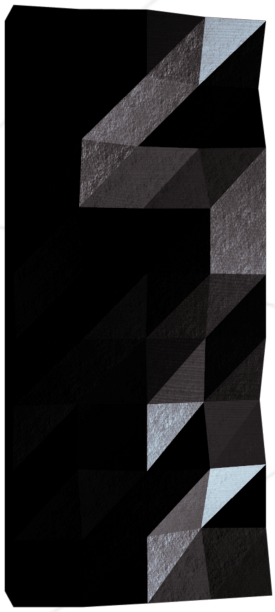


ShaderPixel

Shaders' complexity

Summary: Learn to create shaders for complex object rendering... or just to make cool things.

Contents

Chapter I

Foreword

Here's what you can read about NyanCat's origins on Wikipedia

I.1 Animated GIF

On April 2, 2011, the GIF animation of the cat was posted by 25-year-old Christopher Torres of Dallas, Texas, who uses the name "prguitarman", on his website LOL-Comics. Torres explained in an interview where the idea for the animation came from: "I was doing a donation drive for the Red Cross and in-between drawings in my Livestream video chat, two different people mentioned I should draw a 'Pop Tart' and a 'cat'." In response, he created a hybrid image of a Pop-Tart and a cat, which was developed a few days later into the animated GIF. The design of Nyan Cat was influenced by Torres' pet cat Marty, who died in November 2012 from feline infectious peritonitis.

I.2 Song

The original version of the song "Nyanyanyanyanyanyanya!" was uploaded by user "daniwell" to the Japanese video site Niconico on July 25, 2010. The song features the Vocaloid Hatsune Miku. The Japanese word nya is onomatopoeic, imitating the call of a cat (equivalent to English "meow").

On January 30, 2011, a user named "Momomomo" uploaded a cover of "Nyanyanyanyanyanyanya!" featuring the UTAU voice Momone Momo. The voice source used to create the Momone Momo voice was Momoko Fujimoto, a Japanese woman who lives in Tokyo.

I.3 YouTube video

YouTube user "saraj00n" (whose real name is Sara) combined the cat animation with the "Momo Momo" version of the song "Nyanyanyanyanyanyanya!", and uploaded it to YouTube on April 5, 2011, three days after Torres had uploaded his animation, giving it the title "Nyan Cat". The video rapidly became a success after being featured on websites including G4 and CollegeHumor. Christopher Torres said: "Originally, its name was Pop Tart Cat, and I will continue to call it so, but the Internet has reached a decision to name it Nyan Cat, and I'm happy with that choice, too."

