Constructing Sierpinski Gasket Using GPUs Arrays

Amira Sallow[1], Dhuha Abdullah[2]

[1] Computer and I.T. College, Nawroz University
Duhok, Iraq

[2] Computer Sciences and Mathematics College, Mosul University
Mosul, Iraq

Abstract

A fractal is a mathematical set that typically displays self-similar patterns, which means it is "the same from near as from far". Fractals may be exactly the same at every scale, they may be nearly the same at different scales. The concept of fractal extends beyond trivial self-similarity and includes the idea of a detailed pattern repeating itself. The algorithms to constructing different fractal shapes in many cases typically involve large amounts of floating point computation, to which modern GPUs are well suited. In this paper we will construct Sierpinski Gasket using GPUs arrays.

Keywords: *fractal, Sierpinski gasket, self-similar, GPU.*

1. Introduction

1.1 Fractal

The formal mathematical definition of fractal is defined by Benoit Mandelbrot [??]. It says that a fractal is a set for which the Hausdorff Besicovich dimension strictly exceeds the topological dimension. However, this is a very abstract definition. Generally, we can define a fractal as a rough or fragmented geometric shape that can be subdivided in parts, each of which is (at least approximately) a reduced-size copy of the whole. Fractals are generally self-similar and independent of scale **[5].**

1.2 Properties of Fractal

A fractal is a geometric figure or natural object that combines the following characteristics **[3]** :

a) Its parts have the same form or structure as the whole, except that they are at a different scale and may be slightly deformed;

b) Its form is extremely irregular or fragmented, and remains so, whatever the scale of examination;

c) It contains "distinct elements" whose scales are very varied and cover a large range;

d) Formation by iteration;

e) Fractional dimension.

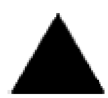
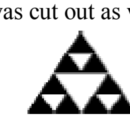
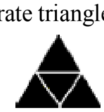
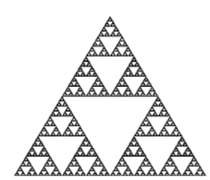
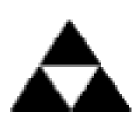
1.3 Sierpinski Triangle

The Sierpinski's Triangle is named after the Polish mathematician Waclaw Sierpinski who described some of its interesting properties in 1916. It is one of the simplest fractal shapes in existence It can be generated by infinitely repeating a procedure of connecting the midpoints of each side of the triangle to form four separate triangles, and cuting out the triangle in the center.

1.4 Constructing Algorithm

Construct the Sierpinski Triangle Taking a equilateral triangle as an Table (1) **[2]**:

Table 1: Constructing Sierpinski Triangle

1. Start with the equilateral triangle. 	4. Repeat the steps 1, 2 and 3 on the three black triangle left behind. The center triangle of each black triangle at the corner was cut out as well. 
. Connet the midpoints of each side of the triangle to form four separate triangles. 	5. Further repetition with adequate screen resolution will give the following pattern. 
3. Cut out the triangle in the center. 	

2. Graphics Processing Units

Graphics Processing Units (GPUs) used to be fixed function processors that helped the CPU with displaying images. This changed early 2001, when the first programmable GPU was launched. With the GPU being programmable, developers had much more opportunities to control the graphics operations themselves.

Programming a GPU though, is very different from programming a conventional CPU.There are programming languages designed for programming a GPU that are easy to learn,but taking full advantage of the computational power of the GPU, requires knowledge about the GPU. To fully exploit the power of the GPU, it is advisable to understand its architecture and its capabilities. Also, knowing how the data exactly passes through the GPU is required in order to build efficient programs, **[4]**.

3. GPU architecture

A graphics processing unit (GPU) is a specialized processor that offloads 3D or 2D graphics rendering from the microprocessor. It is used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics, and their highly parallel structure makes them more effective than general-purpose CPUs for a range of complex algorithms. In a personal computer, a GPU can be present on a video card, or it can be on the motherboard **[1]**.

