TD5: DGtal, Tracing and Border Extraction

In this TP, we introduce `DGtal` and do some elementary border processing.

The `DGtal` Project

`DGtal` is an open-source library focusing on digital geometry tools (`http://dgtal.org`). In this TD, we will only use the DGtal board framework to display elementary digital objects.

`DGtal` has been installed in `/home/dcoeurjo/DGtal/`, you will just have to specify this path in order to compile something using the library.

- Get the DGtal `cmake` project skeleton:

  ```
  git clone  https://github.com/dcoeurjo/lectureDG.git
  ```

- Go to `practical-work/` folder.

- Compile the examples using `cmake`[1] "out-of-source-build principle":

  ```
  cd DGtalSkel
  mkdir build && cd build && echo "Temporary build folder created"
  cmake .. -DDGtal_DIR=/home/dcoeurjo/DGtal/build
  make
  ```

  The `make` command will build examples located in the source folder (path ".." here). If you want to clean your build, just remove the `build` folder (a "make clean" command also exists in the build folder).

- Execute: in the build folder, you should have `simpleboard` executable. Run it and you should see several new test.* files (EPS, SVG, ...).

- Have a look to `simpleboard.cpp`, `testSet.cpp` and the EPS files they generates. You will see a very simple description of elementary DGtal objects that will be used here (`Board2D`, `Point` and `DigitalSet`).

- To add a new C++ file, copy and rename one of the example and add its name into the `CMakeLists.txt` file (`SRCs` variable). In the build folder, the next `make` command will build the brand new file.

- If you want to install DGtal and DGtalSkel on your own machine, please see below.

Exercise 1 Chessboard

Create a program that will generate (EPS file) a black/white 16x16 chessboard.

[1] `cmake` is a project generator (e.g. makefiles on Unix, Xcode projects on MacOS, VisualStudio projects on MS) from a very simple recipe file (see `CMakeLists.txt`)

Exercise 2 Bresenham DSS Tracing

Questions:

- Let us consider a point (x, y) with $0 < y < x$ and the Bresenham's digitization of the Euclidean segment defined by the origin and the point. Build a `DigitalSet` corresponding to this segment (cf `testSet.cpp` for a demo on `DigitalSet`) and display the result in a EPS file (using `board`). You can use the version of the algorithm you want.
- Customize this function to handle any point position. Again, display an example.
- **(Optional)** We would like to have a Bresenham-like tracing function to display digital circle. The idea is similar to the DSS case: in the first octant ($0 < y < x$), start from the point $(r, 0)$ and use the previous point (x, y) to decide the next one ($(x - 1, y + 1)$ or $(x, y + 1)$). By symmetry of the circle, generalize the first octant circular arc to the other ones. Similarly to the DSS tracing, try to optimize your code to avoid all floating point operations/comparisons.

Exercise 3 Border extraction

Questions:

- In a new program, create a function that construct a digital disc, parametrized by a center and a radius. Generate the Gauss digitization of the associated Euclidean disc.
- Given such a disc digital set, and given a (κ, λ)-Jordan pair, iterate over the points of the set and detect the pixels belonging to the border (color them differently in the board output).
- Make the border detector a bit more efficient in the following ways:
  - Iterate over the domain to get a first point in the set which belongs to the border.
  - Look at local neighborhood around this point, decide which pixel would be the next one in the border and iterate (you'll have to fix an orientation). Thanks to the orientation and to make this step as efficient as possible, try to reduce the number of points to "probe" in the neighborhood.
  - Repeat the tracking until you have closed your border (i.e. your have found your starting point)

Install DGtal on your own machine

If you want to install DGtal on your machine, you will need development packages for `boost` and `boost-programs_option`. Then, get the code and compile the lib:

```
git clone https://github.com/DGtal-team/DGtal.git
cd DGtal && mkdir build && cd build
cmake .. -DDGTAL_BUILD_TESTING=OFF
make
```

Then, for the DGtalSkel `cmake`, replace the `/home/dcoeurjo/DGtal/build` by your DGtal build folder.