (linearized nD image)
- Image by STL `map` (mapping points↔values)
- HashTree image container (generalized octree with hashing functions)

IO Package

- `Boards`: export to illustrate objects/algorithms (eps,pdf,svg,png,tikz. . .)
- `Viewers`: simple 3D viewer (Qt/QGlViewer)
- Readers/writers for various image formats

DGtal 0.5.1

- Project started in Jan 2010
- 200k lines of code
- *env*. 557 C++ classes
- Used in couple of research projects (ANR digitalSnow, collaboration with Chemical lab in Lyon, collaboration with Agricultural in Nancy,...)

DGtal Team

DGtal

http://liris.cnrs.fr/dgtal

LIRIS

D. Cœurjolly
G. Damiand
M. Tola

LAMA
Laboratoire de Mathématiques
Université de Savoie

J.-O. Lachaud
X. Provençal
T. Roussillon

Loria
Laboratoire lorrain de recherche
en informatique et ses applications

B. Kerautret

GREYC

S. Fourey

gipsa-lab

I. Sivignon

IRCCyN

N. Normand

DGtal principles

Generic Programming

- Data structures $\perp$ Algorithms
- Concepts, models of concepts and concept checking

⇒ C++ with template programming

Concepts ?

Way to ensure (or to describe) that a type (class) satisfies some constraints (syntactically or semantically).

- At design level: very helpful to enhance separability data/algorithms
- At implementation level: concept checking tools to verify that a given type validate a concept

DGtal program skeleton

```cpp

#include "DGtal/base/Common.h"
#include "DGtal/kernel/SpaceND.h"
#include "DGtal/kernel/domains/HyperRectDomain.h"
...
typedef DGtal::int32_t Integer;
typedef DGtal::SpaceND<3, Integer> Space3;
typedef Space3::Point Point;
typedef HyperRectDomain<Space> Domain;

Point p(12, -34,0);
Point q(2, -2, -1);
if (p < q)
  ...

Domain box(p,q);
....
```

DGtal program skeleton

or even simpler with standard definitions:

```

#include "DGtal/base/Common.h"
#include "DGtal/helpers/StdDefs.h"
...
DGtal::Z3i::Point p(12, -34,0);
DGtal::Z3i::Point q(2, -2, -1);
if (p < q)
  ...

DGtal::Z3i::Domain box(p,q);
....
```

DGtal program skeleton (again)

Things to do

1. Fix the dimension
2. Fix the Integer type (commutative ring (+,-,*))
3. Define the digital space DGtal::SpaceND

```
#include "DGtal/base/Common.h"
#include "DGtal/kernel/SpaceND.h"
{...}
typedef DGtal::int32_t Integer;
typedef DGtal::SpaceND<6, Integer> Space6;

typedef mpz_class IntegerGMP;  //mpz_class == DGtal::BigInteger
typedef DGtal::SpaceND<6, IntegerGMP> Space6GMP;
```

Q: what’s wrong with ?

```
typedef DGtal::SpaceND<2, unsigned char> MySpaceUChar;
```

[DETAILS] Concept & Models

Answer

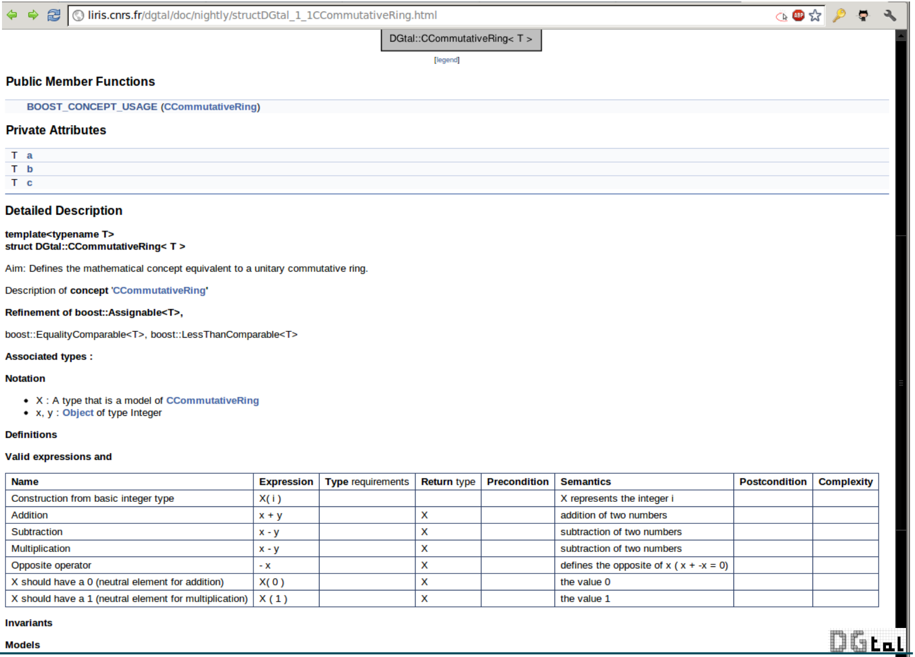
unsigned char does not define a ring !

Constraints on types and template parameters are defined with **Concepts**

Integer in SpaceND should be a model of DGtal::CCommutativeRing.

