**OpenCL, assignment 4**

# Introduction

In Assignment 2, you made a kernel for the Mandelbrot fractal. In that assignment, we displayed that fractal by transporting the generated image to the host memory, then storing it in a file, and finally opening it with MS Paint.   
This was a very clumsy way to display an image, not suitable at all for showing animations.

In this assignment, we will keep the image in the video memory, and show it directly on the screen using OpenGL. This is much faster, and makes it possible to zoom in on the fractal dynamically.

# Making OpenCL and OpenGL cooperate

The task for this week is to get the cooperation between OpenCL and OpenGL working, and then call the Mandelbrot kernel repeatedly with each time a higher zoom-factor (or smaller stepsize).   
This will generate an animated zoom-in on the fractal, with amazing effects.

What do we have to change/add to the existing week 2 program to reach this goal?

* In order to be able to use OpenGL as easily as possible, we will use a library which is called GLUT (OpenGL Utility Toolkit). The original version of this toolkit is not maintained anymore by its creator, but other people took over, and created (and still maintain) FreeGLUT.   
  We will use this version; download release 2.8.1 from <http://freeglut.sourceforge.net/index.php#download>.  
  In the downloaded archive, you can find a Visual Studio solution (actually 3 of them: for VS2008, VS2010 and VS2012). With this project, you can build the required freeglut.lib (needed by the linker) and freeglut.dll (needed at runtime).  
  freeglut.lib has to be added to the Linker section in the properties of the Mandelbrot project, and freeglut.dll can be copied to the debug folder of that project.  
  You will also need to add the path to the freeglut include folder to the C/C++ section of the project properties.  
  OpenGL will probably be usable without doing extra settings in your project.  
  To get acquainted a bit with OpenGL and GLUT you are invited to do the following tutorial up to and including paragraph 3.6:  
  <https://www3.ntu.edu.sg/home/ehchua/programming/opengl/CG_Introduction.html>.
* Now create the GLUT window right at the beginning of the main function.   
  Also create the (yet empty) display function. The glutMainLoop has to be placed at the end of the main function.
* For our purposes, we can initialize OpenGL with the function init\_gl, which is given on SharePoint (OpenGL\_functions.cpp and OpenGL\_functions.h).  
  This function must be called after creating the GLUT window.
* Of course, you must re-use as much as possible the code from week 2 where OpenCL is initialized and the kernel is build and created. The code where the kernel is started (clEnqueueNDRangeKernel) must be moved to another location: the kernel should not run once at startup of the program, but each time when the frame is redisplayed.
* Inside the kernel, the texture has to be specified as a parameter of type image2d\_t (and not anymore as mandelbrot \* as we did in the previous version of week 2).   
  Change the kernel accordingly (use write\_imagef instead of array indices).
* In the OpenGL\_functions, there is also a function for drawing a quad with the texture mapped on it. Call this function at the appropriate moment (after the kernel is finished).

At this point, you should be able see the fractal in the window, yet without special effects like zooming. In the following steps, we are going to add zooming.

* To implement the zooming effect, the stepsize has to be decreased independent of the framerate. You can use the function glutGet(GLUT\_ELAPSED\_TIME) for that; this functions returns the elapsed time in millseconds since GLUT was started.   
  With the following code, the stepsize decreases with a factor ZOOMSPEED per 100 milliseconds. ZOOMSPEED is supposed to be a constant between 0 and 1; the closer to 1, the slower the zooming in occurs.   
  This code must be done in every frame:  
   now = glutGet(GLUT\_ELAPSED\_TIME);  
   stepsize \*= pow(ZOOMSPEED,(now-previous)/100.0);  
   previous = now;  
  Don’t forget to call clSetKernelArg each time again before the kernel is started.
* Choose an interesting point where you want to zoom in on, for example   
  (-0.251995001, 0.0001429999).
* Don’t forget that, because we use OpenGL, the color-components have to be represented by floats between 0 and 1 and not by integers between 0 and 255.   
  This might mean that your call to write\_imagef has to be changed a bit.

You are now far enough to pass this assignment but there are plenty of opportunities to increase the wow-factor:

* To get nicer colors, we need a bigger color table. You can make your own colortable, or use the function given in the file color\_table.cpp (which is not really spectacular).   
  Set MAX\_ITERATIONS and COLORTABLE\_SIZE to the same value, for example 2048.   
  The higher that value is, the less black (and the more color) there will be in the fractal, but also the more calculation time is needed.
* Find other interesting points to zoom in on.
* It would be nice if you can make it such that the animation zooms in on the point where the mouse is.
* You can try to use doubles instead of floats in the kernel; this will probably slow down the framerate a lot, but also enable you to zoom in much deeper.

Add any more features that you feel like.

Show the end result to the teacher.