Chapter II

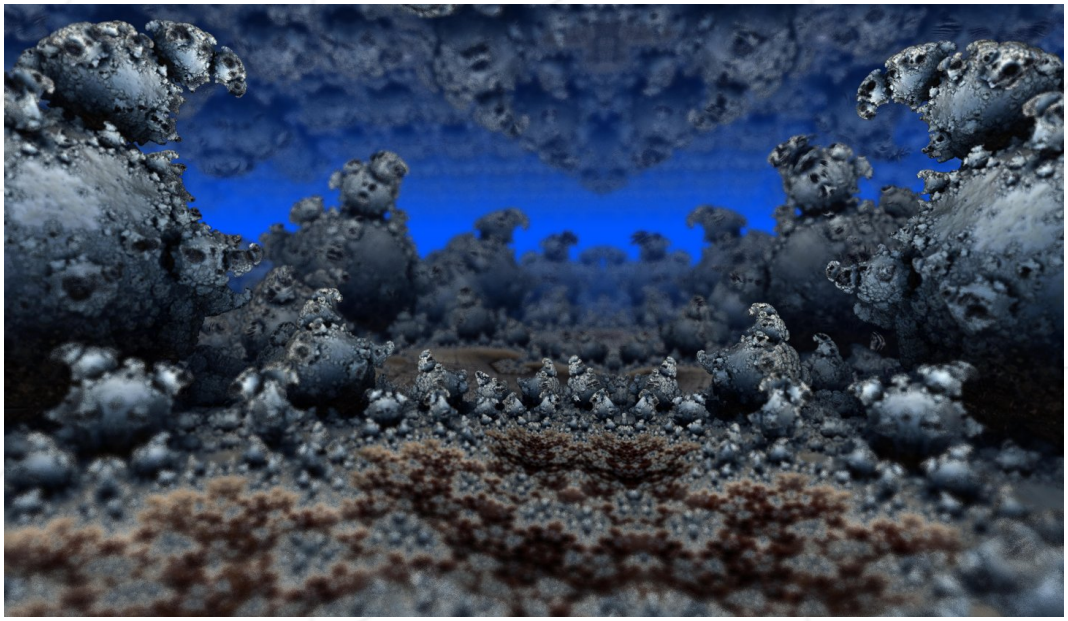
Introduction

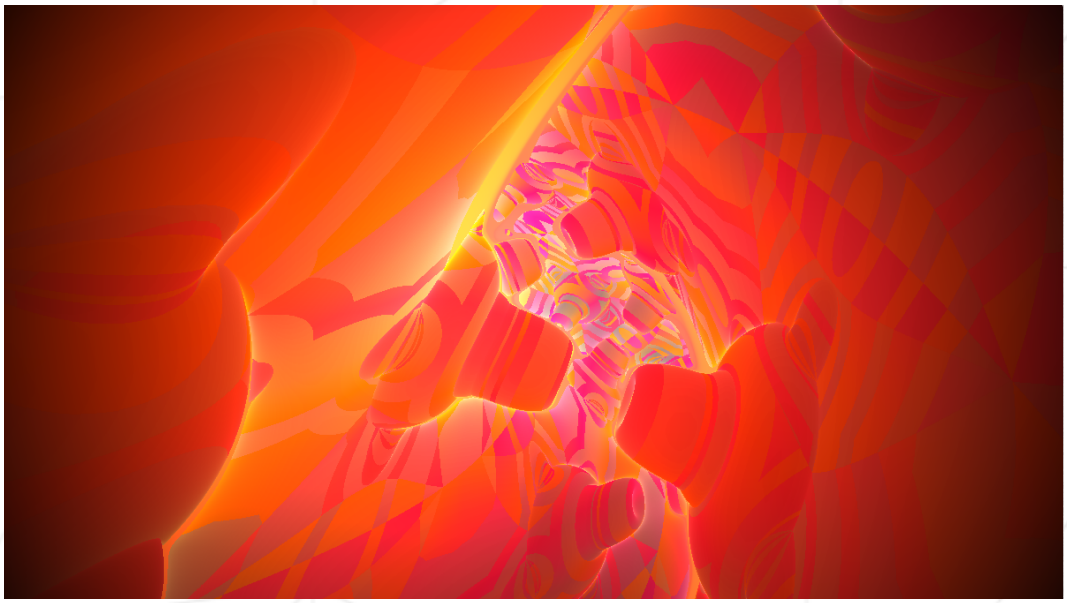
This project will make you discover the wonderful secret world hidden behind your shaders by exploiting multiple rendering techniques.

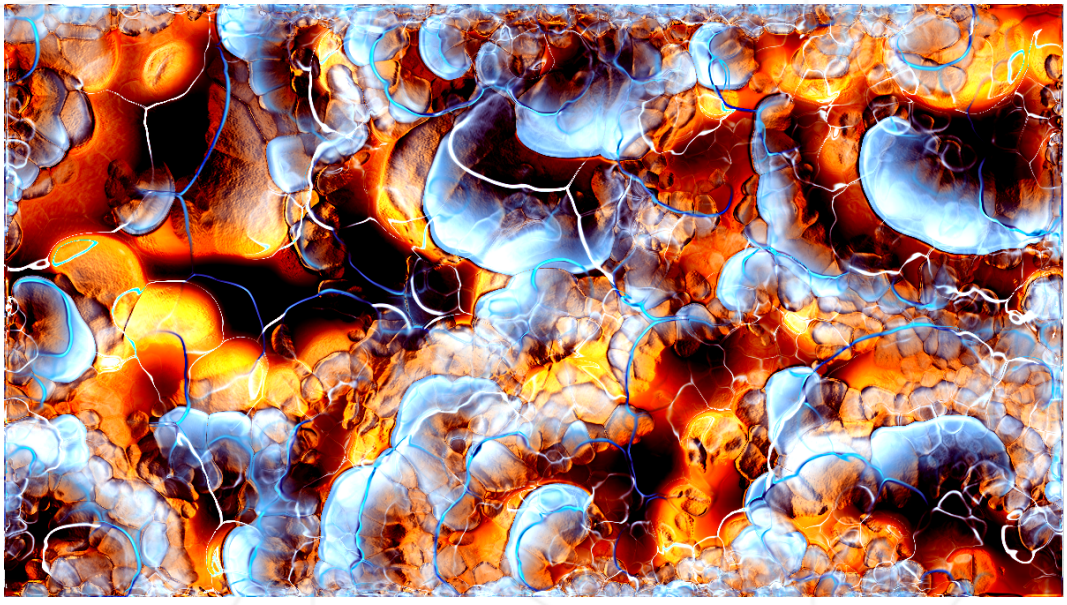
For this project, you will learn how to display 3D fractals, procedural worlds, clouds and many other weird things you have probably never heard of before, all within an exhibition window created to showcase your creations.

