Mandelbulbulator

Mateusz Gniewkowski, Beata Szeląg

Agenda

- OpenCL
  - Architektura
  - Przykład programu
- Mandelbrot
- Mandelbulb
- Rezultaty

OpenCL

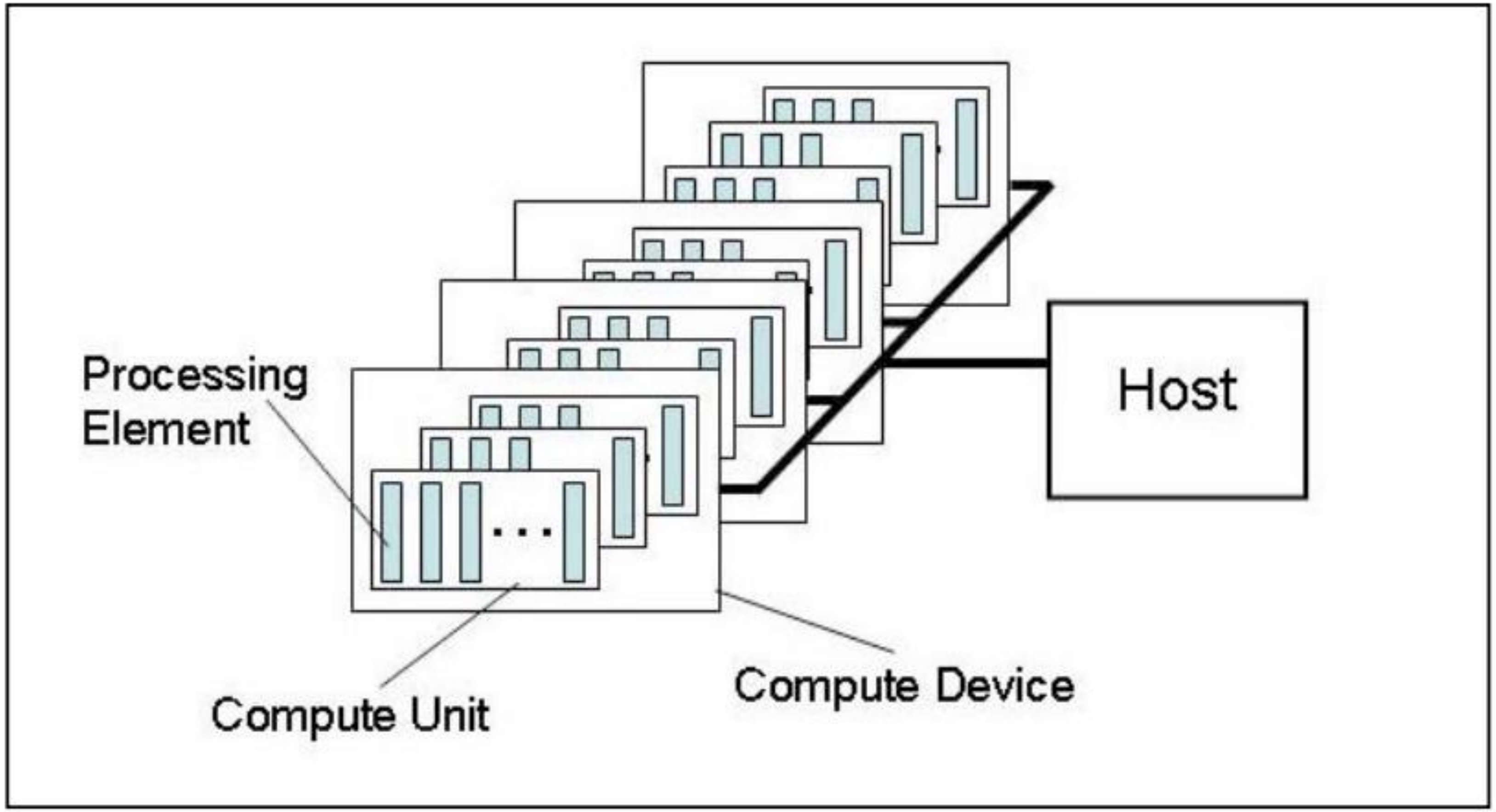


Figure 3.1: *Platform model … one host plus one or more compute devices each with one or more compute units each with one or more processing elements.*

