



ShaderPixel

Complexification des shaders

Antoine Lelievre alelievr@student.42.fr
42 Staff pedago@staff.42.fr

Résumé: Apprenez à créer des shaders pour faire le rendu d'objets complexes (ou des trucs cool tout simplement).

Table des matières

I	Préambule	2
I.1	Animated GIF	2
I.2	Song	2
I.3	YouTube video	2
II	Introduction	3
III	Objectifs	5
IV	Consignes générales	6
V	Partie obligatoire	7
V.1	La scène	7
V.2	Le système des shaders	7
V.3	Les shaders	8
VI	Partie bonus	9
VII	Rendu et peer-évaluation	10

Chapitre I

Préambule

Voici ce qu'on peut lire sur les origines du NyanCat sur 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 "Nyanyanyanyanyanya!" 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 "Nyanyanya-nyanyanya!" 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 "Nyanyanyanyanya!", 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."

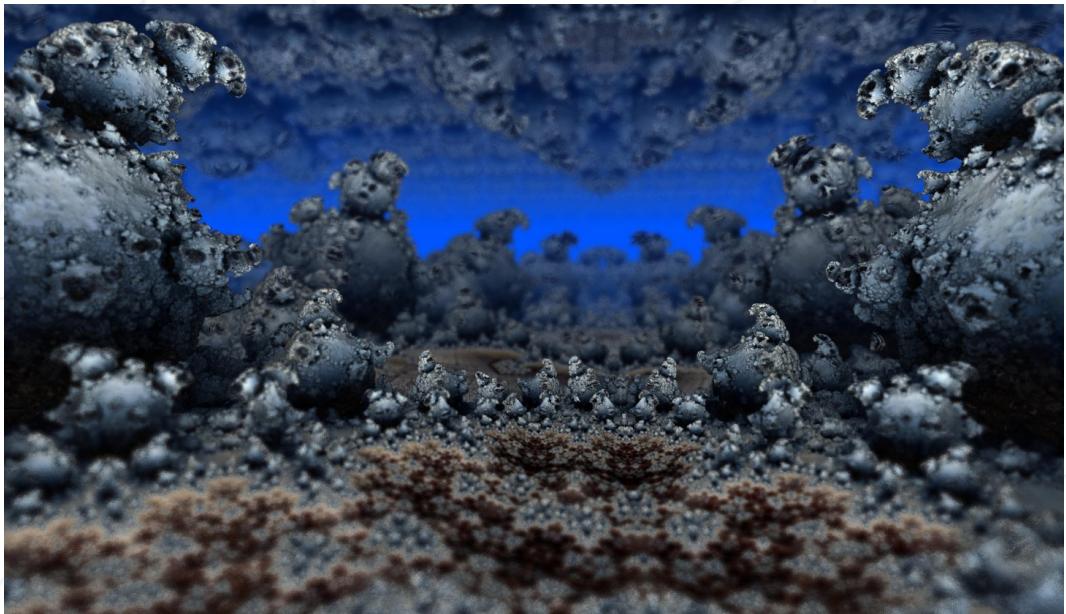
Chapitre II

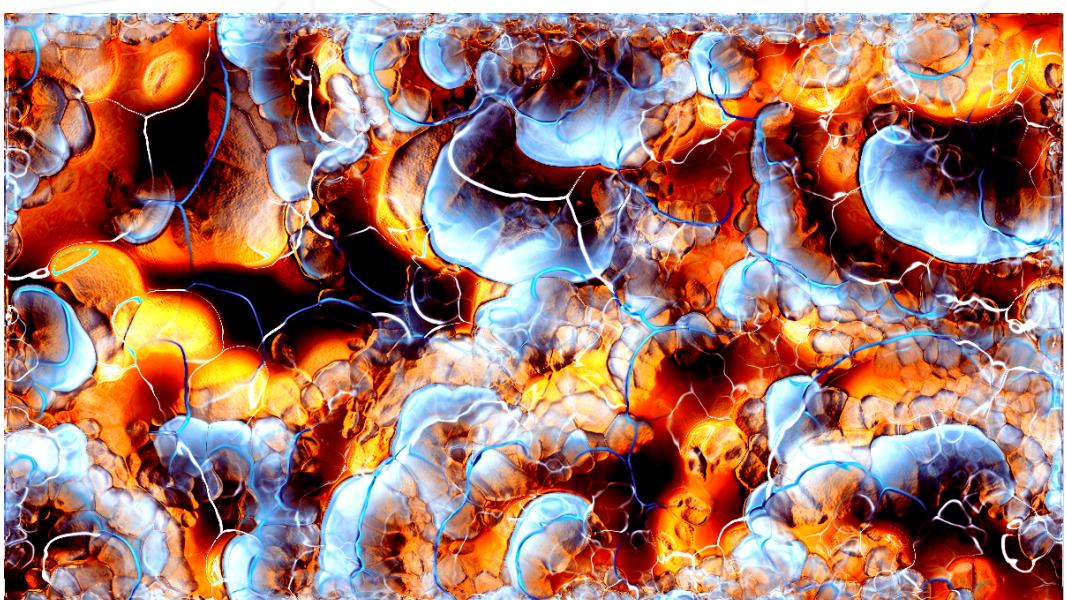
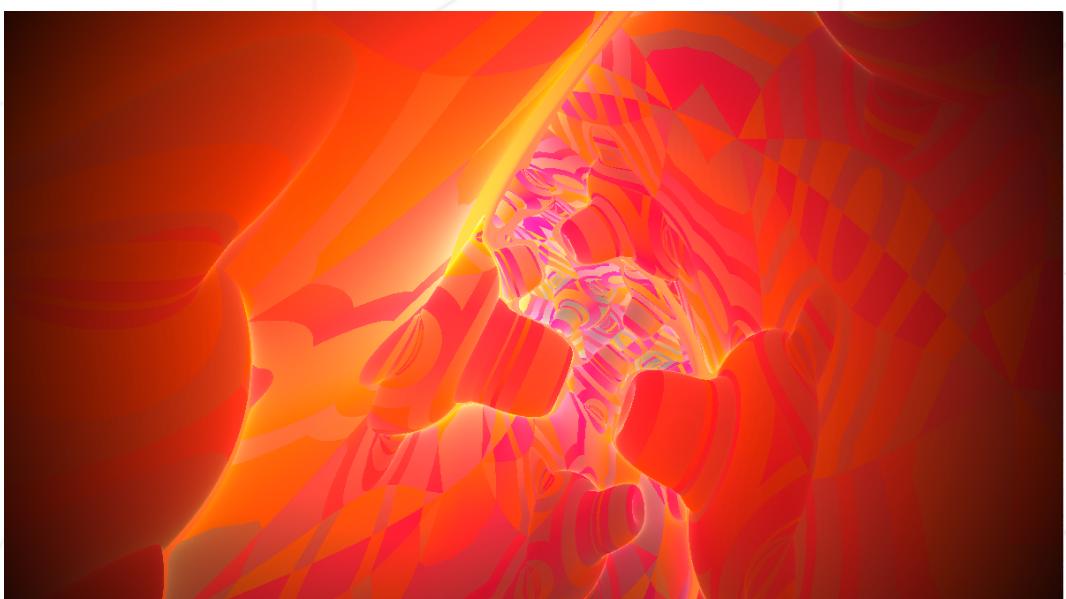
Introduction

Ce projet a pour but de vous faire découvrir le merveilleux monde caché derrière vos shaders en exploitant différentes techniques de rendu. En réalisant ce projet, vous apprendrez à afficher dans une scène d'exposition, des fractales 3D, des terrains procéduraux, des nuages et pleins d'autres trucs bizarres dont vous ne soupçonnez peut être pas l'existence :)