The GPU is especially well-suited to address problems that can be expressed as data parallel computations the same program is executed on many data elements in parallel with high arithmetic intensity the ratio of arithmetic operations to memory operations. Because the same program is executed for each data element, there is a lower requirement for sophisticated flow control; and because it is executed on many data elements and has high arithmetic intensity, the memory access latency can be hidden with calculations instead of big data caches. Data-parallel processing maps data elements to parallel processing threads. Many applications that process large data sets can use a data-parallel programming model to speed up the computations. In 3D rendering large sets of pixels and vertices are mapped to parallel threads. Similarly, image and media processing applications such as post-processing of rendered images, video encoding and decoding, image scaling, stereo vision, and pattern recognition can map image blocks and pixels to parallel processing threads. In fact, many algorithms outside the field of image rendering and processing are accelerated by data-parallel processing, from general signal processing or physics simulation to computational finance or computational biology. Unlike modern CPUs, graphics chips are designed for parallel computations with lots of arithmetic operations. Much more transistors in GPUs work as they should they process data arrays instead of flow control of several sequential computing threads. Figure (1) shows how much room is occupied by various circuits in CPUs and GPUs:

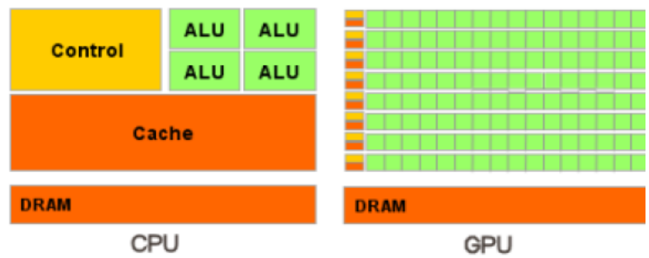


Figure 1: CPUs and GPUs

Therefore, lots of molecular modeling applications are adapted perfectly for GPU computing, they require high processing power, and they are convenient for parallel computing. And using several GPUs gives even more computing power for such tasks.

4. Conclusion

In this paper, Constructing Sierpinski Gasket Using GPUs Arrays where considered. We conclude that Graphics Processing Units are highly useful in parallel computing and are designed for parallel computations with lots of arithmetic operations. The objective of this project was to show that GPUs were more effective than CPUs.

References

[1] Berillo, Alexey , "NVIDIA CUDA." iXBT Labs ,2010. Available online athttp://ixbtlabs.com/articles3/video/cuda-1-p1.html

[2] Jessica Brazelton, “Matrix Multiplication using Graphics Processing Unit”, Tuskegee University, 2010.

[3] Matthias Book, “Parallel Fractal Image Generation A Study of Generating Sequential data With Parallel Algorithms”, The University of Montana, Missoula, USA. Available online at http://matthiasbook.de/papers/parallelfractals/

[4] Peter Geldof, “Generic Computing on a Graphics Processing Unit”, M.S. thesis, Universities van Amsterdam, Amsterdam, Netherlands, 2007.

[5] Werner Hodlmayr, “FRACTAL ANTENNAS: Introduction to fractal technology and presentation of a fractal antenna adaptable to any transmitting frequency”, antenneX, Issue No. 81, 2004.

[1] Amira Sallow is Lecturer in the Department of computer science in Computer and I.T. College at Nawroz University. She received her PhD degree in computer sciences in 2013 in the speciality of distributed real time system and operating system. She also leads and teaches modules at both BSc and MSc levels in computer science.

[2] Dhuha Albazaz is the head of Computer Sciences Department, College of Computers and Mathematics, University of Mosul. She received her PhD degree in computer sciences in 2004 in the speciality of computer architecture and operating system. She supervised many Master degree students in operating system, computer architecture, dataflow machines, mobile computing, realtime, and distributed databases. She has three PhD students in FPGA field, distributed real time systems, and Linux clustering. She also leads and teaches modules at BSc, MSc, and PhD levels in computer science. Also, she teaches many subjects for PhD and master students.