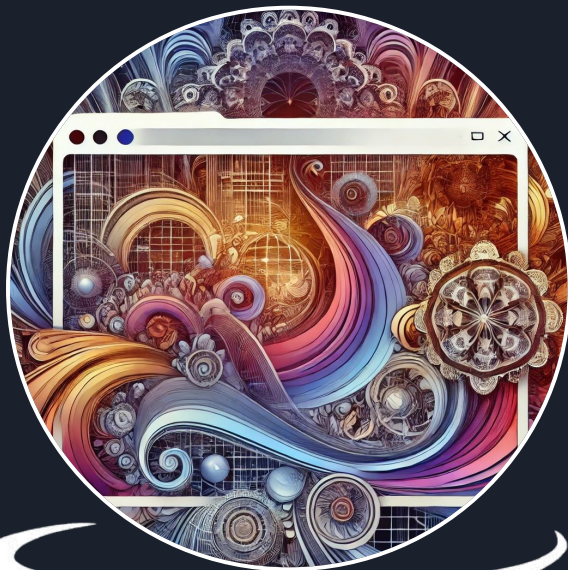




Pour des effets
hallucinants, même sur le
web

Les shaders, une
introduction

Simon Bédard

Professeur d'informatique

- › Canadien (en France depuis 2015)
- › Freelance / professeur
- › Ancienne développeur en SSII
- › Développeur de jeu vidéo amateur
- › Ancien professeur d'art martiaux
- › Avec un adorable Cocker



Sujet d'aujourd'hui

- La place des shaders dans l'écosystème graphique des systèmes informatiques aka, un shader, c'est quoi ?
- Voir des exemples de shaders sur le web
- Découvrir ensemble comment créer un shader pour un site web avec GLSL et Three.js

13^{ème} édition - du 16 au 18 avril 2025

3 jours de conférences, 70 exposants, 4500 visiteurs par jour



DEVOLX FRANCE

A propos ▾

Billetterie

Agenda 2025 ▾

Exposants ▾

Présenter ▾

Contact

13^{ème} édition - 16 au 18 avril 2025 - Paris, France

Devovx France

3 jours de conférences et de rencontres

🇫🇷 Français

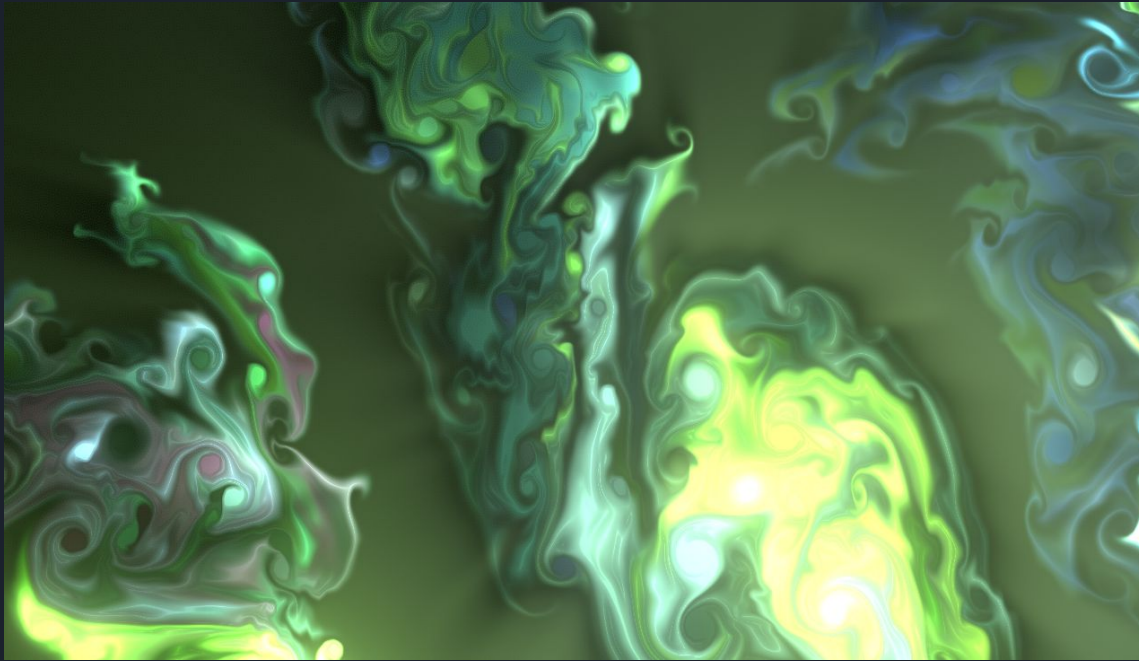
▾ <script>

```
window.RS_MODULES = window.RS_MODULES || {};  
window.RS_MODULES.modules = window.RS_MODULES.modules || {};  
window.RS_MODULES.waiting = window.RS_MODULES.waiting || [];  
window.RS_MODULES.deferred = true;
```

```
    window.RS_MODULES.waiting =  
window.RS_MODULES.waiting.concat([  
    "fluidynamics", "threejs", "mousetrap"]);  
    window.RS_MODULES.moduleWaiting =  
window.RS_MODULES.moduleWaiting || {};  
    window.RS_MODULES.type = 'compiled'; == $0
```