Vous pourrez donc vous amusez à afficher différentes choses tels que du plasma, des portails vers d'autres mondes, des objets translucides dans votre scène OpenGL. Pour ce faire, vous apprendrez à dessiner avec des fonctions mathématiques, à utiliser des estimateurs de distances, à gérer la lumière de façons efficace et [plus](#) encore.

Voici quelques images de ce que vous pourrez (ou pas) réaliser :





Chapitre III

Objectifs

Vous allez ici découvrir la synthèse d'image à partir de shaders et plus précisément de fragment shaders OpenGL.

Vous devrez aussi apprendre à optimiser vos scènes et votre illumination pour éviter de tomber à 5 fps.

Mais avant tout, ce projet est et doit rester purement créatif. Donc, amusez-vous à faire vos shaders et votre scène pour avoir le rendu le plus unique et intéressant possible :)

Voici quelques ressources qui vous seront utiles :

- [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](#)

Chapitre IV

Consignes générales

- Ce projet sera corrigé par des humains. Vous pouvez donc organiser vos ressources comme vous le souhaitez.
- Vous devez fournir un Makefile ou équivalent (cmake, premake, automake, ...) pour compiler votre projet.
- Vous êtes libre sur le choix du langage. Cependant, attention aux performances de ce dernier ! (si vous n'avez aucune idée, le c++ est un très bon choix).
- Vous devez utiliser la version 4.0 d'OpenGL minimum ainsi que la version 330 min pour les shaders.
- Vous avez le droit d'utiliser une ou plusieurs librairies pour gérer la fenêtre, le chargement des objets composant votre scène (images ou modèles 3D), les calculs de matrices / vecteurs / quaternions, le son et le GUI. Mais faites attention à ce que vous poussez !