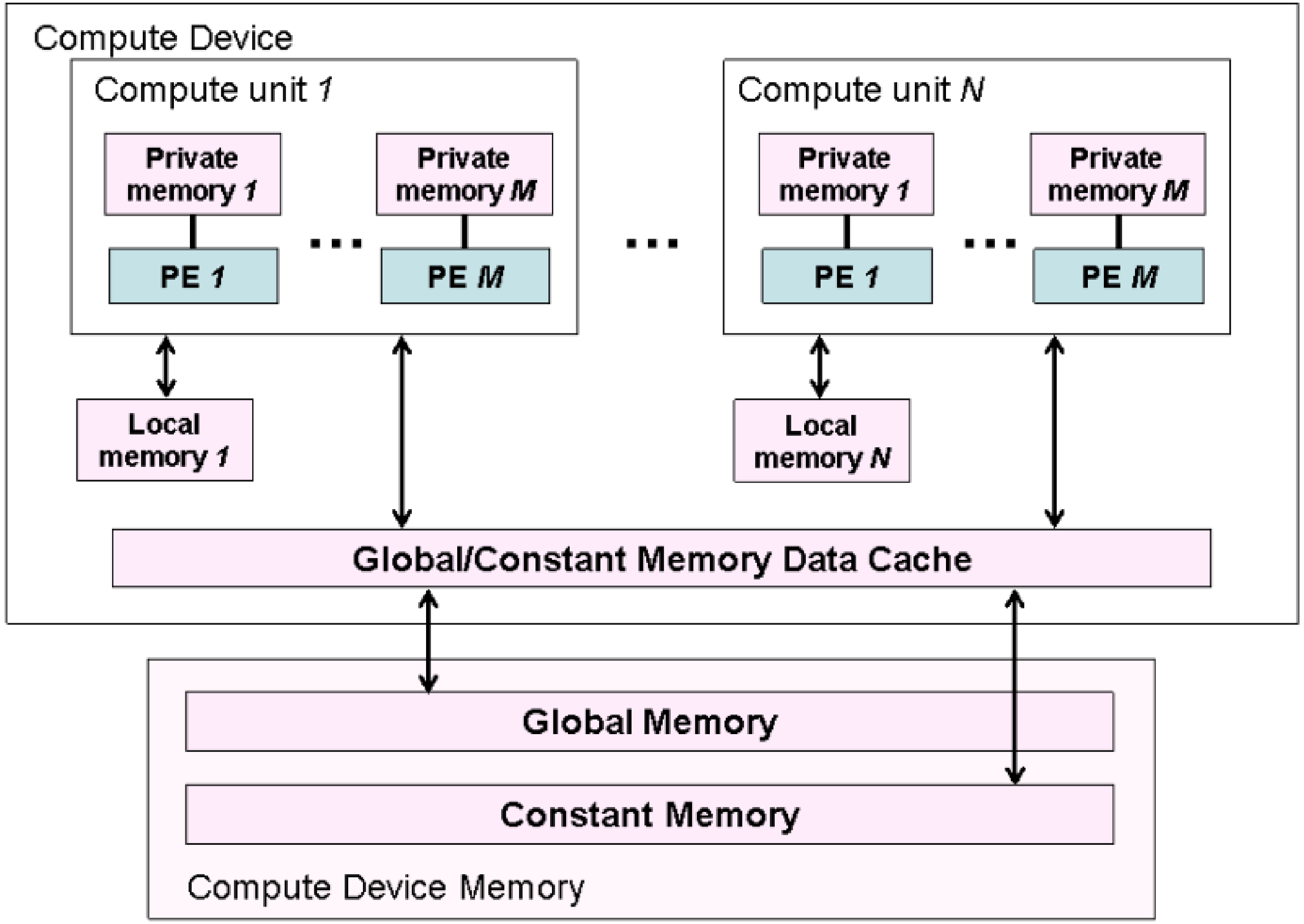
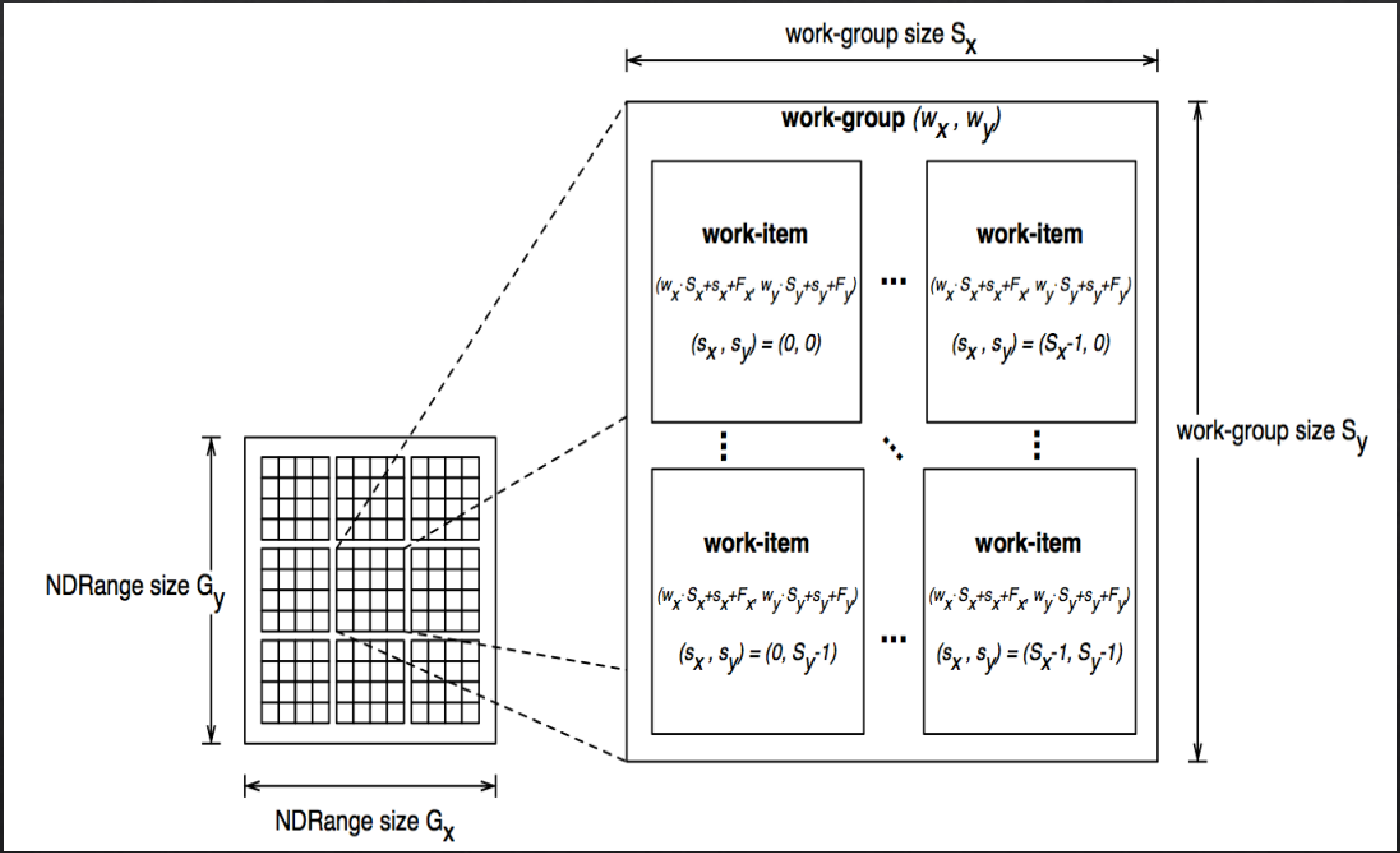


