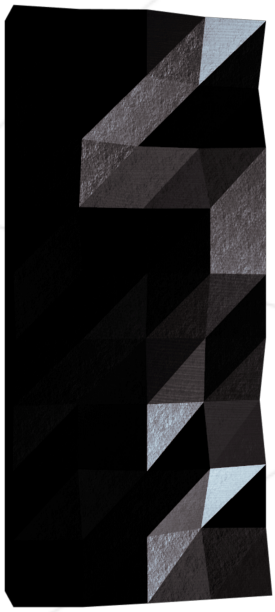


Particle System

Introduction to OpenCL

Rémi DAVID rdavid@student.42.fr
Paul VINCENT pvincent@student.42.fr

Abstract: This subject is an introduction to parallel computing on GPUs with OpenCL.

Contents

Chapter I

Preamble

Compute shaders operate di ff erently from other shader courses. All of the other shader courses-have a well-de fi ned set of input values, some built-in and user-Some fi ned. The frequency at qui shader has executed internship is speci fi ed by the kind of That internship; execute vertex shaders per ounce input vertex, for example (though Some executions can be skipped via caching). fragment shader execution is de fi ned by the fragments generated from the rasterization process.

Compute shaders work very diff erently. The "space" that has a compute shader Operates Largely it is abstract; it is up To Each compute shader to decide what the space means clustering. The number of compute shader executions is fi ned by the function used to compute the executed operation. Most importantly of all, compute shaders-have no user-fi ned inputs and outputs number at all. The built-in inputs only fi do Where in the "space" of execution has Particular compute shader is invoked.

Therefore, if a compute shader wants to Take Some gains have input, it is up to the shader Itself to fetch That data via access texture picture arbitrary load, shader storage blocks, or other forms of interface. Similarly, if a compute shader is to compute anything Actually, it must Explicitly write to an image or shader storage block.

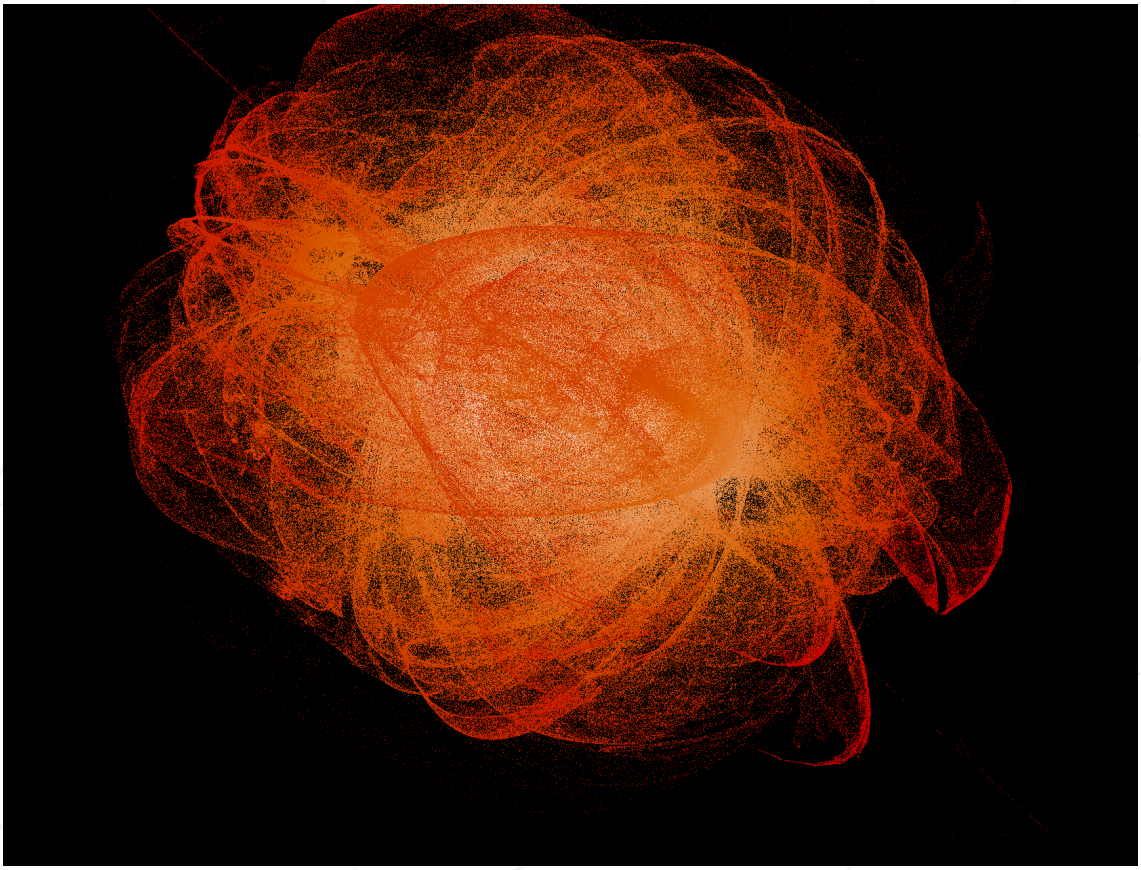
- Core in version 4.5 (2014)
- Core since version 4.3 (2012)

The compute shaders it's great, simple to use and it makes the coffee when it comes to mixing and rendering the calculation. They are present for OpenGL 4.3 (2012 therefore). In 2015, MacOSX supports OpenGL 4.1 HIGH (2009). So .. the compute shaders it's top. But for this, you can not use them. Why ? Because Mac.

