EPITA

First Report

Gaporu

Gaurav Lokwani
Alexandre Poirier-Coutansais
Martin Ruhlmann

This report is for our first defence, this includes our aim for the project, the work done for first presentation, our experiences, and our motivation for the project.

Printed the: February 12, 2020

Contents

1 Introduction

The word Gaporu comes from the initials of the name of the members of our group. 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 Team

2.1 Alexandre Poirier-Coutansais

I am a twenty year old student at EPITA, I have worked on four school projects throughout my stay at EPITA in the last two and a half year. I have some strong passions in algorithmics and software engineering in general, I chose the ISN option in my last year in Lycée and developed my first neural network there in C++. It wasn't my first experience in programming as I have been working on different small projects since I was 13-14 years old. I have a relatively strong knowledge in C, C++, C# and Python, I have also played with PHP, Javascript, Java Rust although I don't quite feel confident in these languages yet. I also play a lot with scripting languages such as Bash and Lua. I plan to do my career in algorithmic research or in software engineering, for the latter, hopefully as a System developer as network isn't really my type of thing. I have a big side project on which I am working on besides this project which is focused in simulating the gravitational lensing effect around black holes as accurately as possible. This school project will help me a lot towards this side project as some of the experience I was missing for this latter project will be present in this school project. I am really looking forward to this project and I am happy to be part of it.

2.2 Gaurav Lokwani

I got interest in Computer Science when I was in class 10th and when I completed my 12th grade I decided to join EPITA which is one of the well known colleges for Computer Science and now I further decide to continue in this field. I am particularly amused by Cyber Security and I decide to pursue my career in this area. I wanted to be a part of this project so that i can learn new things in this domain and i have not created any graphic related software before so it is going to be a whole new experience for me to get into the graphics designing. If i got interested in this area there are chances that i might keep following it and try my best to be a good graphic designer. The things I learned from this project are:
1 : Dedication - We fixed a particular day in every week to work on our project and we had our goals set so we can have a track record of our accomplishments from time to time.
2 : Teamwork - We always tried to synchronize with each other and i cannot even encounter moments when we had a fight because every time we were working , we used to help each other with the problems.
3 : Motivation - The aim for fixing a day in every week and working together was to motivate each other to work hard , to do their best , and to achieve excellent results.
4 : Appreciation - This is the art which everyone should know because it helps each individual to work harder and even give other members the desire to achieve the best.

2.3 Martin Ruhlmann

I am a twenty year old student at EPITA. After two projects during the two past semesters I am very happy to study on ray marching and 3D fractals. When Alexandre introduced me to ray-marching programs I thought it could be an interesting way to study on 3D. I think that 3D fractals are an interesting way to study mathematics and associate it with a visual representation in such a mesmerizing way.

3 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.
GNU Make	This utility program allows us to easily compile and manage large projects.