Figure 3.3: *Conceptual OpenCL device architecture with processing elements (PE), compute units and devices. The host is not shown.*


work-group size S_x
work-group (w_x, w_y)
work-item
$(w_x \cdot S_x+s_x+F_x, w_y \cdot S_y+s_y+F_y)$
$(s_x, s_y) = (0, 0)$
work-item
$(w_x \cdot S_x+s_x+F_x, w_y \cdot S_y+s_y+F_y)$
$(s_x, s_y) = (S_x-1, 0)$
work-item
$(w_x \cdot S_x+s_x+F_x, w_y \cdot S_y+s_y+F_y)$
$(s_x, s_y) = (0, S_y-1)$
work-item
$(w_x \cdot S_x+s_x+F_x, w_y \cdot S_y+s_y+F_y)$
$(s_x, s_y) = (S_x-1, S_y-1)$
work-group size S_y
NDRange size G_y
NDRange size G_x

Przykład aplikacji

OpenCL kernel

```
kernelsource = """
__kernel void vadd(
    __global float* a,
    __global float* b,
    __global float* c,
    const unsigned int count)
{
    int i = get_global_id(0);
    if (i < count)
        c[i] = a[i] + b[i];
}
```

https://github.com/HandsOnOpenCL/

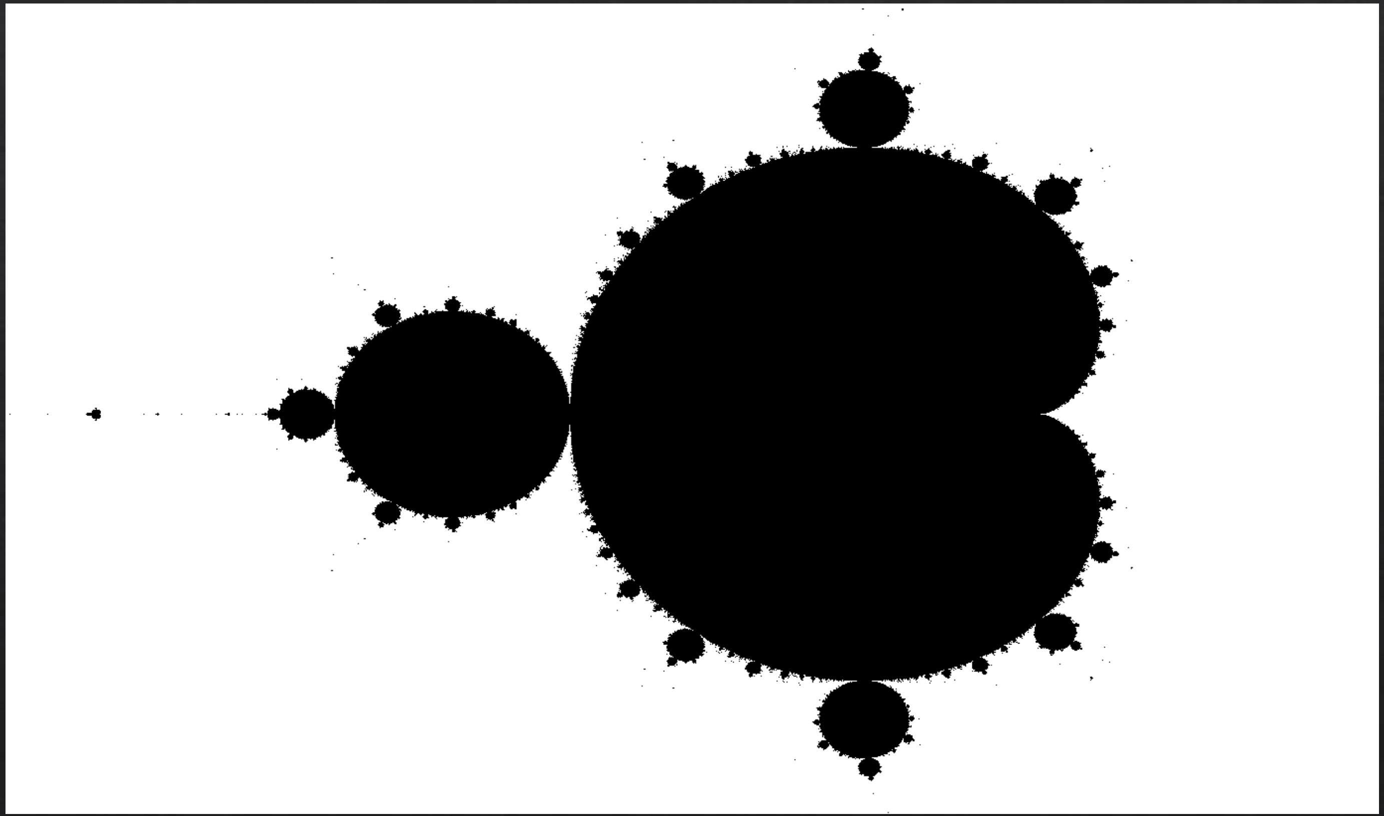
Host program

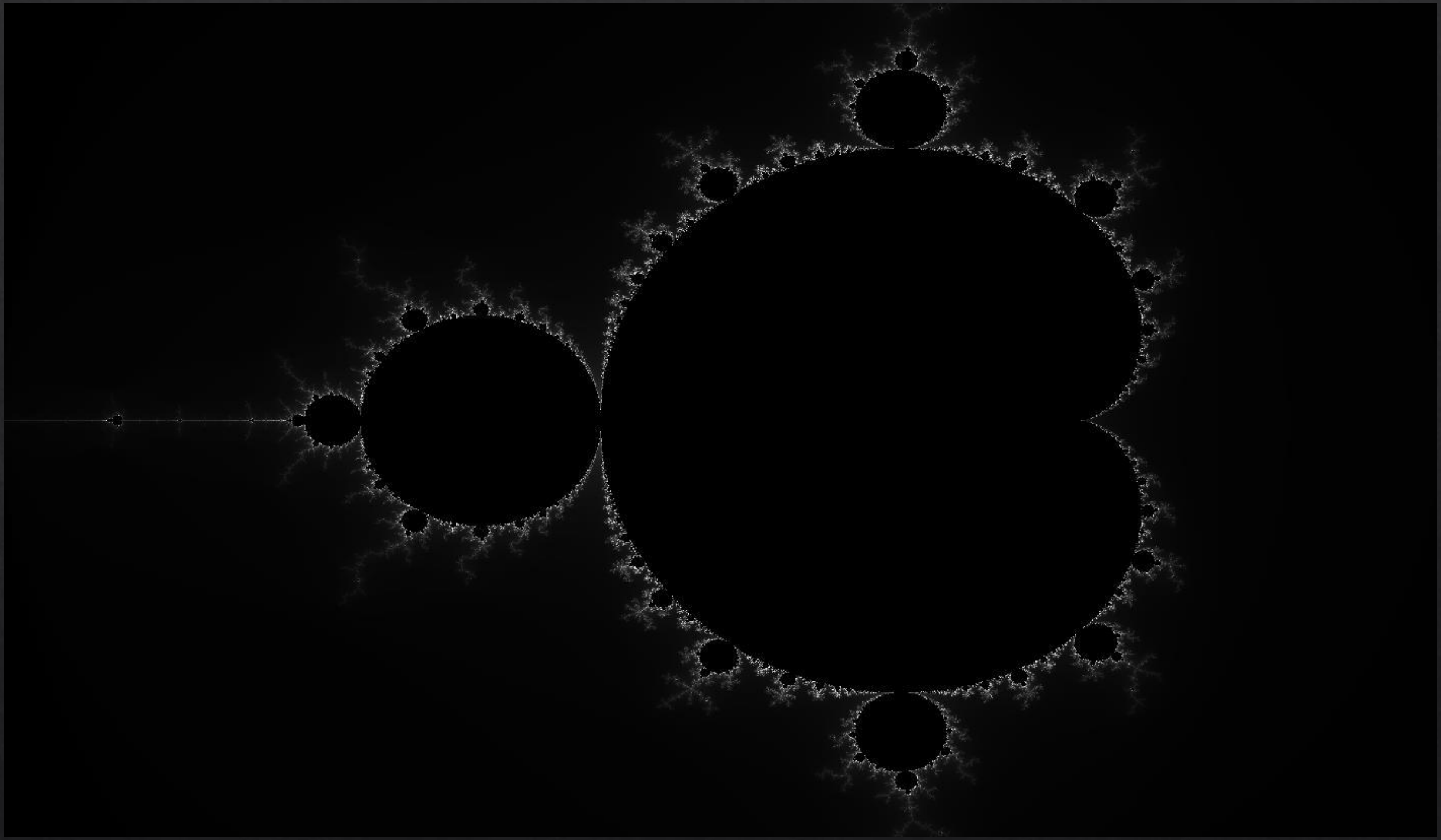
```
# Context
context = cl.create_some_context()

# Create a command queue
queue = cl.CommandQueue(context)

# Create the compute program from the source buffer and build it
program = cl.Program(context, kernelsource).build()

h_a = numpy.random.rand(LENGTH).astype(numpy.float32)
h_b = numpy.random.rand(LENGTH).astype(numpy.float32)
h_c = numpy.empty(LENGTH).astype(numpy.float32)

d_a = cl.Buffer(context, cl.mem_flags.READ_ONLY |
cl.mem_flags.COPY_HOST_PTR, hostbuf=h_a)
d_b = cl.Buffer(context, cl.mem_flags.READ_ONLY |
cl.mem_flags.COPY_HOST_PTR, hostbuf=h_b)
d_c = cl.Buffer(context, cl.mem_flags.WRITE_ONLY, h_c.nbytes)

vadd = program.vadd
vadd(queue, h_a.shape, None, d_a, d_b, d_c, LENGTH)

queue.finish()

# Read back the results from the compute device
cl.enqueue_copy(queue, h_c, d_c)
```

