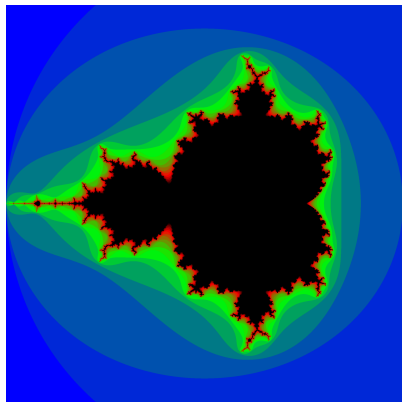


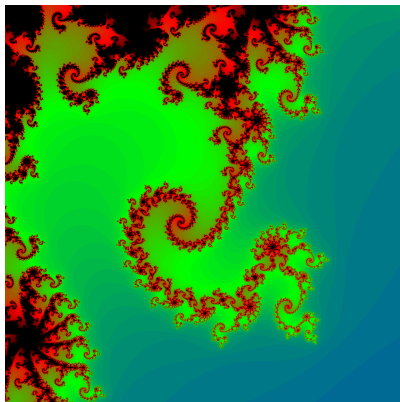
OpenCL exercise 2: Mandelbrot set

Kaicong Sun

Mandelbrot set



$x \in [-2, 1], y \in [-1.5, 1.5], n = 20$



$x \in [-0.813, -0.791],$
 $y \in [-0.188, -0.166], n = 110$

Mandelbrot set

- ▶ Complex sequence: $z_0 = 0, z_{n+1} = z_n^2 + c$ ($c \in \mathbb{C}, z \in \mathbb{C}, n \in \mathbb{N}$)
- ▶ Mandelbrot set: set of complex numbers c constrained that the sequence z_n remains bounded.
- ▶ Once $|z_n| > 2$ the sequence diverges
- ▶ In images: Complex plane for c values is displayed
 - ▶ in Mandelbrot set: black
 - ▶ not in Mandelbrot set: color shows iteration where $|z_n|$ exceeds 2

Host code

```
1  for (size_t i = 0; i < countX; i = i + 1) { //loop in the x-dir.
2      float xc = xmin + (xmax - xmin) / (countX - 1) * i; //xc=real(c)
3      for (size_t j = 0; j < countY; j = j + 1) { //loop in the y-dir.
4          float yc = ymin + (ymax - ymin) / (countY - 1) * j; //yc=imag(c)
5          float x = 0.0; //x=real(z_k)
6          float y = 0.0; //y=imag(z_k)
7          for (size_t k = 0; k < niter; k = k + 1) { //iteration loop
8              float tempx = x * x - y * y + xc; //real of z_{n+1}=(z_n)^2+c
9              float tempy = 2 * x * y + yc; //imaginary part of z_{n+1}
10             x = tempx;
11             y = tempy;
12             float r2 = x * x + y * y; //r2=|z_{n+1}|^2
13             if ((r2 > 4) || k == niter - 1) { //divergence condition
14                 h_output[i + j * countX] = k;
15                 break;
16             }
17         }
18     }
19 }
```

Task 1

- ▶ Implement mandelbrot set on GPU
- ▶ Use 2D index space
 - ▶ one work item for each complex value in the complex plane
- ▶ Check which loops in the CPU implementation can be parallelized
 - ▶ all work items run the same kernel code
- ▶ Time for CPU, GPU, memory transfer, speedups

- ▶ Check results

Task 2

- ▶ Use the second parameter set
- ▶ Explain the results
- ▶ Explain the effect on the speedup

Task 3

- ▶ Modify the number of work items per work group and the number of work groups.
- ▶ What happens?

OpenCL Types

OpenCL C	C++	Info
char	cl_char	signed 8-bit integer
uchar	cl_uchar	unsigned 8-bit integer
short	cl_short	signed 16-bit integer
ushort	cl_ushort	unsigned 16-bit integer
int	cl_int	signed 32-bit integer
uint	cl_uint	unsigned 32-bit integer
long	cl_long	signed 64-bit integer
ulong	cl_ulong	unsigned 64-bit integer
float	cl_float	32-bit float
double	cl_double	64-bit float
bool	-	boolean value (true or false)
size_t	-	pointer-sized unsigned integer
__global T*	cl::Buffer	pointer to data in global memory

OpenCL Types

- ▶ When passing a value to the kernel using `kernel.setArg()`, always specify the (C++) type of the parameter:
 - ▶ `__kernel void fooKernel(__global int* p, int i, float f);`
 - ▶ `fooKernel.setArg<cl::Buffer>(0, d_output);`
 - ▶ `fooKernel.setArg<cl_int>(1, intValue);`
 - ▶ `fooKernel.setArg<cl_float>(2, floatValue);`
- ▶ Pointers to global memory in the kernel (`__global T*`) are `cl::Buffer` in C++
 - ▶ The pointed-to type which matches `__global int* i` in the kernel should e.g. be allocated as `std::vector<cl_int>` in C++
- ▶ `cl_float = float, cl_double = double`

Syntax: Launching a 2D kernel

Syntax:

```
cl::CommandQueue::enqueueNDRangeKernel(Kernel kernel,  
    NDRange offset, NDRange global, NDRange local,  
    eventsToWaitFor = NULL, cl::Event* event=NULL) const;
```

Example for 2D NDRange:

```
queue.enqueueNDRangeKernel(kernel, cl::NullRange,  
    cl::NDRange(globalX, globalY),  
    cl::NDRange(localX, localY), NULL, &event);
```

globalX / globalY = Overall number of work items in X/Y-direction

localX / localY = Number of work items per work group in
X/Y-direction

Syntax: Kernel code

Get global index of the current work item in the x-direction:

```
size_t i = get_global_id(0);
```

Get global index of the current work item in the y-direction:

```
size_t j = get_global_id(1);
```

Get global size in the x-direction:

```
size_t countX = get_global_size(0);
```

Get global size in the y-direction:

```
size_t countY = get_global_size(1);
```