Textures - TD 6

Plaçage de texture

On va travailler sur l'objet `sphere.obj`, il faut donc tout d'abord modifier `Viewer::init()` pour prendre le bon objet.

1. Ajouter en entrée du *Vertex Shader* une variable représentant les coordonnées de la position des textures `vtx_texcoords`, qui sera passée au *Fragment Shader* (comme `vtx_color`). Ajouter son initialisation dans `Mesh::draw()`. Attention au dernier paramètre de `glVertexAttribPointer()` qui représente la position du paramètre dans le fichier, on se décale de deux *Vector3f* (position et normale), et d'un *Vector4f* (couleur).

2. Ajouter à la classe `Viewer` un attribut `GLuint _texID0` qui correspond à **l'identifiant de texture** et l'initialiser dans `Viewer::init()` (juste avant `glEnable(GL_DEPTH_TEST)`).

   ```
   _texID0 = SOIL_load_OGL_texture(DATA_DIR "/textures/earth.jpg",
   SOIL_LOAD_AUTO, SOIL_CREATE_NEW_ID, SOIL_FLAG_INVERT_Y |
   SOIL_FLAG_MIPMAPS);
   ```

   Bon à savoir : *SOIL2* permet grâce à sa fonction `SOIL_load_OGL_texture()` de faire toutes les étapes nécessaires à la création d'un **identifiant de texture** en OpenGL automatiquement (présentées dans l'introduction du TD sur Moodle).

3. Ajouter au *Fragment Shader* une variable uniforme de type `sampler2D` qui représente la texture à utiliser, et remplacer la couleur de sortie de ce *Shader* en fonction.

   ```
   out_color = vec4(texture(<sampler2D_utilisé>, var_texcoords).rgb, 1);
   ```

4. Dans `Viewer::drawScene()`, activer la première **unité de texture** (0) avec `glActiveTexture(GL_TEXTURE0)`, faire le lien avec l'identifiant de texture mis en place au point 2 avec `glBindTexture(GL_TEXTURE_2D, _texID0)`, puis la transmettre au *Fragment Shader* grâce à la variable du point 3 avec `glUniform1i(_shader.getUniformLocation("tex0"), 0)` (tex0 correspondant au nom du *Sampler2D* dans le *Fragment Shader* et 0 correspondant à l'unité de texture à transmettre).

   Toujours dans `Viewer::drawScene()`, mais cette fois après avoir dessiné la scène, activer la même unité de texture avec `glActiveTexture(GL_TEXTURE0)`, et lui retirer le lien avec l'identifiant de texture avec `glBindTexture(GL_TEXTURE_2D, 0)`.

5. Il est alors possible de lancer le projet et de visualiser l'objet texturé : la Terre (woaw).

6. Ajouter la prise en compte de la lumière ajoutée au TD précédent. Pour cela, commenter (`//`) la sortie `out_color` que l'on vient de faire, on va partir de celle du TD précédent qui utilise `blinn()`.

```
out_color = vec4(blinn(normalize(n), normalize(v), l, vCol, sCol, s),
1);
```

On va seulement remplacer la couleur de l'objet par la couleur de la texture : `texture(<sampler2D_utilisé>, var_texcoords).rgb`.

7. Il est alors possible de lancer le projet et de visualiser la Terre éclairée par la lumière ajoutée et ombragée à l'opposée.

8. Comme précédement, ajouter deux indentifiants de textures et les initialiser pour `earth_clouds.jpg` et `earth_night`, ajouter deux variables uniformes de type `sampler2D` au *Fragment Shader*, et ajouter l'activation, le lien, la transmission au *Fragment Shader*, et le délien, des identifiants de textures, aux unités de textures 1 et 2.

   Il est alors possible (pour tester) d'afficher chaque texture individuellement en modifiant le *Sampler2D* utilisé par le *Fragment Shader* dans l'appel à `blinn()` dans la définition de `out_color`.

9. On va commencer par mélanger la texture de la Terre et la texture des nuages afin d'avoir une texture finale de la Terre avec nuages. Dans l'appel à `blinn()`, on utilise une couleur diffuse/ambiante. On va désormais la remplacer par la couleur renvoyée par la fonction `mix()` de GLSL. Celle-ci prend en paramètre la première couleur (texture 0 sans le `.rgb` car on veut garder le *alpha channel*), la deuxième couleur (pareil pour la texture 1), et le premier canal (`.x`) de la deuxième couleur. (Je suis désolé j'ai très pas compris pourquoi, mais en tous cas c'est la valeur qui représente l'importance de la deuxième texture sur la première. Donc en gros si t'as un nuage important (pixel blanc) tu affiches que le nuage, si tu as un nuage moyen (pixel gris) tu mélanges à 50-50, et si tu as aucun nuage (pixel noir) tu affiches la que la Terre.) On obtient un *vec4* il faut donc ne pas oublier de passer uniquement les coordonnées *x*, *y* et *z*.

10. Il est alors possible de lancer le projet et de visualiser la Terre illuminée, avec des nuages en plus.

11. De la même manière, mélanger cette fois la toisième texture et le mélange effectué précedémment, avec comme valeur d'importance le terme diffus.

12. Il est alors possible de lancer le projet et de visualiser la Terre illuminée, avec des nuages, et les lumières de la nuit dans les zones d'ombres en plus.

N.B.
Il semble important de noter que sur les images du sujet sur *Moodle*, j'ai l'impression qu'il n'y a pas la lumière spéculaire. Peut-être faut-il donc enlever la partie spéculaire en mettant sont terme à 0 (ou bien en commantant tout le calcul etc.).