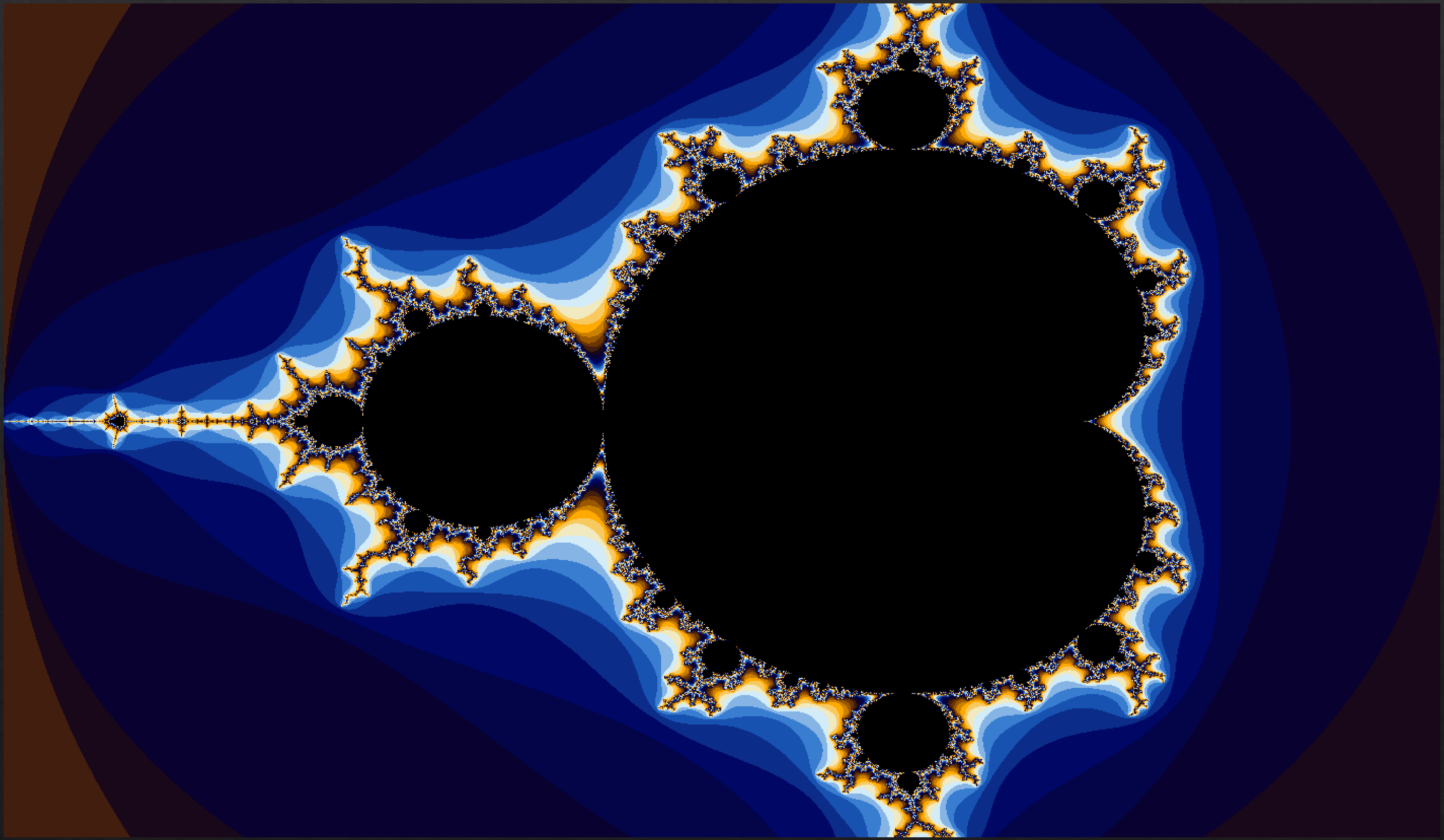
Mandelbrot

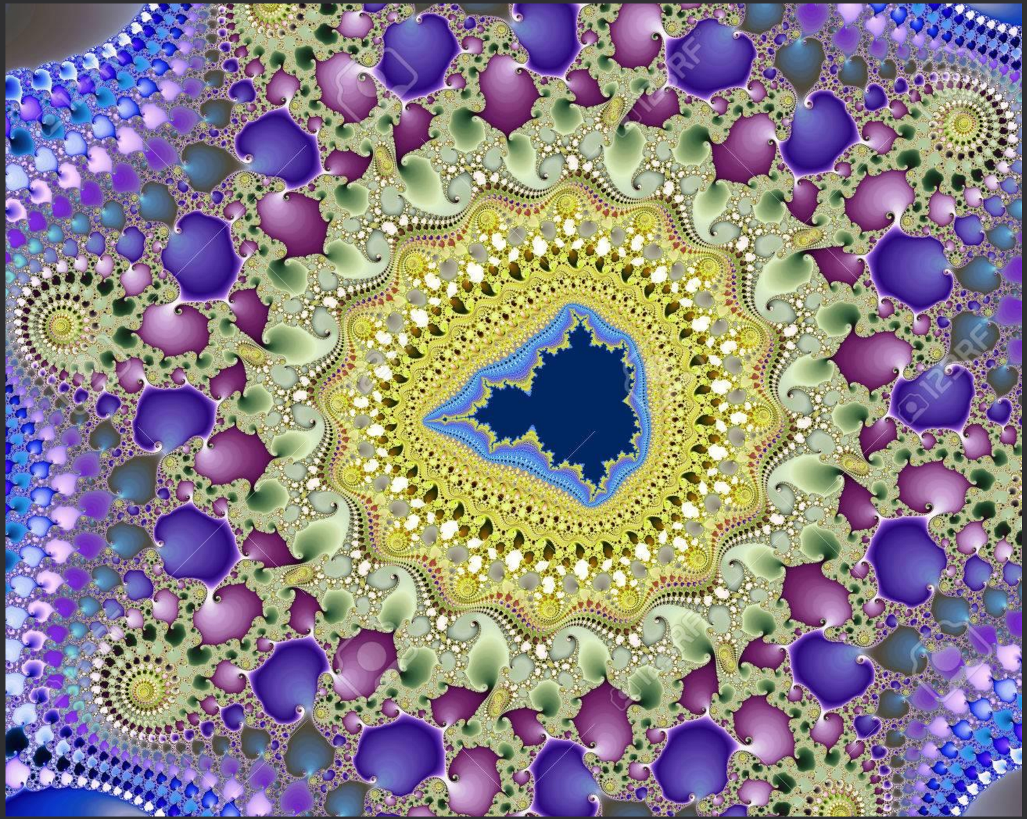
$$\begin{cases} z_0 = 0, \\ z_{n+1} = z_n^2 + p. \end{cases}$$

$$M = \{p \in \mathbb{C} : \forall_{n \in \mathbb{N}} |z_n| < 2\}.$$









https://www.123rf.com; by visharo

https://jonathanfrech.wordpress.com; by JFRECH

Mandelbulb

Daniel White and Paul Nylander in 2009

White and Nylander's formula for the "nth power" of the vector $\mathbf{v} = \langle x, y, z \rangle$ in $\mathbb{R}^3$ is

$$\mathbf{v}^n := r^n \langle \sin(n\theta)\cos(n\phi), \sin(n\theta)\sin(n\phi), \cos(n\theta) \rangle$$

