EPITA

Book of Specifications

Gaporu

Gaurav Lokwani
Alexandre Poirier-Coutansais
Martin Ruhlmann

As part of the S4 semester at EPITA, we were asked to do a free project, meaning we choose our subject. The project has to be using the C language and run on EPITA's computers. While the project idea we had is not likely to run well on EPITA's computers, it certainly can. Our topic is therefore described in this document along with the schedule we set ourselves for the delivery of said program.

Printed the: February 12, 2020

1 Overview

This project aspires to create a rendering program for 3D fractals using ray-marching. Fractals have always been a fascination for mathematicians and artists, their beauty and complexity emerges from a usually simple formula or pattern. This program would allow for a deeper view into these shapes.

Ray-marching is a rendering technique similar to ray-tracing but optimised for simple shapes. As a fractal is just a repetition of simple shapes, ray-marching allows for a huge optimisation in rendering fractals.

Even though ray-marching allows us to considerably reduce the computing time of our program, the amount of floating-points calculations that has to be done in order to render an image is still huge. A typical CPU will probably take a long time to render a single image, let alone a video. Hence why we decided to utilise the strong floating point calculation efficiency of the GPUs (Graphical Processing Units) along with their capabilities for parallelism. Ideally, we would like for the program to run in real-time, but as this is a big task and although we aim for it, we won't expect it to.

For the user to interact with our program, we would like to have a GUI that would allow for camera movement and settings for the fractal to view. If the fractals cannot be rendered in real-time, we could implement a path saving for the camera with a dummy object as the fractal, and then let the program render a video.

2 Technologies

Several technologies will be required to accomplish this project, from the operating system to the library that will allow us to compute on GPUs to accelerate the rendering.

Linux (Ubuntu, Manjaro, Arch...) This will be the main Operating System we will be developing our program for. The program will work on a large selection of distributions depending on the implementation of OpenCL and Gtk on those.

Gcc (for the C language) The compiler for our C program.

OpenCL This is the library that will allow use to access the GPU and communicate with it. This library also works for Intel CPUs, while not as effective as a GPU, it will allow us to make the program run on a computer not equipped with a GPU.

Gtk Since we plan to show our results through a GUI, Gtk will be a requirement as well as Glade to conceptualise the GUI.

3 Tasks

3.1 Schedule

Tasks	1rst Defence	2nd Defence	3rd Defence
Ray Marching	40%	80%	100%
Still Image Saving	100%	100%	100%
Website	80 %	80%	100%
Interface	0%	60%	100%
Lighting	10%	80%	100%
Video Saving	0%	90%	100%
Variable Setting	0%	30%	100%
Camera Movement	0%	0%	100%
Integration	0%	0%	100%
Instructions	0%	0%	100%

3.2 Assignment

Tasks	Alexandre	Gaurav	Martin
Ray Marching	*	*	
Still Image Saving		*	*
Website		*	*
Interface	*		*
Lighting	*	*	
Video Saving	*		*
Variable Setting	*	*	
Camera Movement		*	*
Integration	*		*
Instructions	*	*	

3.3 Bonus

- Colouring
- Ambient Occlusion
- Real-Time Rendering
- Real-Time Variable Modification

4 Motive

GPU computing is a really important skill for many applications, be it game design, computer simulations, micro-trading, animation, deep-learning, image treatment, and any kind of algorithm that can benefit from parallel computing. We hope that this project will bring us experience on GPU development in the C language. Since OpenCL is widely spread and can be executed on both AMD and NVIDIA's products (both leaders in the GPU market), it is an important tool to learn to use. While it is not as performant as NVIDIA's CUDA, it is more versatile regarding compatibility.

We also hope that this project helps to inspire new student developers and artist to the wonders of fractals, as they are an eye catching mathematical structure. This project can also be further developed as a interactive music animation, syncing music to the fractal's evolution. An other similar project made a game from a fractal rendering program with collision detection on dynamic fractals.