TP N°9
La texture

Créer et afficher une texture :

Nous allons dessiner un carré et afficher dessus une image (texture). Les images texture doivent avoir des dimensions multiples de 2.
Voici les étapes à suivre pour la suite:

1- Créer un objet texture :

```
GLuint tab_texture[n] ;
glGentextutes(GLuint n,  GLuint *tab_texture) ;
```

Avec n, le nombre de textures générées, puis afin de spécifier la texture courante sur laquelle on applique les opérations qui suivent, on la rattache avec glBindTexture(GL_TEXTURE_2D, tab_texture).

2- charger la texture à utiliser :

L'image utilisée est au format « raw », qui est un tableau de pixels non compressé.

```
char image[largeur * hauteur  * 3];
FILE *f = fopen(FileName, "rb");
fread(image, largeur * hauteur  * 3, 1, f);
fclose(f);
```

3- Transférer la texture à la carte graphique :

```
glTexImage2D( target, level, components, w, h, border, format, type, *texels) ;
```

target: GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D.
level : 0 sans mip-mapping.
components : nombre d'éléments par texel.
w, h : largeur et hauteur.
border : bordure supplémentaire (toujours à zéro).
format : GL_RGB, GL_RGBA, …
type : type des éléments de la texture.
texels : l'image.

4- Définir le mode de filtrage:

Il existe des filtres pour indiquer comment un texel (pixel de la texture) doit être réduit ou agrandi afin de correspondre à un pixel :

```
glTexParameterf(GL_TEXTURE_2D, type, mode);
```

le type peut être : GL_TEXTURE_MAG_FILTER ou GL_TEXTURE_MIN_FILTER.
le mode peut être : GL_NEAREST ou GL_LINEAR (interpolation bilinéaire).
On peut voir l'effet de l'interpolation des texels lorsque la texture est de faible résolution et doit être étirée pour se positionner sur un objet plus grand.

5- Assigner les coordonnées de la texture, normalisées entre 0 et 1, aux sommets de l'objet dessiné.
(0,0) est le point en bas à gauche et (1,1) est le point en haut à droite de l'image.
Les coordonnées de texture seront transmises au vertex shader puis au fragment shader pour qu'elles soient interpolées pour tous les fragments.

Exemple :

- Afin de définir les coordonnées de texture, nous allons changer la structure *STRVertex* et y ajouter la texture :

```
struct STRVertex
{
   vec3 position;
   vec3 couleur;
   vec2 texture;
};
```

- Compléter le tableau *vertices[]* avec les coordonnées de texture pour chaque sommet.
- Spécifier l'attribut texture :

```
glEnableVertexAttribArray(2);
glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE, sizeof(STRVertex) , (void*) offsetof(STRVertex,texture) );
```

- Appeler la fonction *InitTexture()* dans le main puis récupérer la location de la variable uniforme:

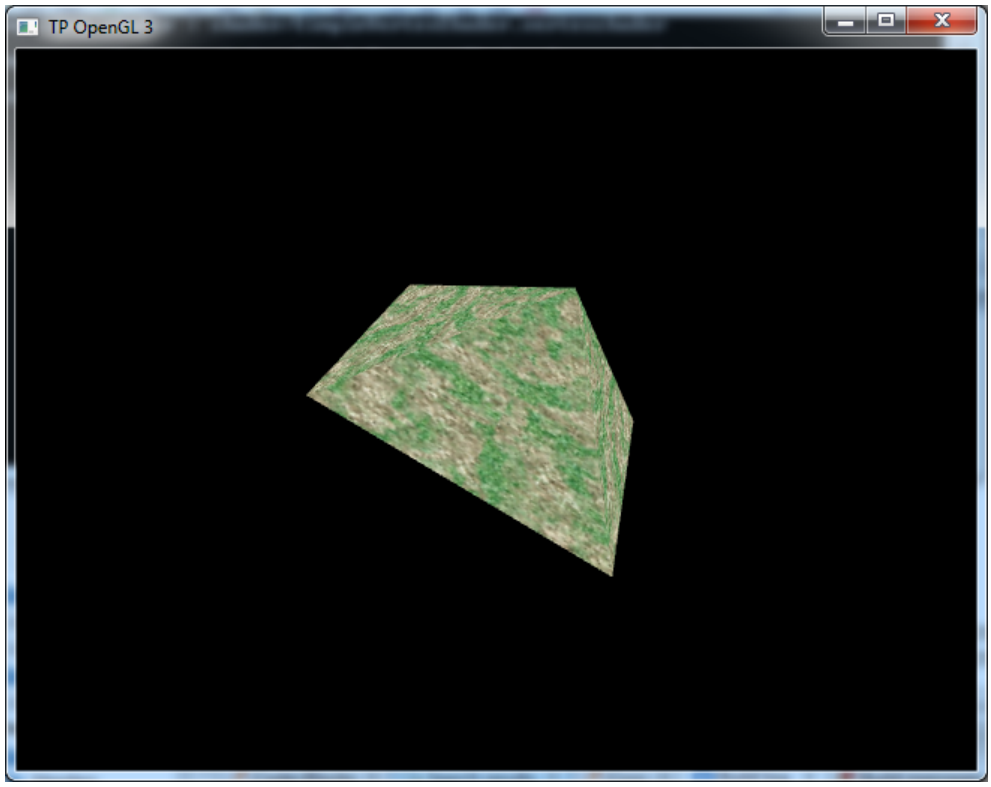
```
Gluint TextureID  = glGetUniformLocation(ShaderProgram, "ourTexture");

GLuint text;   // une seule texture
void InitTexture(void)
{
        char data[128*128*3];
        FILE *f = fopen("textures/herbe.raw", "rb");
        if(f)
        {
                fread(data, 128*128*3, 1, f);
                fclose(f);

                glGenTextures(1, &text);
                glBindTexture(GL_TEXTURE_2D, text);
                glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 128, 128,        0, GL_RGB, GL_UNSIGNED_BYTE, data);
                glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
                glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        }
}
```

- Ensuite dans la boucle de rendu et avant de dessiner, on binde la texture et on fixe la variable uniforme à 0 (le 0 représente the texture unit):

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, text);
glUniform1i(TextureID, 0);
```

- Le vertex shader va prendre ces coordonnées de texture et les passer au fragment shader. Le fragment shader va utiliser la fonction *texture()* pour échantillonner la texture qui sera placée dans la variable uniforme de type *sampler2D*:

```
in vec2 vTexture;
out vec4 color;
uniform sampler2D ourTexture;
void main()
{       color = texture(ourTexture, vTexture);        }
```



Le mode de bouclage:

Lorsqu'une coordonnée de texture n'est pas dans l'intervalle [0,1], nous avons deux possibilités : la même texture est répétée ou bien les valeurs des extrémités sont utilisées :

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,  mode);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T,  mode);
```

mode peut être: GL_REPEAT, GL_MIRRORED_REPEAT, GL_CLAMP_TO_EDGE ou GL_CLAMP_TO_BORDER.
Avec ce dernier mode, il faut spécifier la couleur de bordure :

```
glTexParameterfv(GL_TEXTURE_2D, GL_TEXTURE_BORDER_COLOR, borderColor);
```

Exercices :

- Plaquer la texture de damier (damier.raw) sur la base de la pyramide. Doubler puis tripler le nombre de carreaux sur chaque face sans modifier l'image.
- Appliquer des images différentes sur chaque facette.
- Récupérer les coordonnées de texture du modèle obj comme fait pour la position et la couleur et plaquer une texture.