</script>

Site Devovx

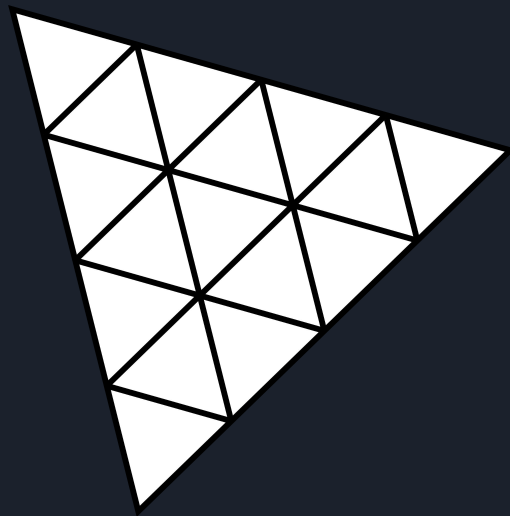


Slider revolution

Simulation des
Fluides

ThreeJS

- Projet OpenSource
- Commencé en 2010
- Permet de faire de la 3D dans votre navigateur
- “Purement” en javascript



Lien - Ressources Web

Projets Three.js

Exemples Three.js



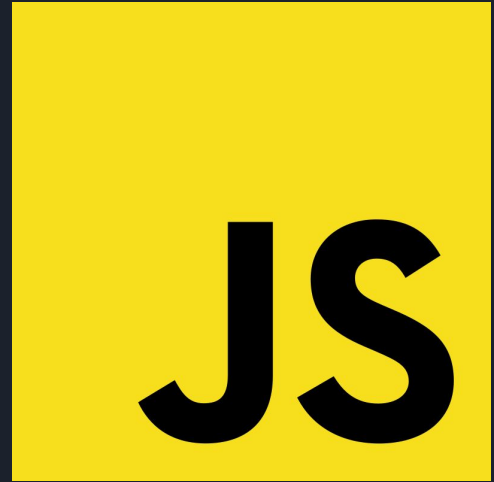
GLSL (OpenGL Shading Language)