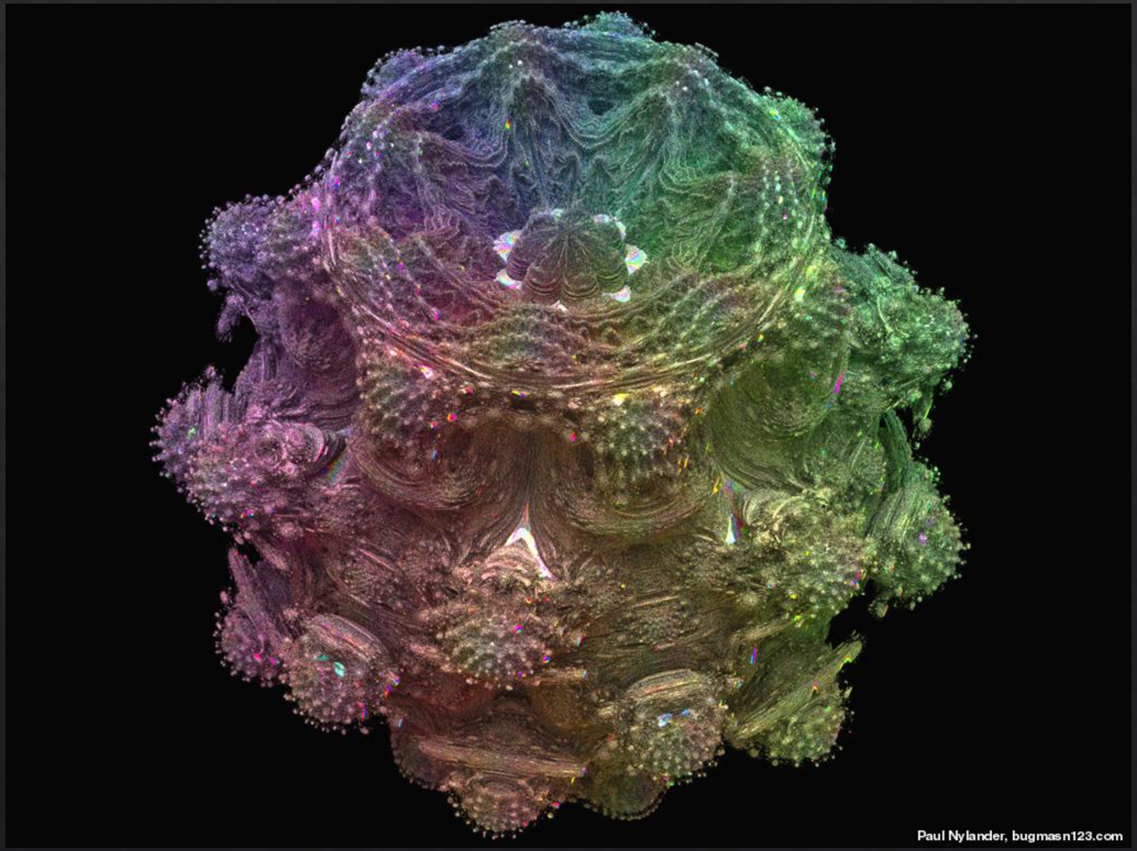
where

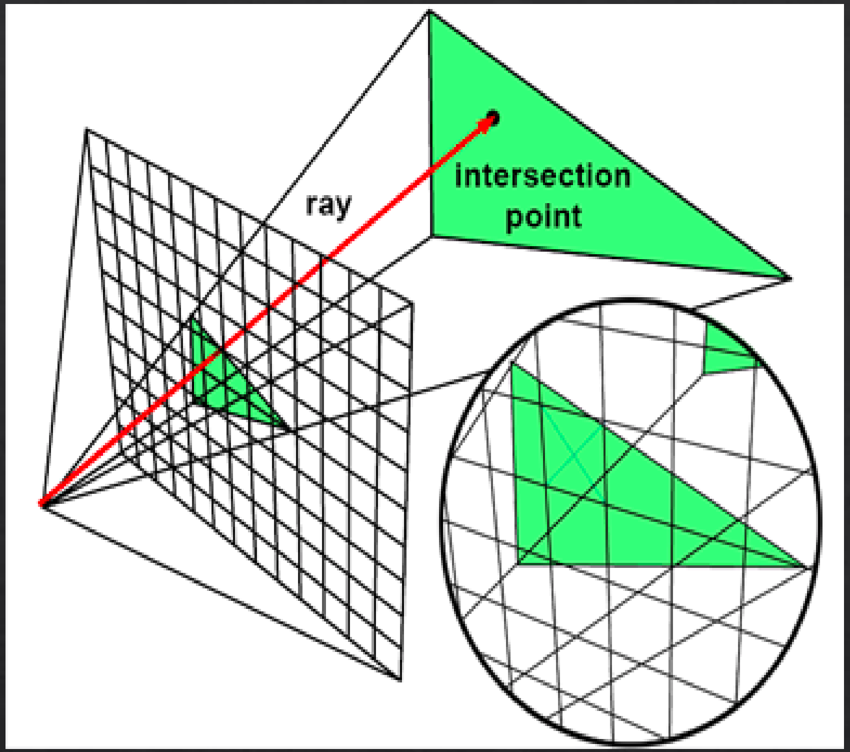
$r = \sqrt{x^2 + y^2 + z^2}$,

$\phi = \arctan(y/x) = \arg(x + yi)$, and

$\theta = \arctan(\sqrt{x^2 + y^2}/z) = \arccos(z/r)$.

