TP : OpengGL 4 - *Shaders*
Practical Work: OpengGL 4 - *Shaders*

Jonathan Fabrizio
http://jo.fabrizio.free.fr/

Objectif

L'objectif de ce T.P. est de programmer des *Shaders*.

The goal of this practical work is to implement *Shaders*.

Note : Veillez à bien conserver les différentes versions de vos *Shaders*.

Note: Do not forget to save a copy of each version of your *Shaders*.

1 Travail préliminaire — Preliminary work

Téléchargez le squelette du programme. Regardez les sources et essayez de comprendre ce qu'il fait et comment il fonctionne.

Les *Shaders* sont dans les fichiers `vertex.shd` et `fragment.shd`. Vous devez compléter ces fichiers pour répondre aux questions posées.

Download the skeleton of the program. Have a look to the source code and try to understand the content of the program.

The *Shaders* are expected to be in the files `vertex.shd` and `fragment.shd`. You have to fill these files to answer to the following questions.

2 La couleur The color

Dans cette partie vous utiliserez seulement les champs `position` et `color`. `position` contient la position du sommet dans l'espace, `color` donne la couleur associée.

Question 1 : Complétez les *shaders* afin d'afficher l'objet en jaune.

Question 2 : Modifiez les *shaders* afin d'afficher la couleur de l'objet qui est passée par le *VBO* `color`. D'après vous, que représente cette couleur ?

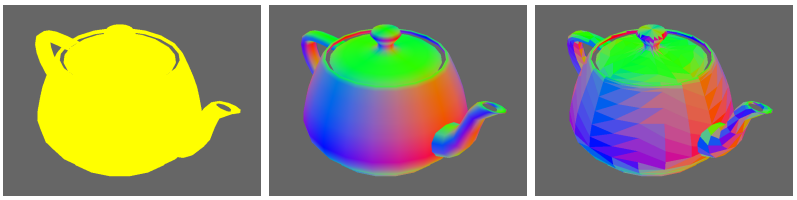
Question 3 : Modifiez les *shaders* afin de désactiver l'interpolation de la couleur entre les *vertex shader* et *fragment shader*. Vous devriez voir apparaître les triangles qui compose l'objet.

In this part, only the fields `position` and `color` are required. `position` provides the position of the vertex in space and `color` the associated color.

Question 1 : Complete the *shaders* in order to display the object in yellow.

Question 2 : Modify the *shaders* in order to display the object using the color provided in the *VBO* `color`. What is (in your opinion, represented by this color?)

Question 3 : Modify the *shaders* in order to disable color interpolation between *vertex shader* and *fragment shader*. You should then see the triangles of the mesh.



3 Les normales The normal vectors

Dans cette partie vous utiliserez encore le champs `position` et le champs `color`. En plus, vous utiliserez les deux champs `normal`. `normalFlat` donne, pour chaque sommet, la normale au triangle. `normalSoomth` donne, pour chaque sommet, une interpolation des normales des triangles adjacents.

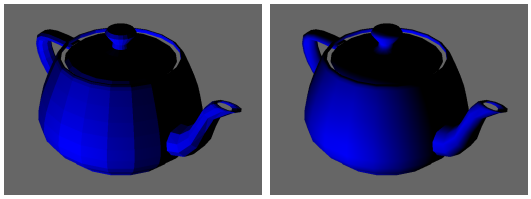
Question 1 : Affichez maintenant l'objet en bleu mais en tenant compte de la lumière. Chaque face doit avoir une couleur unis.

Question 2 : Lissez maintenant l'objet (toujours en bleu).

In this part, `position` and `color` fields will be still used but the two `normal` fields will be also required: `normalFlat` field provides, for each vertex, the normal vector of the related triangle and `normalSoomth` field provides, for each vertex, the interpolated normal vector.

Question 1 : Draw the object in blue according to the lighting (compute a uniform lighting on each triangle).

Question 2 : Next, compute the lighting but smooth the object (still in blue).



4 Les textures The textures

Dans cette partie vous utiliserez encore le champs `position` et `normalSoomth`. Vous utiliserez aussi le champs `uv` qui donne pour chaque sommet ses coordonnées dans le repère texture.

Question 1 : Affichez comme couleur : sur le canal rouge le u et sur le canal vert le v.

Question 2 : Affichez la texture sur l'objet (et uniquement la texture) sans tenir compte de la lumière.

Question 3 : Affichez la texture sur l'objet en tenant compte de la lumière.

Question 4 : Pour simuler l'effet d'un spot, ne tenez plus compte de la lumière mais combinez la texture (que vous pouvez un peu éclaircir pour un résultat plus joli) avec l'image du spot.