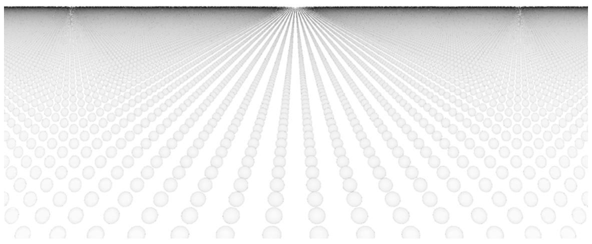
4 Progress

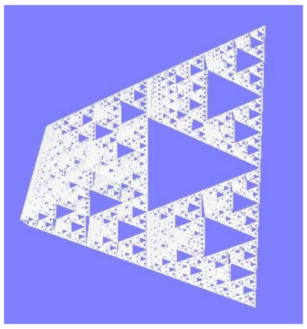
4.1 Ray Marching Engine

4.1.1 OpencCL Code

Renderer: The renderer is the core of our implementation of the ray marching algorithm. This is an OpenCL kernel that performs the ray marching for height * width pixels (height and width are two variables which can be modified my the user when the interface will be complete and define the resolution of the final view). This kernel takes in parameters a matrix (dimension of height * width) of 3D vectors which corresponds to the direction vector in which ray marching should be performed for each pixel. As a second parameter, it takes in the camera position in the scene and as a third parameter, the image buffer in which it will write the pixel data of the image. A few preprocessor defined variables are also affecting this kernel such as the maximum render distance from the nearest object, the collision threshold and the maximum number of steps to be taken by the ray marching algorithm (this is mostly a safety limit to avoid infinite loops when the Distance Estimator is not converging towards either of the previously defined bounds). These preprocessor defined arguments will be available in the future for the user to adjust according to his computing resources once the interface is implemented. The Distance Estimator (ED) used by the renderer is a function that estimates the distance of a point to the nearest surface of all objects in the scene (see below).
At the moment, the renderer kernel determines the color and lighting through hard-coded operations, however, the future goal is to have different functions for lighting and color which can be affected by user-defined arguments. The current lighting implementation is also very crude and assumes that the light source is at the camera's position which is not optimal as it removes a large part of the perspective that are shadows. However, this implementation does allow a rudimentary ambient occlusion which allows the distinction of shapes from others. This implementation of lighting is only here for demonstration purposes and will not be staying for the final implementation.

Distance Estimator: The Distance Estimator (ED) is, as stated above, a function that estimates the distance from a point to the nearest surface of all objects in the scene. It returns a scalar which the renderer kernel will use to determine the length of the next step for a given pixel. It only takes in as argument a 3D vector representing the point from which to calculate the distance. This function is usually relatively simple yet powerful. A sphere for example is easy to estimate the distance to, calculate the norm of point vector, subtract from it the radius of the sphere, and the result is the distance from the point to the nearest surface of the sphere at position (0;0;0) of the previously mentioned radius. If you pass each component of the point vector through a modulo n, and then apply to this new vector the previously mentioned expression, you have an infinite grid of spheres separated by n units in each coordinates. Thus we are able to render an infinite amount of spheres (still limited by the render distance) with minimal computing effort.
The downside to this technique is that the DE has to be rewritten for each scene and recompiled. Thankfully, OpenCL already compiles the OpenCL code at run-time, thus this will only result as loading times for the user. Since OpenCL compiles the kernel sources at run-time, this will allow us to create an interface in which the user could code his own Distance Estimators and thus explore fractals we don't provide Distance Estimators for. However, this is a task that falls under the 'Bonus' category of the project and will only be done last if we are ahead of schedule.
So far, we have already implemented 2 Distance Estimators, one for a 2 dimensional infinite grid of spheres as described above, and a recursive tetrahedron which is a simple fractal.


Infinite Grid of Spheres


Recursive Tetrahedron

Camera Projection: This OpenCL kernel computes a 2D matrix of 3D vectors that represent the direction the ray marching algorithm should take for each pixel. This is taken as argument by the renderer kernel and only needs to be recomputed when the camera changes position. For now, since we are rendering still images, this doesn't provide a performance boost for the engine, however, if we want to achieve real-time rendering, this will provide a significant optimization. This kernel does a lot of trigonometry operations and has proven to be hard to get right. The main problem comes from the fact that we are projecting a sphere onto a plane (our final image), and we need to determine the vectors for each pixel of the plane without getting distortions. A few naive implementations exists but provide fish-eyed views which are not the results we are looking for. The current implementation doesn't distort the image assuming either the pitch or the yaw angle of the camera is 0. This is a bug and will be corrected for the next defence. This has surprisingly turned out to be the most time consuming part of the project so far.

4.1.2 C Code

Environment Structure: This data structure allows us to store settings and parameters which are important for the program to function. A field is dedicated to a Camera structure which holds the camera position and it's pitch and yaw. Other fields contains the Field Of View settings and the resolution represented by it's height and width. This data structure also contains a few important OpenCL structures, such as the OpenCL context, the command_queue attached to the context and the platform and device ids used by our software (If the system running our software has multiple GPUs, we could harness the power of each one of these).

Renderer Handler: This function takes an environment as an input and returns a GdkPixbuf image in output. This is the orchestrator of the OpenCL environment and queues the different kernels for execution. For now, it is a crude implementation of what it will end up being in the future as, in it's current form, each call to the renderer will recalculate the camera angles, recompile the OpenCL code and allocate and free the memory buffers. The renderer calls other C functions we defined such as the program compiler.