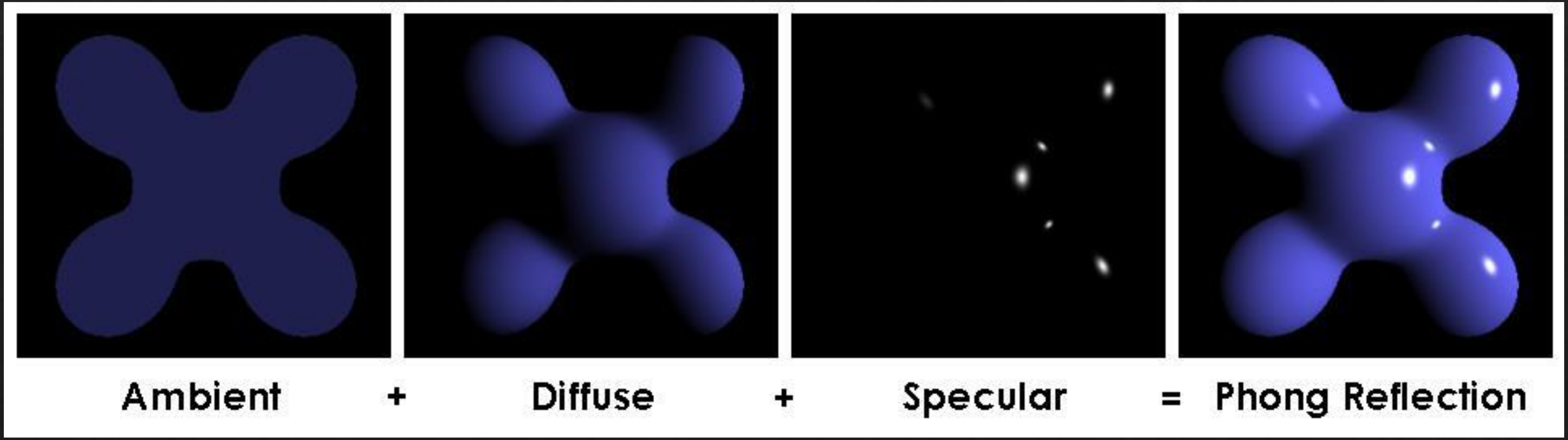
$$\mathbf{v} \mapsto \mathbf{v}^n + \mathbf{c}$$

