Arbeitsjournal

Mikail Gedik

November 13, 2020

1 2020-05-06

Work on the concept has begun

2 2020-05-07

Work continues
Known Issues: Table of contents shows wrong page numbers, title page has number

3 2020-05-24

Sent first version the supervisor

4 2020-05-29

Work approved, started to add more paragraphs and content

5 2020-06-06

Added minimal modules to code and added time output. Interestingly, creating the image takes longer than calculating the fractal

6 2020-07-12

Added java modules for back and frontend. Simple multithreading. Start of summer break

7 2020-08-17

Added changed way data is stored in backend. End of summer break

8 2020-05-31 to 2020-07-05

Focus was on basic implementation.

The current goal is to render a simple Mandelbrot set into an image file. The next step was the modularization and mutlithreading, which were both achieved around (2020-07-12). Next up was a basic interactive windowed frontend, as before the only way to interact with the program was the command line. Using the java swing API, I created a simple window, where the image was rendered. It could be moved around with the mouse and zoomed in. Although the image was resized on zooming in, it was not recalculated, resulting in looking very coarse. However, solving this problem required a new underlying way the calculated data was stored. Motivated by this I started to develop different algorithms and data structures to store the data, each with different approaches to a solution, as well as different trade offs (2020-08-07).
Due to these changes I have been able to buffer the image creation in such a way that the whole image does not have to be recalculated and can be based on the previous in case of zooming in or moving around.
Lastly, the color function was changed to display the number of iterations a value of the set had to do to be excluded of the Mandelbrot set.

9 2020-09-29

The application can now zoom in into the image infinitely. However, even though no new values are created, RAM is reserved for not yet calculated values. Because the RAM usage rises exponentially, all available RAM is quickly consumed, leading to a systemwide slowdown of all open applications, as SWAP will be used. When a new image with a higher quality is requested, only the new values of the required layer are calculated.

10 2020-10-05

A small section was added to the mandelbrot set

11 2020-10-13

The project was largely refractored in reimplemented. It can now zoom in without major lag or RAM hugging. Especially reworked was data storage (uses clusters and levels now). Problems or bugs:

1. RAM hugging when the buffer of generated images becomes too big, especially if resolution is high
2. Application crashes when zooming out too much
3. Cannot zoom infinitely for not yet identified reason

TODO/Addition functions

1. Document work in Thesis paper

2. Export video
3. OpenCL implementation
4. Save data to disk
5. Intelligent computation (do not calculate points which are clearly in the fractal and require long)
6. Connect multiple PCs over Ethernet
7. GUI rework

   Maybe with GLFW

12 2020-10-16

Added image generation documentation in thesis paper

13 2020-10-25

Added OpenCL support TODO

1. GPU work is splitted evenly on GPUs even if one is massively faster
2. OpenGL support to make image manipulation
3. Add GLFW, Native file dialogs (and maybe OpenAL?)
4. OpenCL memory management