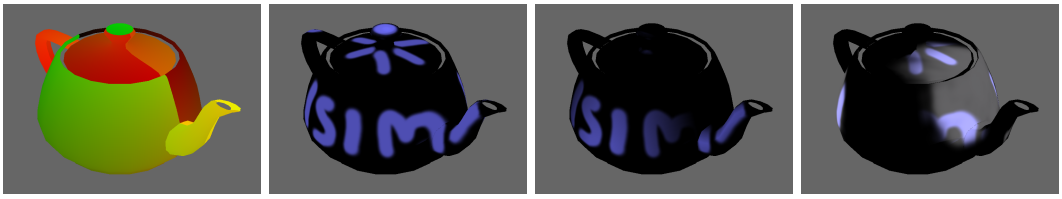
In this section, the fields `position` and `normalSoomth` will be used. You will also use the `uv` field which provides, for each vertex, the coordinates in the texture coordinate space.

Question 1 : Interpret the `uv` coordinates as a color: copy the value of u into the red channel and v into the green one.

Question 2 : Map (and then display) the texture onto the object (without taking into account the lighting).

Question 3 : Map (and then display) the texture onto the object and compute the lighting.

Question 4 : To simulate spot light effect, do not take care about the light but combine your spot light texture with the object texture (you can increase the bright of the texture to improve the result).



5 Des effets Multiple effects

Reprenez simplement votre objet avec la couleur d'origine.

Question 1 : Affichez l'image en niveau de gris

Question 2 : Affichez l'image en vert (style *night vision*. Pour faire plus style essayez de changer l'intensité en fonction des lignes - c.f. `gl_FragCoord`).

Question 3 : Affichez l'objet en vert et modulez l'intensité en fonction de la profondeur.

Question 4 : Ajoutez du brouillard (pensez à changer la couleur de fond).

Question 5 : Affichez l'objet sur un damier seulement (sur des carrés de 6×6 séparés par 2 pixels ; Entre les carrés il y aura la couleur de fond ou du noir).

Question 6 : Affichez une image plus terne.

Question 7 : Affichez un dégradé "couleur vers niveaux de gris" en fonction de la distance au centre de l'image.

Question 8 : A l'aide de *normalmap*, faites apparaître des bosses sur la théière. Les normales sont codées dans le repère tangent. Chaque composante est codée entre 0 et 1 et doivent être ramenée entre -1 et 1.

Restart with your object, with the initial color.

Question 1 : Display the image in gray scale levels.

Question 2 : Display your image in green (*night vision* like: to improve the result, change the brightness according to the current line - c.f. `gl_FragCoord`).

Question 3 : Display the object in green and change the brightness according to the depth.

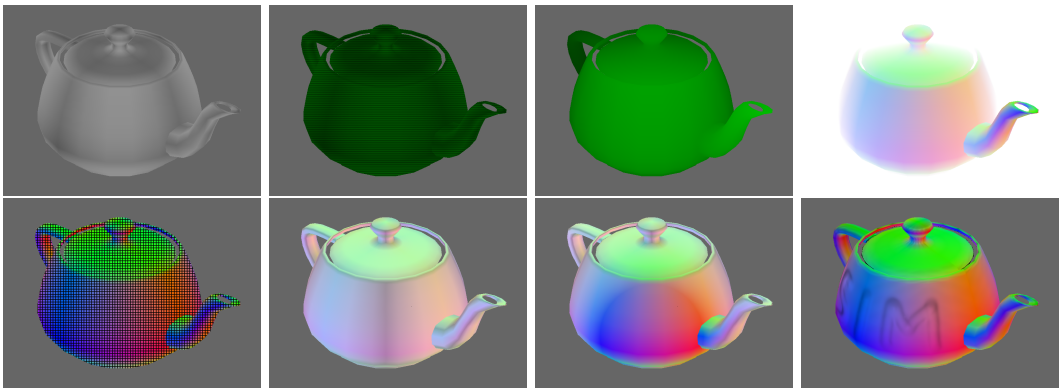
Question 4 : Add fog (change the background color to improve the result).

Question 5 : Display a checkered image of the object (on 6×6 squares separated by 2 pixels; leave background color or black between the squares).

Question 6 : Display a more colorless image.

Question 7 : Display a radial gradient from the color to the gray level scales (from the center to the boundary of the image).

Question 8 : Using *normalmap*, try to bump the teapot. The normal vectors are in the tangent coordinate space and every values are between 0 and 1 and should be converted between -1 and 1.



Bonus

Bonus

Parmi toutes les images, une est fausse. Laquelle ?

Among all these images, one of them is wrong. Which one?