TSE6-OCL Assignment 5

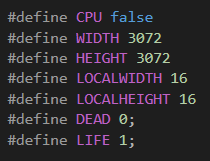
## A short description of the problem you solved.

We challenged ourselves with making a GPU and CPU implementation of the well-known “Game of Life” by John Conway. In this ‘game’, a pixel in a two-dimensional grid can be either alive or dead, depending on its (living) neighbors. The game is based on four simple rules:

* If the pixel is alive and it has no neighbors, it dies of loneliness.
* If the pixel is alive and it has four or more neighbors, it dies of over population.
* If the pixel is alive and it has two or three neighbors, it survives.
* If the pixel is dead and it has three neighbors, it pops into life.

Because the state of every pixel needs to be determined individually, largely irrelevant to the other pixels (apart from its eight adjacent pixels), this problem could benefit a lot from parallelization using GPU computation. We set out to prove this and implement the “Game of Life” to run on both CPU and GPU.

The resulting Visual Studio project is attached in a zip file. To set the game to run on CPU or GPU and to edit other parameters the defines at the top of the “.cpp” file can be edited:



When CPU is set to true, the game will run on the system’s CPU. However, no image will be displayed as the CPU is not outputting an OpenGL texture. The terminal will show the time needed to calculate the next iteration in the game, and the ENTER key can be used to render the next frame. When the game is rendered on the GPU the terminal window will behave similarly, but additionally the GLUT window will show the state of the grid with alive (white) and dead (black) pixels. When the resolution is set very high, such as in the following tests, the pixels might be too small to see.

## Comparison

Several data points were gathered using the implementations to measure relative performance between settings and devices. Settings that were tested consisted of the CPU (Intel i7 3630QM) and four different workgroup sizes on the GPU (AMD FirePro M4000). Additionally, frames were rendered in five different resolutions. The data, measured in milliseconds per frame, is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Setting\Resolution | 1024x1024 | 2048x2048 | 3072x3072 | 4096x4096 | 8192x8192 |
| CPU | 15,8 | 47,7 | 107,5 | 192 | 770 |
| GPU 1x1 | 22,4 | 85 | 188,6 | 334,7 | 0 |
| GPU 4x4 | 2,6 | 6,8 | 14,3 | 24,6 | 115 |
| GPU 8x8 | 1,8 | 4,1 | 8,3 | 13,2 | 90 |
| GPU 16x16 | 3,1 | 3,5 | 6,4 | 10,7 | 90 |

This data can also be plotted in a graph to better show the differences:

As can be seen in the graph, the GPU can be significantly faster at calculating new frames than the CPU, when the right workgroup size is chosen. When the workgroup is set to 1x1, the GPU bogs down and is beaten by the CPU. When using this workgroup size, the GPU was also unable to finish one frame at the highest resolution in a reasonable amount of time. When the correct work group size is used however, the GPU outperforms the CPU by at least one number of magnitude, showing the “Game of Life” algorithm greatly benefits from parallelization.

Because this implementation of the game does not produce divergent threads or non-coalescent memory access, these things cannot be optimized further in our eyes. Because all cores take an equal amount of time to calculate a frame, because the same number of steps are taken for every possible situation, memory access will be coalescent and no divergence will occur.

We did however want to test the performance impact of copying the resulting grind from GPU memory back to system memory. We did this test at 3072x3072 with a workgroup size of 16x16, the maximum our GPU would allow for. The following graph shows the performance impact is sizeable:

This shows the use of OpenGL to display the resulting frames to the user is greatly benefitting performance, as otherwise the resulting frame would need to be copied back to system memory before being displayed. With different resolutions, the performance delta would differ, but it will never be better than displaying directly from GPU memory.

## Grading

As the assignment demands it, we will mention the grade we think we deserve for our execution of assignment 5. We think we have earned an 8/10 because we went beyond the regular assignment to implement a working “Game of Life” for the GPU and CPU. We have been successful in doing so.

Additionally, we have shown to understand different ways of optimizing an OpenCL application, even though we later found out some of these were not applicable to our “Game of Life” because it did not display signs of divergent threads or non-coalescent memory access. We did however document our performance testing well and will be able to explain our implementation to you very well.