Project Seminar:
Parallel and Distributed Systems

Assignment 6 (Submission deadline: Mar 2nd 2015, 13:30 CET)

General Rules

The assignment solutions have to be submitted at:

https://www.dcl.hpi.uni-potsdam.de/submit/

Our automated submission system is intended to give you feedback about the validity of your file upload. A submission is considered as accepted if the following rules are fulfilled:

- You did not miss the deadline.
- Your file upload can be decompressed with a zip / tar decompression tool.
- Your submitted solution contains only the source code files and a Makefile. Please leave out any Git / Mercurial repository clones or SVN / CVS meta-information.
- Your solution can be compiled using the "make" command, without entering a separate sub-directory after decompression.
- You program runs without expecting any kind of keyboard input or GUI interaction.
- **There will be no validation script for this assignment.**

If something is wrong with your submission, you will be informed via email (console output, error code). Re-uploads of corrected solutions are possible until the deadline.

All tasks must be submitted accordingly in order to pass the assignment.

Assignment 6

One of the unsolved challenges in leveraging heterogeneous resources is to write portable code. In theory, OpenCL is supposed to provide the means for implementing portable code for arbitrary kinds of hardware. However, with the current landscape of diverse hardware, this is close to impossible.

The sixth assignment covers programming for heterogenous compute hardware using OpenCL. You will study the basics of OpenCL and implement an image-generating algorithm. You may choose between the Mandelbrot set or a Noise Generator such as Worley Noise. Besides Perlin Noise and Simplex Noise, Worley Noise[1] aka "Cell Noise"[2] is one of the fundamental ways to create beautiful realistic textures for games and simulations[3]. We ask you to choose one algorithm and implement it for three hardware targets: Desktop-GPUs, Mobile GPUs and exotic accelerator hardware. Please choose your algorithm wisely, you should be able to fully comprehend the detailed effects of each single line of code, so that you can squeeze the best performance out of your OpenCL kernels.

You have to solve the given programming exercises in C/C++ and using OpenCL[4].

[1] If you want to toy around with a noise generator, check out: http://aftbit.com/cell-noise-2/
[2] https://code.google.com/p/fractalterraingeneration/downloads/detail?name=CellNoise.1.0.cpp
[3] http://www.no-mans-sky.com/ is a game that entirely relies on such procedural generation methods
[4] https://www.khronos.org/opencl/

Task 6.1 Desktop-grade GPUs

Please implement the image-generating algorithm of your choice (see Assignment 6 header text). The target platform for this Task is the integrated R7-GPU of an AMD A10-7870K APU (Spectre architecture) and/or the dedicated FirePro W8100 GPU (Hawaii architecture). Since this hardware provides full support of OpenCL 2.0, we encourage you to explore the new capabilities, such as support for the Heterogeneous System Architecture (HSA) standard. We recommend you to consult the hardware specifications and the AMD APP SDK Documentation for further details:

http://www.amd.com/en-gb/products/processors/desktop/a-series-apu
http://developer.amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-app-sdk/

Using the familiar credential scheme "firstname.lastname" and "pvprog" as your initial password, you can develop and test your implementation on the following machine:

```
gpu-machine (172.16.64.161)
```

In addition to the Please use the program located in the `devices` folder to familiarize yourself with the hardware. Very detailed information is provided by the `clinfo` command-line utility.

Input

Your application has to be named "**imggenAMD**" and accept **two** parameters: the *width* and the *height* of the image that is produced.

```
Example: imggenAMD 3840 2160
```

Output

The program must terminate with exit code 0 and produce a bitmap file with the name "output.bmp" in the same directory.

Task 6.2 Mobile GPUs

Please implement the image-generating algorithm of your choice (see Assignment 6 header text). The target platform for this Task is the Mali-T628 MP6 GPU employed in the Odroid XU-4. Since the GPU-design drastically differs from desktop GPUs, you should consult the hardware specifications and the Mali OpenCL SDK Documentation for further details:

http://www.arm.com/products/multimedia/mali-gpu/high-performance/mali-t628.php
http://malideveloper.arm.com/downloads/deved/tutorial/SDK/opencl/index.html

Using the familiar credential scheme "firstname.lastname" and "pvprog" as your initial password, you can develop and test your implementation on the following Odroid XU-4 machines:

```
odroid03 (172.16.64.153)
odroid04 (172.16.64.154)
```

Please use the program located in the `devices` folder to familiarize yourself with the hardware.

Input

Your application has to be named "**imggenMALI**" and accept **two** parameters: the *width* and the *height* of the image that is produced.

```
Example: imggenMALI 3840 2160
```

Output

The program must terminate with exit code 0 and produce a bitmap file with the name "output.bmp" in the same directory.

Task 6.3 Exotic accelerator hardware

Please implement the image-generating algorithm of your choice (see Assignment 6 header text). The target platform for this Task is the Epiphany E16G301 16-core co-processor employed in the Parallella Boards. Since this accelerator approach is yet entirely different and does not possess the properties of a GPU, you should consult the E16G301 Datasheet and the Epiphany Architecture Reference Manual for further details:

http://www.adapteva.com/docs/e16g301_datasheet.pdf
http://adapteva.com/docs/epiphany_arch_ref.pdf

Using the familiar credential scheme "firstname.lastname" and "pvprog" as your initial password, you can develop and test your implementation on the following Parallella machines:

```
parallella03 (172.16.64.143)
parallella04 (172.16.64.144)
```

Please use the program located in the `devices` folder to familiarize yourself with the hardware.

Input

Your application has to be named "**imggenEPIPHANY**" and accept **two** parameters: the *width* and the *height* of the image that is produced.

```
Example: imggenEPIPHANY 3840 2160
```

Output

The program must terminate with exit code 0 and produce a bitmap file with the name "output.bmp" in the same directory.