Concept Checking with boost

```
...
//Integer must be signed to characterize a ring.
BOOST_CONCEPT_ASSERT(( CCommutativeRing<TInteger> ) );
...
```


liris.cnrs.fr/dgtal/doc/nightly/structDGtal_1_1CCommutativeRing.html
DGtal::CCommutativeRing< T >
[legend]

Public Member Functions

BOOST_CONCEPT_USAGE (CCommutativeRing)

Private Attributes

T	a
T	b
T	c

Detailed Description

template<typename T>
struct DGtal::CCommutativeRing< T >

Aim: Defines the mathematical concept equivalent to a unitary commutative ring.

Description of **concept** 'CCommutativeRing'

Refinement of boost::Assignable<T>,

boost::EqualityComparable<T>, boost::LessThanComparable<T>

Associated types :

Notation

- X : A type that is a model of CCommutativeRing
- x, y : Object of type Integer

Definitions

Valid expressions and

Name	**Expression**	**Type** requirements	**Return** type	**Precondition**	**Semantics**	**Postcondition**	**Complexity**
Construction from basic integer type	X(i)				X represents the integer i		
Addition	x + y		X		addition of two numbers		
Subtraction	x - y		X		subtraction of two numbers		
Multiplication	x - y		X		subtraction of two numbers		
Opposite operator	- x		X		defines the opposite of x (x + -x = 0)		
X should have a 0 (neutral element for addition)	X(0)		X		the value 0		
X should have a 1 (neutral element for multiplication)	X (1)		X		the value 1		

Invariants

Models

Main DGtal objects/concepts in one slide

CSpace : where all your computations lie, provides you an algebra

CPositiveIrreducibleFraction : well.. you get the idea...

CDomain : provides you ways iterate on points (classical model: `HyperRectDomain`)

CDigitalSet : containers of a collection of digital points, provides you iterators, insert/delation methods,...

Object : union of a digital topology and a digital set (neighborhood , connected components, simple points test, ...)

CDigitalSurface{Container,Tracker} : models to construct/track digital surfaces

CSegment : given a 2D generic contour, models which associate a “property” to a part of it

CSegmentComputer : refinement of CSegment whose models provides methods to “recognize” part of the curve satisfying the “property” (*e.g.* DSS, DCA, ...)

CImage : models which associate values to point in a domain.

Board2D, Viewer3D, Board2DTo3D : viewers, exporters,...

Example using Image concepts

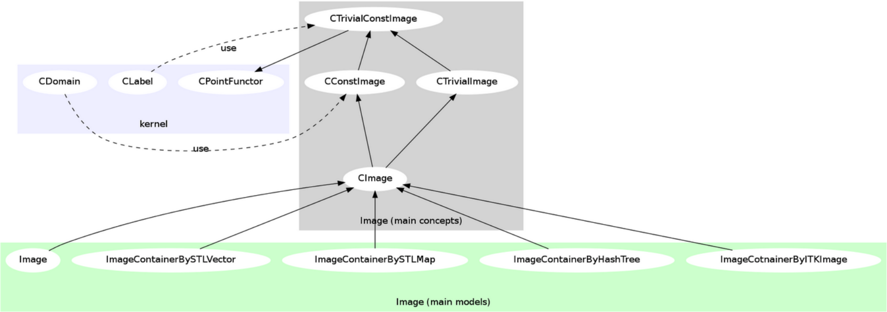


Image vs. ImageContainers

```
#include "DGtal/base/Common.h"
#include "DGtal/helpers/StdDefs.h"
{...}
using namespace DGtal;

typedef double Value;
typedef ImageContainerBySTLVector<Z3i::Domain, Value> ↩
     ImageContainer;
typedef Image<ImageContainer> LightImage;

Z3i::Point p(0,0,0),q(100,100,100);
Z3i::Domain domain(p,q);

LightImage myImage ( new ImageContainer(domain) );

myImage.setValue( Z3i::Point(42,42,42) , 42 );

Value b  = myImage( Z3i::Point(42,42,42) ); //b == 42;)

LightImage myImage2 = myImage;  //Nothing copied in memory (↩
     CopyOnWrite)

myImage2.setValue( Z3i::Point(1,1,1), 1) ; //ok, copying now..
```

Iterating on image values

Key concept: CxxxRange

```
...
LightImage myImage ( new ImageContainer(domain) );
typedef LightImage::Range::Iterator Iterator;
typedef LightImage::Domain::ConstIterator DomainConstIterator;


//Setting values iterating on the domain points
for(DomainConstIterator it = myImage.domain().begin(), itend =↩
      myImage.domain().end();
    it != itend; ++it)
  myImage.setValue( *it , 42 );  // (*it) is a Point


//Fast init of the image using container built-in iterator
for(Iterator it = myImage.range().begin(), itend = myImage.↩
     range().end();
    it != itend; ++it)
  *it =  42 ; // (*it) is a container cell
```

Image::Range (R/W bidirectional range), Image::ConstRange (Read-only bidirectional const range), ...

Switching from `ImageContainerBySTLVector` to `ImageContainerBySTLMap` won't change your code ! (just the performances)

DGtal Meeting Program

- 9h30: Introduction to DGtal and its concepts
- 10h00: Digital Surfaces in DGtal
- 10h30: Irreducible fractions, patterns and straightness in DGtal
- 11h00: <Coffee break>
- 11h20: Representation and analysis of digital curves
- 11h50: DGtal boards and viewers
- 12h20: DGtal & DGtalTools project management
- 12h50: Visit of the IVC Research team
- 13h10: <Lunch>
- 14h30: Feedback on my DGtal Experience (Isabelle Sivignon)
- 15h00: Install party, use-case discussions, future plans,...