



## Introduction to DGtal and its Concepts

<http://liris.cnrs.fr/dgtal>

David Coeurjolly

## Digital Geometry

### 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

# 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



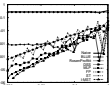
Objects



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 (*à la* Rosenfeld)
- Cartesian Cellular Topology: cells, surfaces and contours (*à 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 $\leftrightarrow$ 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/QGViewer)
- 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 INRA at Nancy,... )

# 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 validates a concept



# DGtal program skeleton

```
1
2     #include "DGtal/base/Common.h"
3     #include "DGtal/kernel/SpaceND.h"
4     #include "DGtal/kernel/domains/HyperRectDomain.h"
5     ...
6     typedef DGtal::int32_t Integer;
7     typedef DGtal::SpaceND<3, Integer> Space3;
8     typedef Space3::Point Point;
9     typedef HyperRectDomain<Space3> Domain;
10
11     Point p(12, -34,0);
12     Point q(2, -2, -1);
13     if (p < q)
14         ...
15
16     Domain box(p,q);
17     ....
```

## DGtal program skeleton

or even simpler with standard definitions:

```
1
2     #include "DGtal/base/Common.h"
3     #include "DGtal/helpers/StdDefs.h"
4     ...
5     DGtal::Z3i::Point p(12, -34, 0);
6     DGtal::Z3i::Point q(2, -2, -1);
7     if (p < q)
8         ...
9
10    DGtal::Z3i::Domain box(p, q);
11    ....
```

# 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

```
1  #include "DGtal/base/Common.h"
2  #include "DGtal/kernel/SpaceND.h"
3  {...}
4  typedef DGtal::int32_t Integer;
5  typedef DGtal::SpaceND<6, Integer> Space6;
6
7  typedef mpz_class IntegerGMP; //mpz_class == DGtal::↔
   BigInteger
8  typedef DGtal::SpaceND<6, IntegerGMP> Space6GMP;
```

Q: what's wrong with ?

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

## [DETAILS] Concept & Models

### Answer

`unsigned char` does not define a ring !

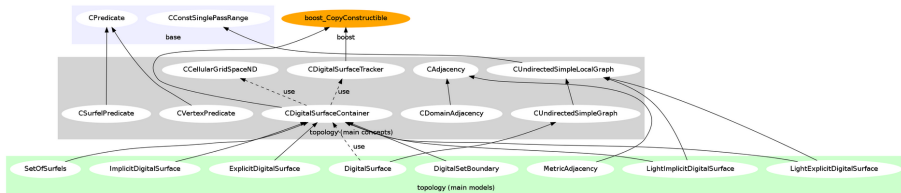
Constraints on types and template parameters are defined with **Concepts**

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

Concept Checking with `boost`

```
1    ...  
2    //Integer must be signed to characterize a ring.  
3    BOOST_CONCEPT_ASSERT(( CCommutativeRing<TInteger> ) );  
4    ...
```

## Example using Image concepts



## 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

# 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

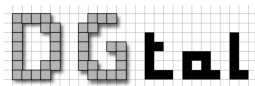
**CSegment** : given a 2D generic contour, models which associate a “property” to a part of it

**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, Board3DTo2D** : viewers, exporters,...

# DGtal Team



<http://liris.cnrs.fr/dgtal>  
<http://github.com/DGtal-team>



D. Coeurjolly  
G. Damiand  
M. Tola



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



B. Kerautret



S. Fourey



I. Sivignon



N. Normand