Go ahead and have fun, you can display whatever you like on your window: plasma, portals to other worlds, translucent objects, the sky is the limit - or not. In order to do that, you will have to learn how to draw with mathematic functions, use distance estimators, manage light in an efficient way and much more.

Here are some sample pictures of what you might be able to create (or not):







Chapter III

Goals

This project will introduce you to CGI through the use of shaders, and more precisely OpenGL/Vulkan fragment shaders.
You will also have to learn how to optimize your displays and light effects if you don't want to drop to 5 fps.
But above all, this project is supposed to be about creativity and imagination. So have fun, make sure that your shaders and scenery are as unique and interesting as possible :)

Here are some useful resources:

- OpenGL 4.0
- fractalforums
- hvidtfeldts
- the book of shaders
- iq's distance functions
- rendering a world with two triangles
- 2D shaders tutorial
- Volumetric rendering
- Volumetric raymarcher

Chapter IV

General instructions

- This project will be evaluated by humans, so you can organize your resources as you wish.
- You should provide a Makefile or something similar (cmake, premake, automake) which will compile your project.
- You can use any language you want, but take into account the performance issues! (If you have no idea what you're doing, go for C++. Great choice.)
- IF you are using OpenGL, you should use at least the v4.0 version of OpenGL, and the 330 version for the shaders.
- You can use Vulkan to replace OpenGL. Use the latest version available.
- You can use all the libraries you want for window management, objects loading (images or 3D models for your display window), matrix / vector / quaternions computation, sound and GUI. But pay attention to what you'll push on your repo!

Chapter V

Mandatory part

V.1 The scenery

Your scenery should include two types of objects:

- The first type of objects will be used only as an environment. You can think of it as a building where art pieces are being displayed. You can apply wahtever texture / shader you want to this background.
- The second type will support the shaders' rendering. You will therefore have to find several types of objects in order to apply all the requested shaders, so that they can render the desired object in the end.

You should be able to move freely within the scenery with the keyboard and mouse so that you can explore all the various shaders displayed there.

As you may have understood by now, this project will make you create your very own artist portfolio, a showcase of super-stylish shaders with plenty of visual effects and breathtaking objects.

V.2 The shaders system

For your shaders, feel free to invent any system that properly displays raymarched objects in your window (quite like holograms). For your uniforms, you can find inspiration there: shadertoy. The examples listed here are detailed enough to meet this project's requirements.

V.3 The shaders

For this project, you have to code the shaders listed below.
You are totally free to design the scenery as you wish, as long as you respect all the following requirements. As for the perspective, you should assume that the objects contained in the shaders belong to the real world and so manage a camera that can go inside these objects.

- A mandelbox, with lights (shadows are optionals).
- A 3D Fractal of your choice (other than the mandelbox), with light, surrounding occlusion and shadows.
- An IFS.
- A transluscent object (using volumetric ray-marching) with non-volumetric diffuse and specular lighting on its surface (like colored glass marble). Think to implement a mode to diplay one the specular/diffuse term of the lighting to verify that your lighting works.
- A volumetric cloud with a volumetric light, plus an object (volumetric too) inside with a higher density. The denser object must cast (volumetric) shadows on the cloud (i.e you must manage the light absorption inside the participating media). Example for the cloud, without the object.
- A shader applied on a surface, evoking "another world" in 3D, quite like a portal or a window. Pay attention to the perspective, the rendering must like a window in real life.
- A 2D shader of your choice that uses a renderbuffer.
- A 3D (or 4D) shader of your choice.

Pay close attention to the way you code your shaders. Keep in mind that a clean and well organized code is always better than a tangled pile of obscure operations with one-letter variables :)

Take the time to think about the objects that will frame your shaders, try to be original, to think outside the box and go further than a simple shader rendering inside a cube...

Chapter VI

Bonus part

For the bonuses for this project, you are free to do literally whatever you want, even just increase the size of your exhibition with ever more shaders.

If you're short on ideas, here is a brief, non-comprehensive list of options:

- A Hot-Reload for your shaders, if their sources are modified.
- Add more lights in your scenery that impact your shaders.
- 3D objects in your shaders that project shadows in your scenery.
- Settings inputs that appear when the view gets close to a shader, and that immedtialy change the said shader.
- Real portals that lead you inside a shader (with a way out).
- Shaders that take sound as an input.
- Support VR.
- Stage the scenery as AR on a table.
- Collision for the camera when you are inside of objects.
- Post processing such as bloom, tone mapping, motion blur, and so on.

Chapter VII

Turn-in and peer-evaluation

Turn in your work using your `GiT` repository, as usual. Only the work that's in your repository will be graded during the evaluation.