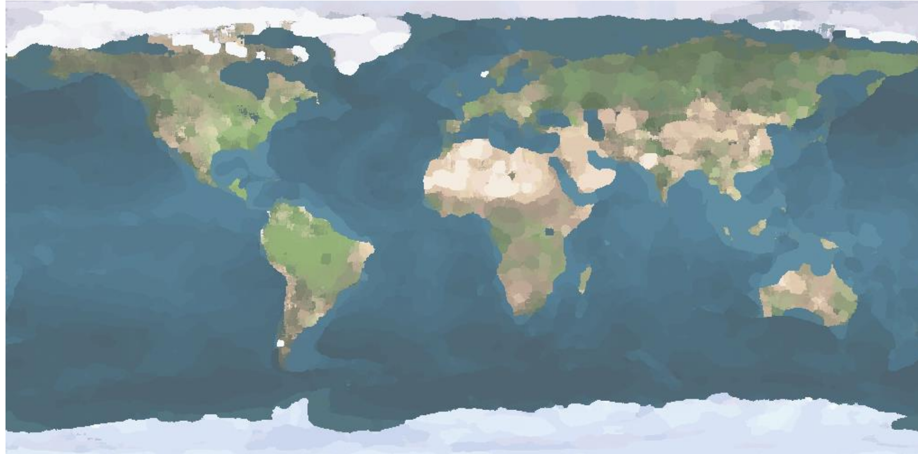
GPU Programming

TP3 – Oil Painting



We want to apply a special filter to an image which makes it look like an oil painting.

Discover and execute the provided base code. It uses the pnglite library which again depends on the zlib library for reading and writing png image files (zlib is the current standard for lossless data compression). If zlib is not installed on your system, just copy the provided file zlib1.dll to C:\Windows\System32 and C:\Windows\SysWOW64.

The sample filter "swap_GB" swaps the blue and the green component of each pixel. Before attacking the oilpainting filter, you may want to warm up by creating some other simple ones, e.g.

- invert all pixel colors
- convert image to gray scale
- binarize the image (black and white pixels only)
- flip the image vertically or horizontally
- …

Now let's look into the oilpainting algorithm. The algorithm is explained here:
http://supercomputingblog.com/graphics/oil-painting-algorithm/

Exercices (1 point per question)

1) Write a CPU version of the algorithm.

2) Write a (naive) GPU version: each thread is assigned to one pixel and calculates the resulting pixel color by browsing the adjacent pixels of the source image in the global GPU memory. Use 2D block- and thread-indexing, i.e. tile the image into blocks of NxN threads (N*N<=1024), for example:

```
dim3 nbT (20,20);
dim3 nbB ((width+nbT.x-1)/ nbT.x,(height+nbT.y-1)/ nbT.y);
oilpainting<<<nbB, nbT>>>(img_in, img_out, ...);
```

3) Write an optimized GPU version using <u>shared memory</u>: each block of threads loads a tile of the source image into shared memory, so that the threads can browse the pixel data in the shared memory instead of the global GPU memory. Note that the GPU blocks have to be larger than the processed pixel tiles because the oil painting algorithm requires the data of each pixel's neighbors.