Université Mohamed Khider-Biskra
Faculté des Sciences Exactes et Sciences de la Nature et de la Vie
Département d'informatique

Master 1 IVA **Module: SGPU**
Durée: une séance. **2019/2020**

TP N°05

But : manipuler des frames buffer objects (FBO).

Rappel :

Les frames buffer objects (FBO) ont été introduits pour faciliter le rendu dans des textures. Notons qu'OpenGL supporte aussi un concept de render buffer pour le calcul d'images. La distinction entre texture et image est qu'une texture peut être composée de plusieurs images qui ont toutes le même format. Une image est un seul tableau de pixels. Un FBO est une structure de données similaire au frame buffer. Cette structure est en quelque sorte un handle sur des textures de couleur, profondeur, etc. À la différence du frame buffer, les FBO sont indépendants du système de fenêtres. Avant de tenter d'utiliser les FBO, il est sage de vérifier si votre carte graphique les supporte avec la fonction **gluCheckExtension**.

Création de FBO :

void glGenFramebuffers(GLsizei n, GLuint *ids);

- **n** : Nombre de FBO à créer.
- **ids** : Noms (identificateurs) des FBO.

Cette fonction crée un (ou des) FBO. Pour qu'il soit possible de l'utiliser, il faut lui associer des textures.

Destruction de FBO :

void glDeleteFramebuffers(GLsizei n, GLuint *ids);

- **n** : Nombre de FBO à créer.
- **ids** : Noms (identificateurs) des FBO à détruire.

Structure d'un FBO : Un FBO est composé de plusieurs images (textures). Celles-ci doivent être créées indépendamment du FBO et attachées à celui-ci par la suite.

Les points d'attache d'un FBO sont les suivants :

- **GL_COLOR_ATTACHMENTi** : tampons images (i=0,1,..., GL_MAX_COLOR_ATTACHMENT-1).
- **GL_DEPTH_ATTACHMENT** : tampon de profondeur (nécessaire si on veut activer test de profondeur).
- **GL_STENCIL_ATTACHMENT** : tampon stencil.

Avant d'utiliser un FBO, il faut en faire le FBO courant.

Change le FBO courant.

void glBindFramebuffer(GLenum target, GLuint fb);

- **target** : **GL_FRAMEBUFFER** ;
- **fb** : Nom du FBO.

Lorsqu'on a terminé d'utiliser le FBO, rappeler cette fonction avec fb = 0.

Points d'attache

Avant qu'il soit possible d'utiliser un FBO, il faut lui associer des images. Dans le cas d'images 2D, on utilise la fonction suivante :

void glFramebufferTexture2D(GLenum target, GLenum attachment, GLenum textarget, GLuint texture, GLint level);

- **target : GL_FRAMEBUFFER;**
- **attachment** :Un des points d'attache (GL_COLOR_ATTACHMENTi, etc) ;
- **texture** : Nom de la texture;
- **textarget** : GL_TEXTURE_2D, etc;
- **level** : Pour mipmap (mettre 0 pour une image).

Cette fonction associe une texture à un des points d'attache du FBO courant.

Les images attachées à un FBO doivent répondre à certains critères (par exemple, être de taille non-nulle, au moins une image doit être attachée au FBO, etc.). Un FBO est dit complet lorsqu'on peut l'utiliser. La fonction suivante devrait être appelée avant l'utilisation.

Vérification de la validité d'un FBO :

GLenum glCheckFramebufferStatus(GLenum target);

- target : GL_TEXTURE_iD.

Cette fonction retourne la valeur **GL_FRAMEBUFFER_COMPLETE** si le FBO peut être utilisé.

Si on utilise un FBO pour le calcul d'image, le point d'attache 0 sera utilisé par défaut. On peut changer la destination avec la fonction suivante.
Indique à OpenGL dans quel point d'attache dessiner.

void glDrawBuffer(GLenum mode) ;

- **mode** : **GL_COLOR_ATTACHMENTi**.

Exemple :

```
GLuint framebuffer , texture ;
GLenum status ;
glGenFramebuffers (1 , &framebuffer ) ;
glBindFramebuffer (GL_FRAMEBUFFER_EXT, framebuffer ) ;
glGenTextures (1 , &texture) ;
glBindTexture (GL_TEXTURE_2D, texture ) ;
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR) ;
glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR) ;
glTexImage2D(GL_TEXTURE_2D, 0 , GL_RGBA8, W, H, 0 , GL_RGBA, GL_UNSIGNED_BYTE, NULL) ;
.
.
glFramebufferTexture2D (GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, tex ture , 0) ;
status = glCheckFramebufferStatus (GL_FRAMEBUFFER) ;
i f ( status != GL_FRAMEBUFFER_COMPLETE)
        / / Handle error here
/ / Your code to draw content to the FBO
/ / . . .
/ / Make the window the target
glBindFramebuffer (GL_FRAMEBUFFER, 0) ;
/ / Your code to use the contents of the FBO
/ / . . .
/ / Tear down the FBO and texture attachment
glDeleteTextures (1 , &texture ) ;
glDeleteFramebuffers (1 , &framebuffer ) ;
```

Le rendu en texture est une méthode spécifique de rendu d'un objet ou d'une scène 3D en deux passes. L'idée de base est simple, on fait un premier rendu de notre scène dans une texture, puis l'on fait un second rendu de cette texture directement à l'écran. Généralement, on plaque cette texture sur un quadrilatère faisant toute la taille du canvas OpenGL. Cette opération permet de mettre en place de nombreux effets en travaillant sur la texture générée.

Question :

Vous devez créer une scène 3D de votre choix, puis vous devez rendre cette scène dans un FBO (texture), puis afficher cette texture (qui contient la scène) sur écran (sur un quadrilatère).

Les étapes à suivre :

- Création d'une texture vierge : Il faut d'abord créer une texture RGB vierge. Cette texture sera remplie directement par les couleurs de sortie du premier fragment shader lors du rendu de votre scène.
- Mise en place d'un FrameBufferObject pour contenir la texture.

Première passe OpenGL : Génération de la texture (utiliser les shaders).

- Remplissage du FrameBufferObject.

Seconde passe OpenGL : Placage de la texture sur un quad

- Création d'un quad.
- Affichage de la scène sur ce quad (utiliser les shaders).