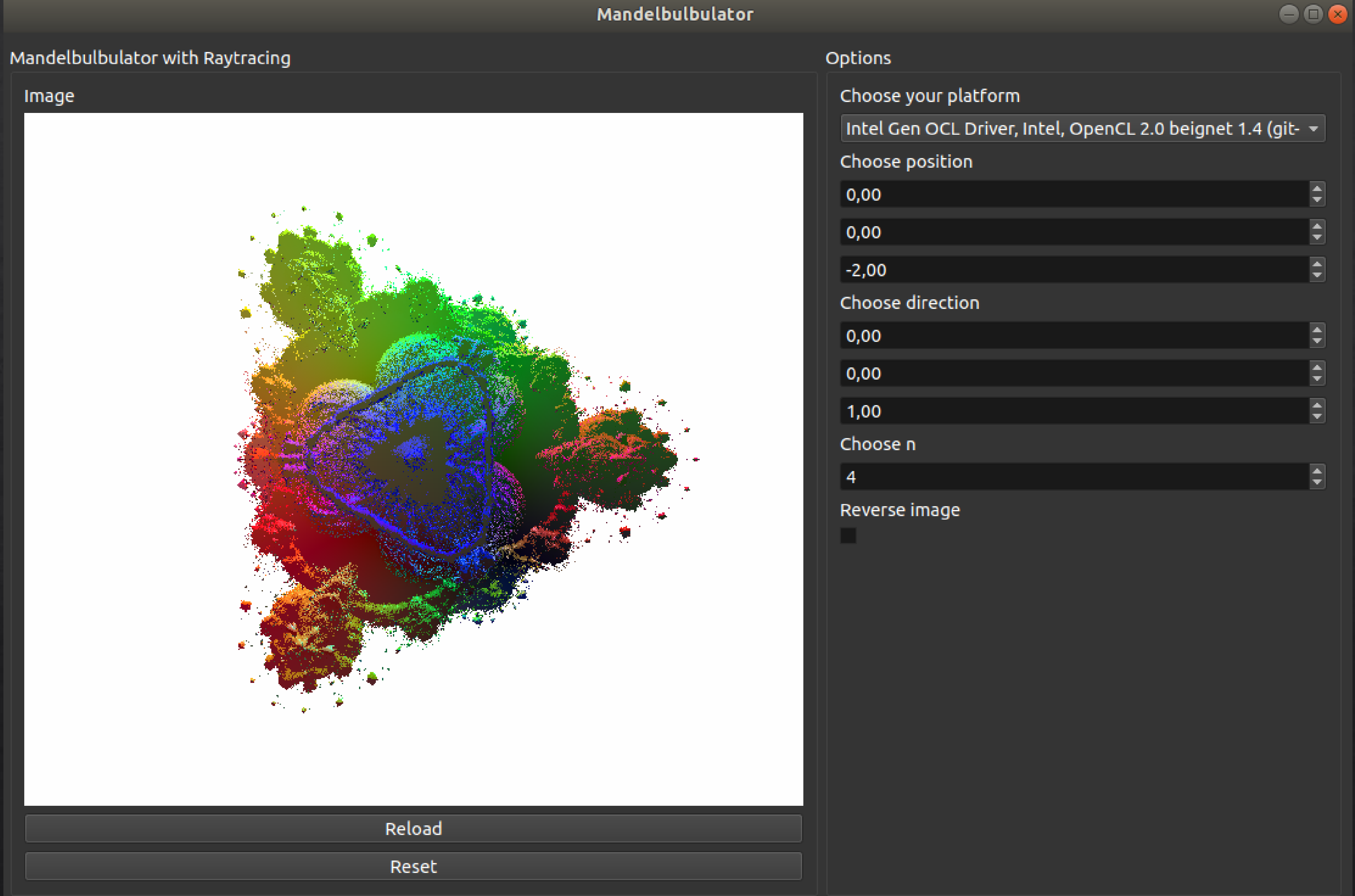


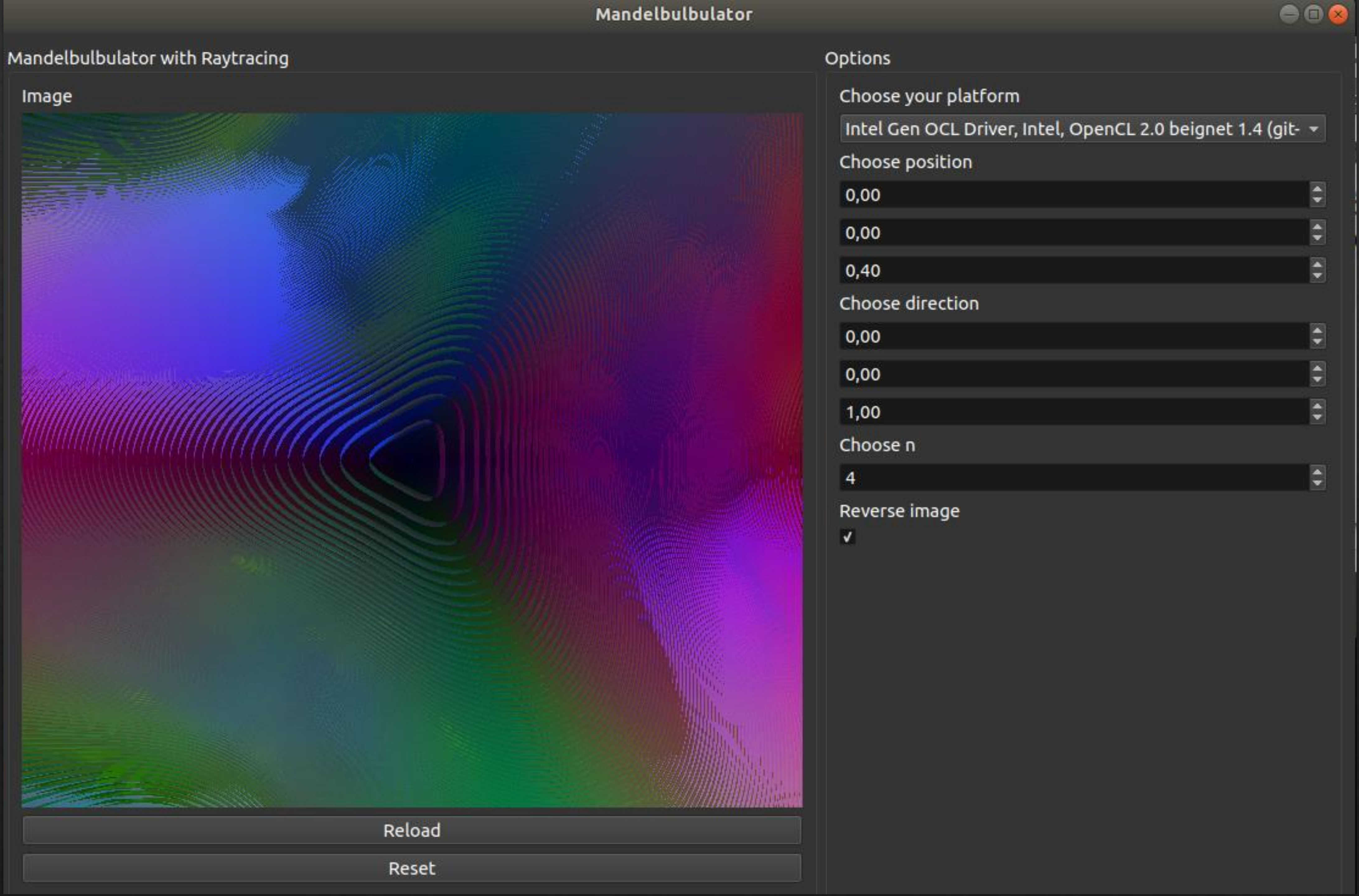


Znalezienie punktu przecięcia ze zbiorem polega na sprawdzaniu co pewien odcinek kolejnych punktów leżących na prostej zgodnej z wystrzelonym promieniem (wektorem).

Znalezienie wektora normalnego do powierzchni w danym punkcie wymaga próbkowania przestrzeni leżącej wokół niego.




Mandelbulbulator
Mandelbulbulator with Raytracing
Image
Reload
Reset
Options
Choose your platform
Intel Gen OCL Driver, Intel, OpenCL 2.0 beignet 1.4 (git-
Choose position
0,00
0,00
-2,00
Choose direction
0,00
0,00
1,00
Choose n
4
Reverse image


Mandelbulbulator
Mandelbulbulator with Raytracing
Image
Reload
Reset
Options
Choose your platform
Intel Gen OCL Driver, Intel, OpenCL 2.0 beignet 1.4 (git-
Choose position
0,00
0,00
0,40
Choose direction
0,00
0,00
1,00
Choose n
4
Reverse image
✓

Bibliografia

- https://www.khronos.org/registry/OpenCL/specs/opencl-1.2.pdf
- https://en.wikipedia.org/wiki/Mandelbulb
- https://github.com/HandsOnOpenCL
- https://cnugteren.github.io/tutorial/pages/page1.html
- https://www.scratchapixel.com/