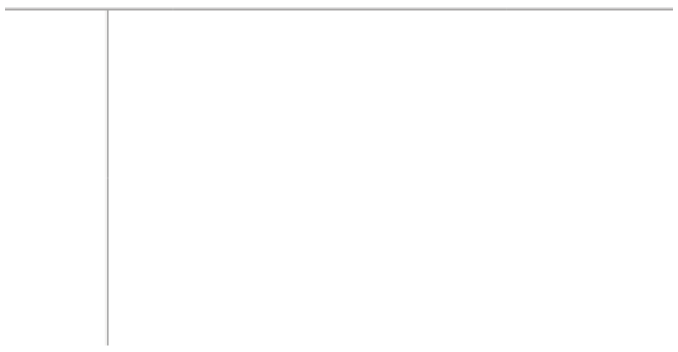
Program Compiler: This function handles the OpenCL calls to compile an OpenCL program. It takes in as argument a path to the source file in which the OpenCL code is contained. It reads the file, stores it into a buffer of size defined through the preprocessor as MAX_SRC_SIZE, this would be the maximum size of the OpenCL source code in bytes.

4.2 Makefiles & File structure

Our choice for the directory tree of the project and the Makefile disposition was important to do as early as possible as it defines how we will work and how the project will be structured. We have separated each component of the project, the main loop, the renderer and the interface, into different directories. In each of these directories the files are separated into: a 'src' directory which contains '*.h' files, '*.c' files and '*.cl' files; a 'obj' directory which holds '*.o' files and '*.d' files; and a unit-test directory which contains a test.c file and it's corresponding test.o and test.d files. The content of this last directory allows us to compile individual components of the project into a separate testing main loop to perform demos, unit tests and functional tests.
Each component has it's own Makefile with targets such as 'tests' and '<the component's name>'. The latter compiles the content of the 'src' directory into object files which are saved into the 'obj' directory. The 'tests' target calls the target we just described and also compiles the content of the 'tests' directory, then it links all the object files created along with the libraries used within the component, and outputs an executable file which can be executed to perform the tests specified in the 'tests/tests.c' file. In the project's main directory, another Makefile is responsible for calling the Makefiles of all components with their respective target '<the component's name>" and linking all the object files created into one executable. This allows this project to be really flexible and easy to test.

4.3 Website

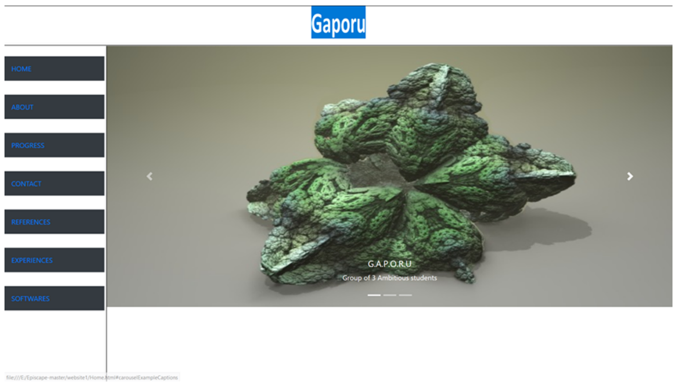
The Website was very innovative part as it is created purely by coding, when we searched for the ways of creating a website we found several already made layouts but the core motive of the project was to learn something new. Then we went on by learning how to make a website and it was very enriching experience as we started with very basic HTML, and while learning we came to know about bootstrap. It helped a lot as it uses JavaScript and it helped us to provide animations to my website.



In the Website we used frames as we found it to be easily customizable and it allows us to access several parts of a particular webpage without modifying others. It was really helpful as it provided us the opportunity to put the navigation buttons on the left and display their results on the Main Screen. In our website we have provided a progress page which describes the prediction and the amount of work actually done till date.

We also have an “About” page which describes the the naming and origin of our group and description of the members, in order for the consumers to give feedback we also have a “Contact” page with each Epita-Email-Address of the members so users can contact us for bug reporting when the software is released. In the website, we also feature a “References” Page which provides the download link to our Book of Specifications, Project Report from different Presentations and most importantly our “Software” .