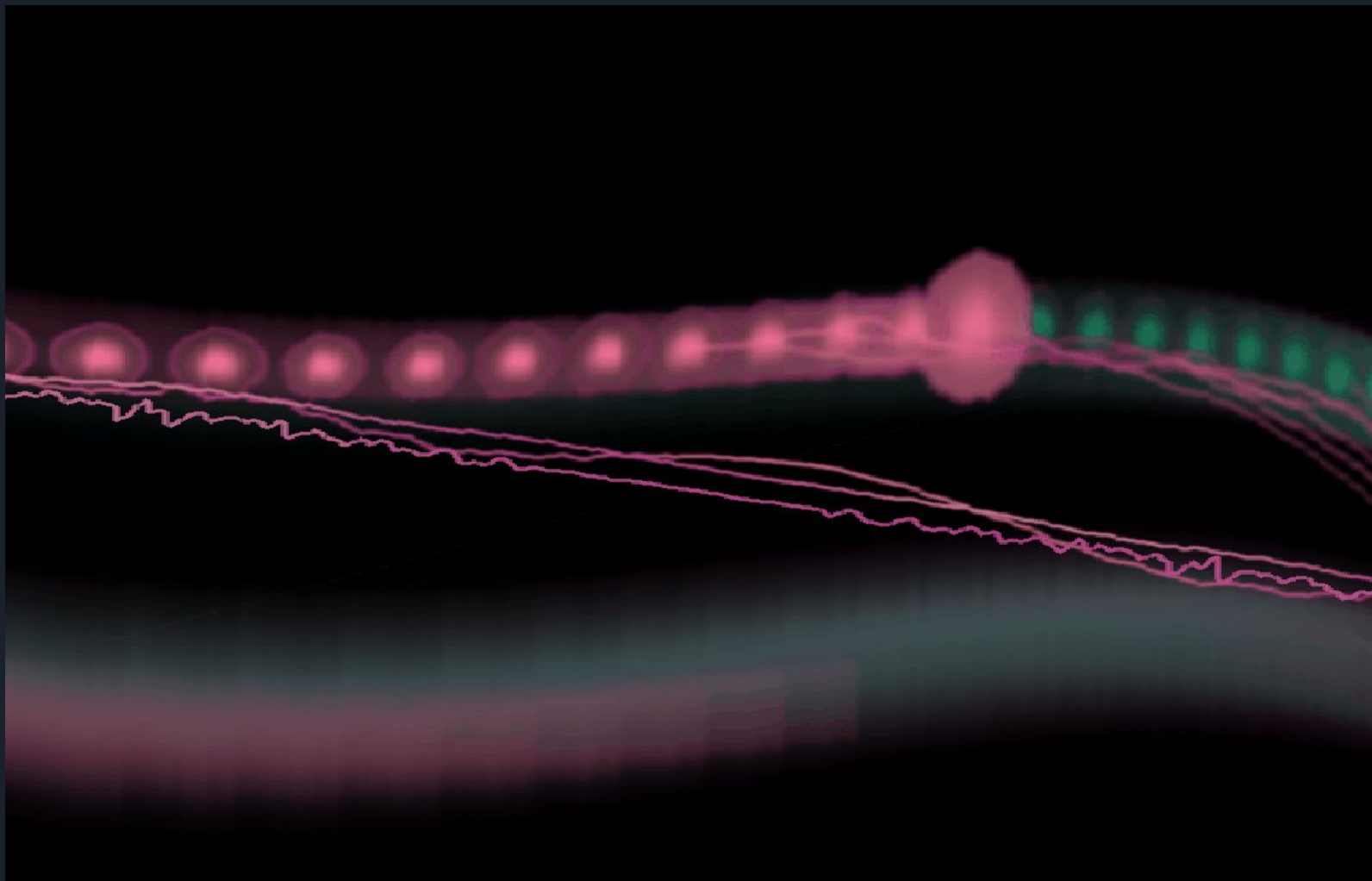


- Le langage le plus utilisé pour les shaders
- L'alternative principale est le HLSL (High-Level Shading Language pour DirectX)
- Support multiplateforme
 - Il existe des alternatives de plus bas niveau pour une optimisation directe du GPU



+





[Media Player
Exemple](#)



[Windows Media
Visualizations](#)



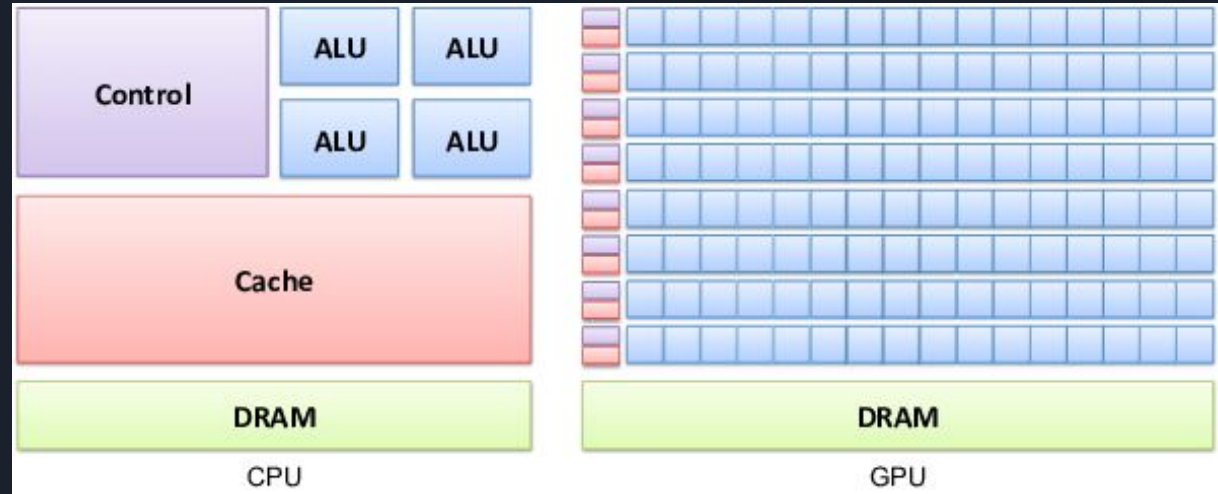
Shaders : Le Contexte

Un shader c'est quoi ?

Un shader est un programme dédié au GPU, appliqué indépendamment sur chaque pixel de chaque image pour créer des effets graphiques complexes en temps réel.

Un shader, pourquoi ?

1920 x 1080
2,073,600
(+deux millions pixels)
60 Frame Seconde
124,416,000
Calcule / Seconde

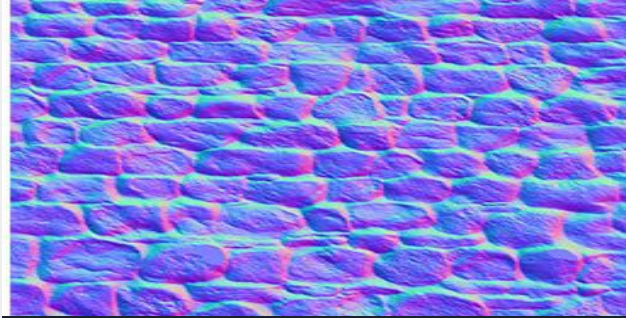




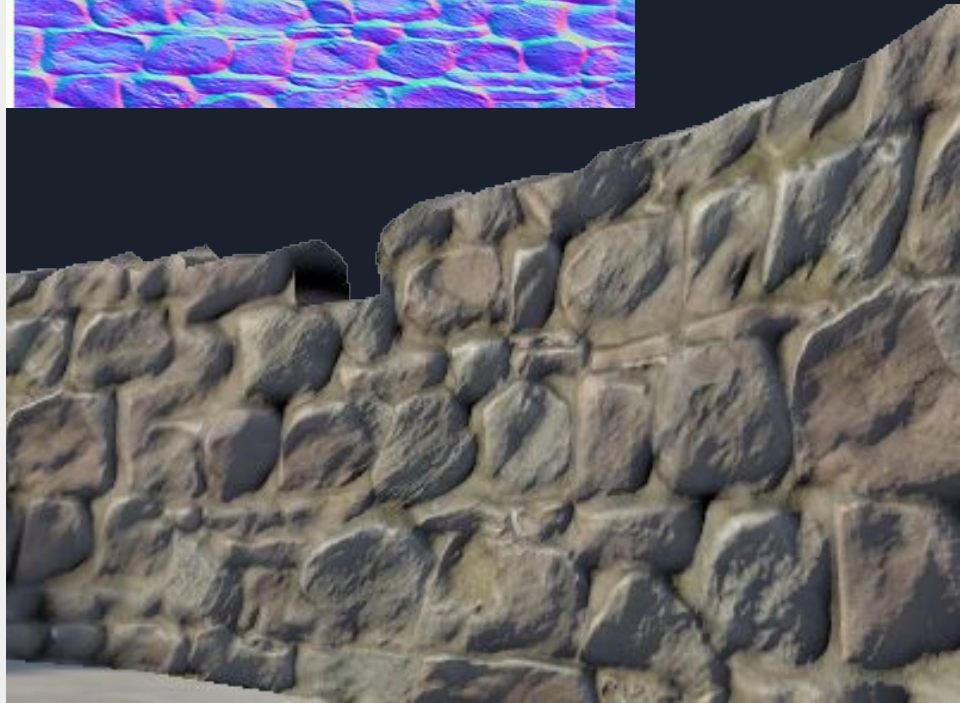
[La Thèse](#)

Historiquement

- ❑ Edwin Catmull
- ❑ Née en 1945
- ❑ Créer le premier shader en 1974 (âge 29)
- ❑ Pour son doctorat à l'université de l'Utah
- ❑ Simulation d'effets de surface complexes
- ❑ Un des fondateurs du studio Pixar



Unity



Effet texture



[Godot](#)



Effet d'ombre

Beyond Belief Shaders



BSL Shaders



Continuum Shaders



Oceano Shaders



Default



ProjectLUMA Shaders



SEUS Shaders



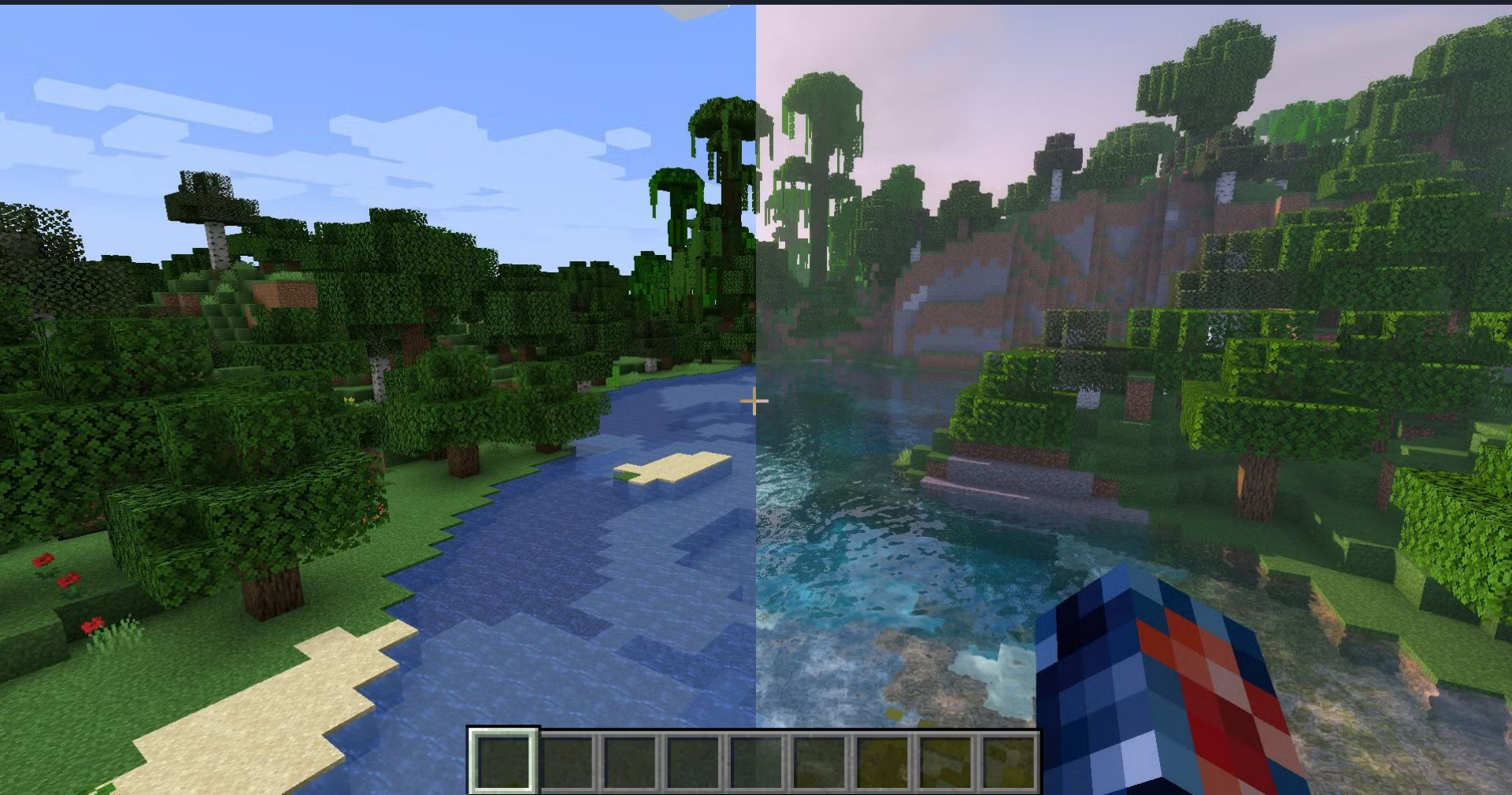
Sildur's Shaders



Sora Shaders



Effet d'eau





Exemple en web

ShaderToy



Inigo Quilez



Pol Jeremias Vila

Shader Art Coding Introduction



Balatro Background Shaders



Cannelés



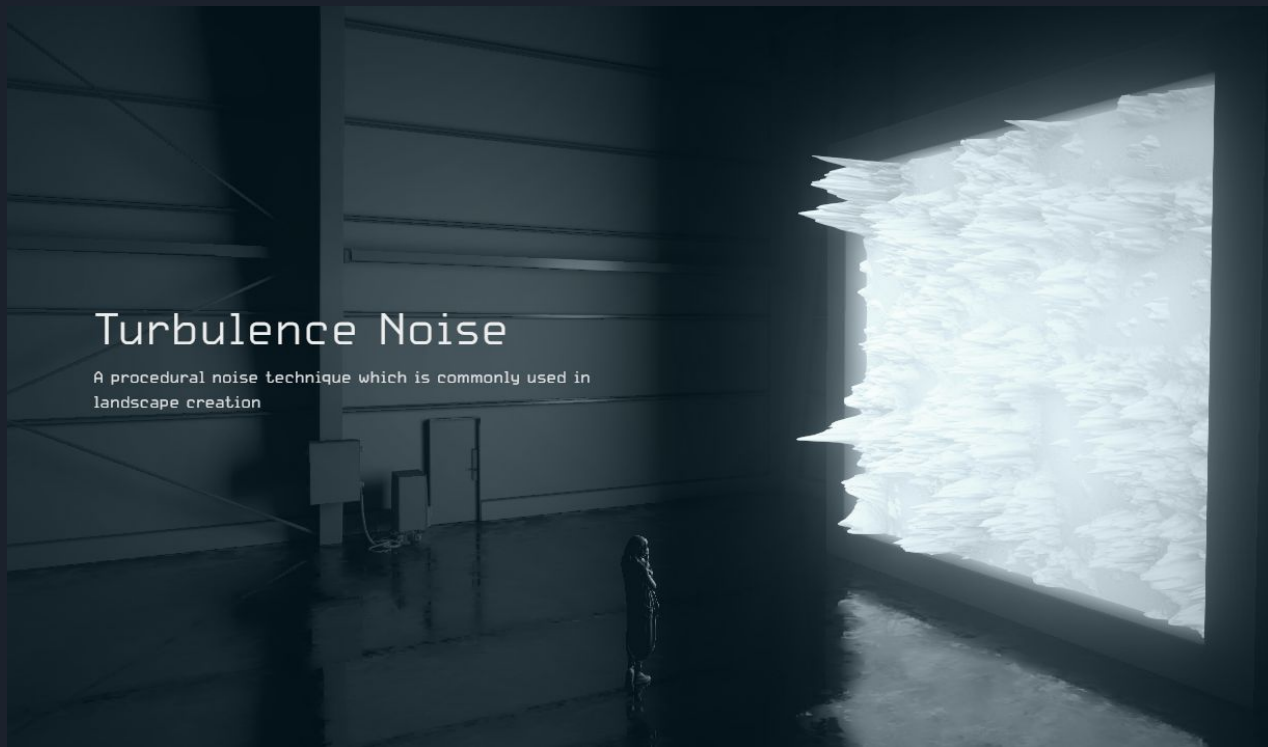
Pont de Pierre



Cineshader

Turbulence Noise

A procedural noise technique which is commonly used in landscape creation





Active Theory

Sustainable Horizons



Lien - Ressources Web

Projets Three.js

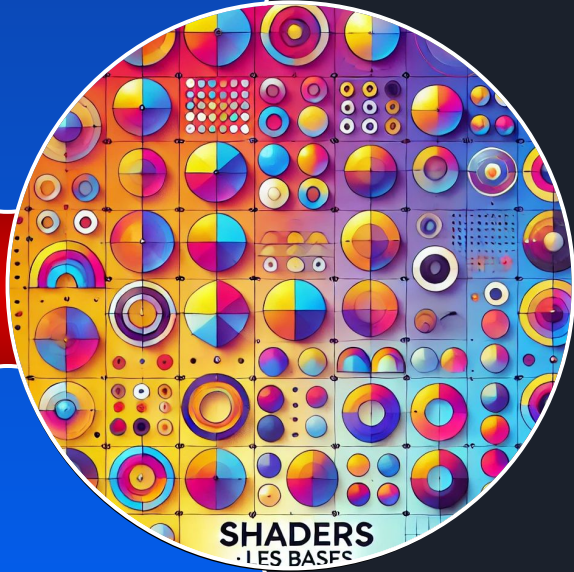


Les avantages

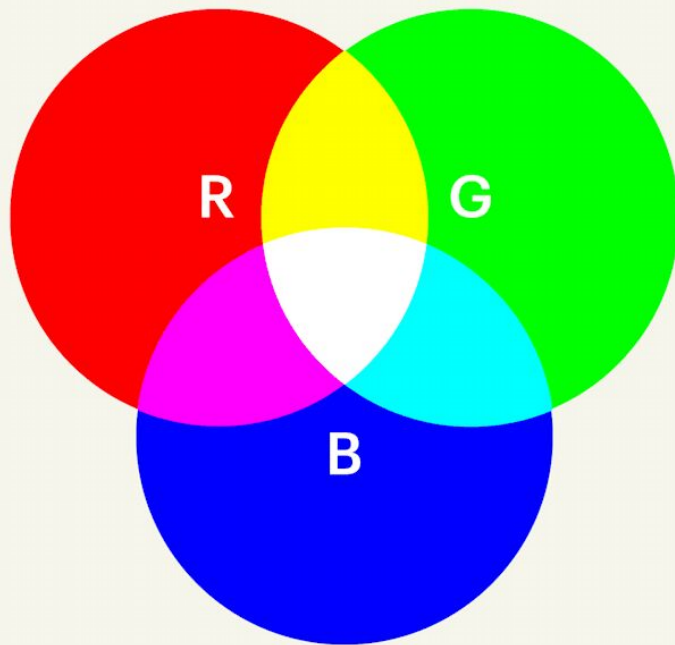
- Taille relativement légère
 - Une fois que le moteur est en mémoire.
- Calcule fait sur le GPU, une ressource sous exploité par votre navigateur
 - Si l'accélération des ressources est ON

Donc, dans le web ?

- Encore peu utilisé professionnellement.
- Permet des effets incroyables
- Permet à l'utilisateur plus d'intimité avec le site



Shaders : Les bases



Red

Green

Blue

R : Rouge

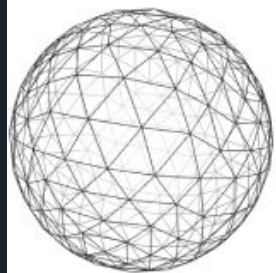
G : Vert

B : Bleu

Valeur entre 0 / 1

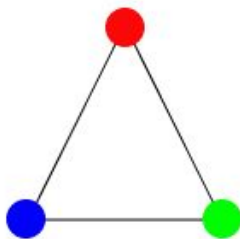
Typage fort des variables

- Typage fort sans casting automatique
- 4  n'est pas un float
- 4.0  est un float
- 4.  est un float



Mesh

Attributes
→



Vertex Shader



`gl_Position`

Uniforms



Varyings
→



Fragment Shader



`gl_FragColor`

Les types de Shaders

Vertex

- Divise mes éléments
- Est exécuté en premier

Fragment

- Applique leurs couleurs à chacun des pixels
- Est activé après le vertex

Le vertex de base

```
varying vec2 vUv;
```

```
void main() {
```

```
    vUv = uv;
```

```
    gl_Position = projectionMatrix * modelViewMatrix * vec4(position, 1.0);
```

```
}
```

- Créer un vertex qui couvrira à lui seul l'ensemble de l'objet
- Prends les coordonnées de la texture générées par ce vertex et stocke-les dans vUv.
- Set une perspective neutre (position, 1.0).
- Adapter l'espace au point de vue de la caméra (projectionMatrix * modelViewMatrix).
- Assigne le résultat final à gl_Position pour définir où le vertex apparaîtra à l'écran.

Le fragment de base

```
varying vec2 vUv ;
```

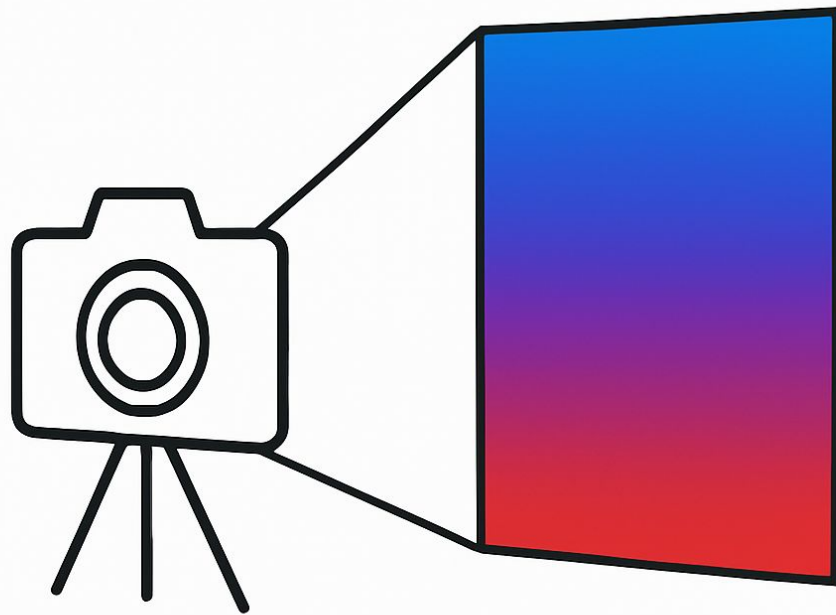
```
void main() {
```

```
    vec3 color = vec3(0, 0, 1.0 - vUv.x);
```

```
    gl_FragColor = vec4(color, 1.0);
```

```
}
```

- Attrape la position de ton pixel fournie par le vertex (`varying vec2 vUv`)
- Créer une couleur avec aucun Rouge, aucun Vert `vec3 color = vec3(0, 0`
 - et un bleu allant du absolue au vide `(1.0 - vUv.x)`
- Applique cette couleur sans transparence `gl_FragColor = vec4(color, 1.0);`



CAMERA

PLANE



Lien - Ressources Web

Github

Version Site





Processus Artistique

Lancé par IA

Claude, créer moi un shader pour représenter la Devoxx de Paris. Ce shader sera ensuite exécuté sur shaderToy

Voici le point d'entrée de shaderToy

```
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    // Normalized pixel coordinates (from 0 to 1)
    vec2 uv = fragCoord/iResolution.xy;
    // Time varying pixel color
    vec3 col = 0.5 + 0.5*cos(iTime+uv.xyx+vec3(0,2,4));
    // Output to screen
    fragColor = vec4(col,1.0);
}
```

Très bon point de départ



[The Book of Shaders](#)



[Shader Art Coding](#)
[Introduction](#)



Merci !



Simon Bédard

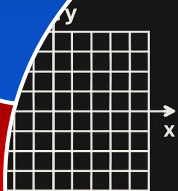


SimonBedardProg@Gmail.com



JavaKhanStudio

SHADER ART



COORDINATES



COLOR GRADIENT



DISTANCE
FROM CENTER



BASED
ON



SIMP

Rappel

Création d'un nuage de point



Type de variable

Uniforms

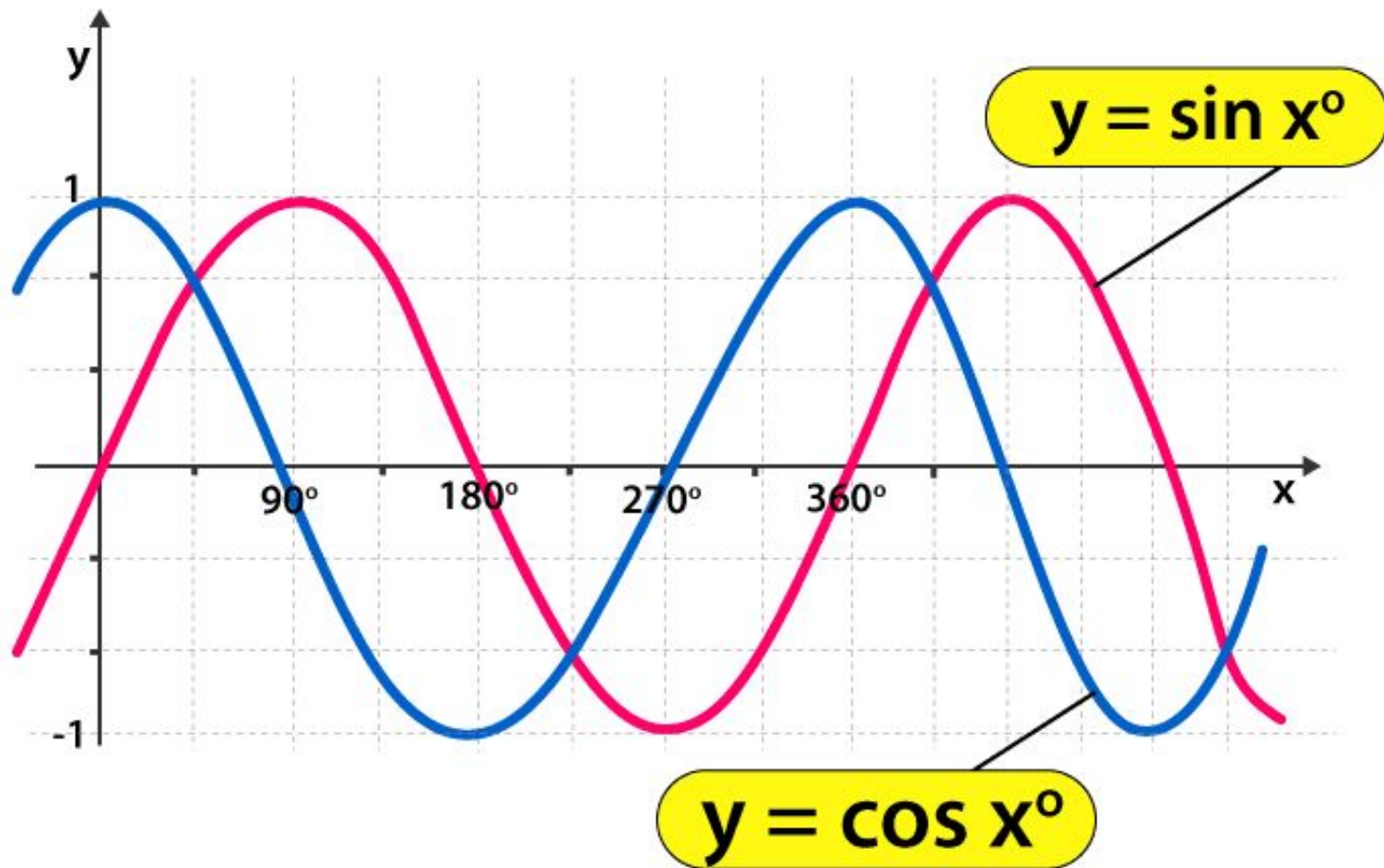
- Valeurs globales pour tout le shader pendant une exécution
- Reste la même valeur pour chaque pixel ou vertex traité
- Peut être injecté avec Javascript

Varyings

- Variables interpolées entre les vertices
- Important pour correctement optimiser
- Pas important pour ce qu'on fait aujourd'hui

Vector

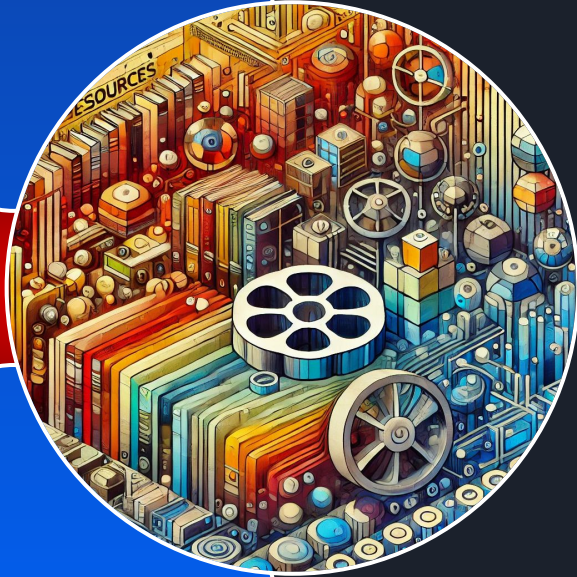
- Les vecteurs sont TRÈS utile pour faire des shaders
- `vec2(x,y)` `vec(x,y,z)`
- `vec4(r,g,b,a)`
- `vec4(vec3,a)`
- `vec3(vec3.xy,z)`
- La plupart des valeurs de base entre 0.0 et 1.0



Lien - Vidéo

Vidéo Concepts de
bases (ENG)





Ressources



Shader languages sup

slevesque



974 339



Syntax highlighter for shader language (...)

Installer



Mise à jour automatique



Shader Toy

v0.11.2

Adam Stevenson



63 479



Live preview of GLSL shaders similar to shadertoy

Installer



Mise à jour automatique



Lien - Slide

Inigo Quilez



Lien - Slide

IDE ShadeRed
Plugin-in Import



Lien - Ressources Web

[Water shader list](#)



Lien - Vidéo

Introduction to Shaders
What are shaders



Lien - Vidéo

Write GLSL



Lien - Ressources Web

Code Maison



Lien - Ressources Web

WebGL fluid
animation



Lien - Ressources Web

Reanimate
Visualizations