Chapitre V

Partie obligatoire

V.1 La scène

Pour votre rendu, vous devrez intégrer deux types d'objets dans votre scène :

- Le premier type d'objet sert uniquement de déco, vous pouvez comparer cela à un bâtiment dans lequel des œuvres sont exposées. Vous êtes libre d'appliquer n'importe quel texture / shader sur cette déco.
- Le second type d'objet sert de support aux shaders pour faire leurs rendus. Vous devez donc trouver plusieurs objets vous permettant d'appliquer l'ensemble des shaders demandés pour qu'ils puissent afficher l'objet voulu.

Vous devez pouvoir vous déplacer dans la scène avec le clavier et la souris pour explorer les différents shaders exposés dans la scène.

Vous l'aurez compris, vous allez créer votre portfolio d'artiste remplis de shaders super stylés avec plein d'effets visuels et d'objets à couper le souffle.

V.2 Le système des shaders

Pour vos shaders, vous êtes libre d'inventer n'importe quel système vous permettant d'afficher des objets raymarchés dans la scène OpenGL (un peu comme des hologrammes). Pour vos uniforms, vous pouvez vous inspirer de ceux de [shadertoy](#), ils sont suffisamment complet pour répondre aux attentes de ce sujet.

V.3 Les shaders

Pour ce sujet, vous devrez coder les shaders listés ci dessous.
Vous avez une liberté total sur le rendu final tant que toutes ces conditions sont respectés.
Pour ce qui est de la perspective, vous devez faire comme si les objets dans les shaders appartenaient au monde et ainsi gérer une camera qui peut rentrer dans vos objets.

Au menu, nous vous demandons de réaliser :

- Une mandelbox avec de la lumière (ombres non obligatoires).
- Une fractale 3D de votre choix (autre que mandelbox) avec une lumière, de l'occlusion ambiante ainsi que des ombres.
- Un IFS.
- Un objet translucide (en raymarching volumétrique) avec de la lumière diffuse non volumétrique sur sa surface ainsi que du spéculaire ([un peu comme une bille de verre colorée](#)). Penssez aussi à implémenter un mode dans le shader pour afficher séparément le spéculaire et le diffuse pour vérifier le fonctionnement de votre lumière.
- Un nuage volumétrique avec de la lumière volumétrique ainsi qu'un objet (aussi volumétrique) d'une densité supérieur à celle du nuage faisant de l'ombre à l'intérieur du nuage. ([exemple pour le nuage](#)).
- Un shader appliqué sur une surface évoquant "un autre monde" en 3D un peu comme un portail ou une fenêtre. Attention à la perspective, le rendu doit être comme avec une fenêtre en vrais.
- Un shader 2D de votre choix utilisant un renderbuffer.
- Un shader 3D (ou 4D) de votre choix.



Attention à votre code dans les shaders, un code propre et bien organisé est toujours mieux qu'un tas d'opérations incompréhensibles avec des variables d'une lettre :)



Réfléchissez bien aux objets qui vont supportés vos shaders, essayez de faire des trucs originaux, d'aller un peu plus loin que la simple exposition de shaders dans un cube ...

Chapitre VI

Partie bonus

Pour les bonus, vous êtes assez libre de faire ce que vous voulez, y compris d'agrandir la taille de votre exposition en rajoutant toujours plus de shaders.

Pour ceux qui n'auraient pas d'idées, voici une liste non exhaustive de bonus :

- un Hot-Reload pour les shaders si leurs sources ont été modifiés.
- ajouter des lumières dans la scène OpenGL qui agissent sur vos shaders.
- vos objets 3D dans les shaders qui projettent de l'ombre dans votre scène OpenGL.
- des inputs pour faire des réglages qui apparaissent quand on se rapproche d'un shader.
- de vrais portails qui vous emmènent à l'intérieur du shader (avec un moyen de sortie).
- des shaders qui prennent du son en paramètre.
- support de la VR.
- faire la scène en AR sur une table.
- des collisions pour la camera quand on est à l'intérieur des objets.
- du post processing (bloom, tone mapping, motion blur, etc.)

Chapitre VII

Rendu et peer-évaluation

Rendez votre travail sur votre dépôt GiT comme d'habitude. Seul le travail présent sur votre dépôt sera évalué en soutenance.