About keeping the website up to date - we are constantly trying to upgrade our website and it would be updated every time we modify something. And it is even possible that if we continue exploring in this Domain then there are chances that this website would still be updated.



And we are trying our level best to have some more animations incorporated in this website. But as we explained before even if this website is not as beautiful as it should be, it is purely coded by us , we have not copied any layout or took help of any Website Designers(online) and by making this website. We also completed our main aim which was to learn something new which will help us in our future since we now have gained this new skill. It will further help us to put this knowledge in use wherever we want and it can be very useful for us.

5 Tasks

5.1 Schedule

Percentage Legend: Goal%(Achieved%)

Tasks	1rst Defence	2nd Defence	3rd Defence
Ray Marching	40%(70%)	80%(70%)	100%
Still Image Saving	100%(100%)	100%(100%)	100%
Website	80 %(80%)	80%(80%)	100%
Interface	0%(0%)	60%(0%)	100%
Lighting	10%(10%)	80%(10%)	100%
Video Saving	0%(0%)	90%(0%)	100%
Variable Setting	0%(0%)	30%(0%)	100%
Camera Movement	0%(20%)	0%(20%)	100%
Integration	0%(0%)	0%(0%)	100%
Instructions	0%(0%)	0%(0%)	100%

5.2 Assignment

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

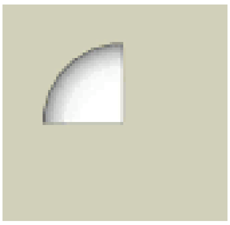
5.3 Bonus

- Colouring
- Ambient Occlusion
- Real-Ti me Rendering (If we can achieve real-time rendering, we won't do video saving)
- Real-Time Variable Modification
- User Defined Fractals

6 Problems Encountered

6.1 Camera Projection

The camera projection's math has been harder than expected, thankfully, distortion are obvious and we can easily be aware of an error in our math. The main struggle comes from determining a normalized vector for each pixel as if a rectangular based pyramid (with the top as the position of the camera and the base as the image) is intersecting with a sphere and mapping each pixel of the base to a point on the sphere. It is still not accurate when having both the camera pitch an yaw different from 0. This comes from the determination of the x component of each pixel direction vector. This will be fixed for the next presentation. A few problems also occurred in the beginning when figuring out how to properly apply the modulo operation to the coordinates without creating artifacts around sign changes. One of the result is shown below. This has been properly fixed since then.



What should be a sphere

6.2 Distance Estimators

We've had several problems in figuring out the math for the recursive tetrahedron's distance estimator, it is still not entirely accurate but we are confident we can solve this problem in the near future. The 2D sphere grid didn't cause a problem.

6.3 OpenCL

Since OpenCL is handled differently between Windows and Linux, we had and are still having trouble running it on Windows. Since some members on the team are mostly windows users, it was necessary to provide them with a solution. We settled on Dual-Booting and Live Linux sessions to help remedy this problem, however this will have to be properly sorted with a Windows compatible implementation in the future.

7 Motivation

7.1 Alexandre Poirier-Coutansais

I am really happy how well the project turned out for the first defense, we are ahead of schedule and this was a bit unexpected as I though this project would have been a larger endeavour than what it is. I am confident that we will be able to achieve our goals, and also impatient to see the final result. Learning OpenCL has proven to be an enriching experience, as I now understand much more about how GPUs work and still expect to learn much more. This project has been made easy to work on, partially thanks to the directory structure, compared to other school projects through the last few years, which has been a huge help in development.

7.2 Gaurav Lokwani

I am satisfied with the progress we did in the project as of now because we are comparatively ahead in our work and this time we did it very fast. I did the part of the website, but since this project is all about learning, instead of using a template or using sphinx to document it, I prefer to code it myself and that's how I learned to code in HTML/CSS.

8 Conclusion

We are really happy for our achievements for this first milestone, we have accomplished all the goals we set ourselves and we are even ahead of schedule for some parts of the project. We hope to continue to be ahead of schedule so that we can implement at least some of the bonus goals as some of these are really interesting and could provide a significant improvement to our project.