Chapter II

Introduction

In this project, you will have to implement a system of particles. With a large number of particles. To do this you will need to use the GPU for its ca- pacity to massively parallelize calculations.



Here's what the project could provide.

Chapter III

Objectives

With this project, you will learn to use the OpenCL API and OpenCL kernels language C to animate your particles.

In addition, you have the ffi Cherez using OpenGL, which will allow you to consolidate your knowledge der this API.

This topic is also an opportunity to begin to re fl ect optimization. Do not be afraid to seek testing and di ff erent approaches. In fact .. you have to do it. So be it, but it will be easier if you are happy to do so.

chapter IV

General instructions

- You will need to use OpenGL 4.0 minimum (of course with shaders) and OpenCL 1.2 minimum.

- The particles should never be allocated on the RAM. Never. Even ini- tialisation. Never. If you are wondering if at any given time it is possible, the answer is: "Never.". Everything will be allocated on the video memory.

- In terms of performance, you have to turn at least a million particles at 60 FPS and three million (minimum) to 20 FPS. Make the number of particles is easily mutable.

- You have to make interoperability and for the sake of cleanliness, you must synchronize the memory between the di ff erent API. The acquisition and release functions are friends.

- You are free to use the language you want.

- You can use the graphics library of your choice (SDL2, SFML, GLFW, MLX, Glut ...).

- A Make the fi or equivalent is required. Only what is on your deposit will be evaluated.

chapter V

mandatory part

You will need to implement a system of particles. These particles will be initialized in a sphere or a cube and it will be possible to move from one to another with the inputs. They may be attracted to an activatable center of gravity and deactivated with a button. This may either be static (under the mouse at the time of input), or follow the mouse of the input.

They will be allocated on the GPU memory (VRAM). In addition, it will be necessary to meet certain performance constraints despite the large number of particles: one million particles at 60 fps and three million to at least 20 (and here you will understand why the GPU is good).

You will also need a ffi expensive the number of FPS in the window (title for example), this will make the simplest benchmarks.

To the delight of the retina it is necessary to put colors. They will have to depend at least the distance of the particles relative to the cursor. Remember that cleanliness code for interoperability is really important. If you do not see what we are talking, review the general instructions.

The particles do not need to have mesh.

Chapter VI Part

bonus

When you are sure that everything works and your performance is good, here are some bonus it would be good to add:

- A camera that can move in the particles, with controllable WASD and mouse. Because it is class.

- Issuers generating particles having a limited life. There will be dedicated to these points and other bonuses to your creativity.

For bonus, performance requested in the constraints do not apply ... do not overdo it. 10,000 particles at 20 fps, it's abuse.

chapter VII

Rendering and peer-assessment

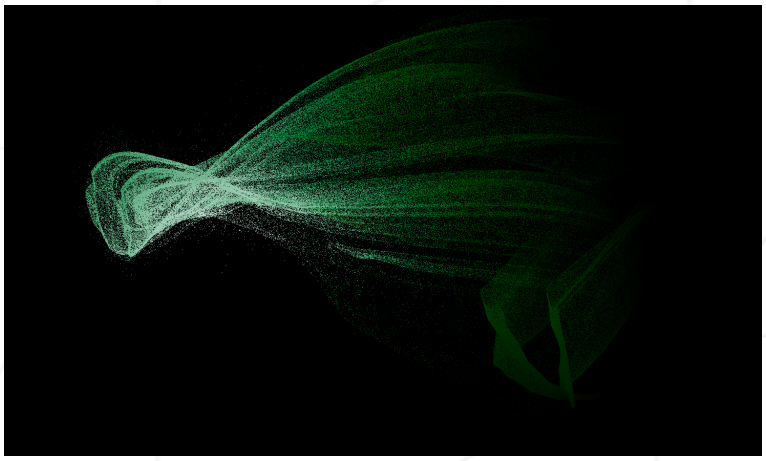
Meet your work on your deposit GiT as usual. Only this work on your deposit will be evaluated in defense.

Each constraint is veri fi ed so be sure to respect them. A non re- stress tested regularly era means a project failure.

Chapter VIII

Illustrations

Particles following the cursor:



And rotating around a center of gravity: