Initiation aux Shaders avec Processing

Le hello World des shaders

shader_01

vert.glsl

```
uniform mat4 transformMatrix;

attribute vec4 position;
attribute vec4 color;

varying vec4 vertColor;

void main() {
  gl_Position = transformMatrix * position;

  vertColor = color;
}
```

freg.glsl

```
#ifdef GL_ES
precision mediump float;
precision mediump int;
#endif

varying vec4 vertColor;

void main() {
  gl_FragColor = vertColor;
}
```

shader_01.pde

```
PShader myShader;
int margin = 32;

void setup() {
 size(800, 600, P2D);

 myShader = loadShader("frag.glsl", "vert.glsl");
 noStroke();
}

void draw() {
  background(0);
  shader(myShader);
  fill(230, 120, 0);
  rect(margin, margin, width - 2*margin, height - 2*margin);
  resetShader();    // Désactive le shader, permet de redessiner normalement
}
```

Communication entre l'application et les shaders

L'application (programme Processing) peut envoyer des données vers les shaders par des variable déclarées avec le mot-clé `uniform`.

Fonctions Processing pour transmettre des données

```
set(String name, int x)
set(String name, int x, int y)  -> vec2
set(String name, int x, int y, int z)  -> vec3
set(String name, int x, int y, int z, int w)  -> vec4
set(String name, float x, ...)
set(String name, PVector vec)   -> vec3

set(String name, int[] vec, int ncoords) // Jusqu'à 4 coordonnées par élément
set(String name, float[] vec, int ncoords)

set(String name, PMatrix2D mat) -> mat2
set(String name, PMatrix3D mat) -> mat4
```

```
set(String name, PImage tex)    -> sampler2D
```

> Attention ! Lorsqu'on transmet des nombres entiers, comme par exemple : `set("u_resolution", 512, 512)`, soyez sûr d'avoir déclaré les variables `uniform` pour des types entiers, comme : `ivec2`, `ivec3`...
>
> Une autre solution est de les convertir en nombres flottants avant de les transmettre : `set("u_resolution", float(512), float(512))`

Textures

Fonctions utiles

Flou Gaussien

Code optimisé, d'après https://www.rastergrid.com/blog/2010/09/efficient-gaussian-blur-with-linear-sampling/
A exécuter en deux passes : horizontale et verticale

```
vec4 blur(sampler2D image, vec2 uv, vec2 resolution, vec2 direction) {
    const float offset[3] = float[](0.0, 1.3846153846, 3.2307692308);
    const float weight[3] = float[](0.2270270270, 0.3162162162, 0.0702702703);

    vec4 colorOut = texture2D(image, uv / resolution) * weight[0];
    for (int i=1; i<3; i++) {
        vec3 color = texture2D(image, (uv + direction * offset[i]) / resolution);
        color += texture2D(image, (uv - direction * offset[i]) / resolution);
        colorOut += color * weight[i];
    }
    return colorOut;
}
```

Librairies Processing

Quelques librairies externes pour l'utilisation de shaders dans Processing :

- https://github.com/diwi/PixelFlow

Ressources

Liste de liens incontournables pour approfondir et aller plus loin...

- https://thebookofshaders.com/
- https://www.shadertoy.com
- https://iquilezles.org/articles/functions/

Article extrait de : http://www.lesporteslogiques.net/wiki/ - **WIKI Les Portes Logiques**
Adresse : **http://www.lesporteslogiques.net/wiki/ressource/code/processing/shaders?rev=1662537876**
Article mis à jour: **2022/09/07 10:04**