

GPGPU

CPE

5ETI IMI

1 But

Dans un objectif d'apprentissage, plusieurs techniques vous seront présentées. Les points importants d'OpenGL abordés sont :

1. les Frame Buffers Objects (FBO),
2. les Transform Feedbacks (TF),

2 Prise en main de l'environnement

2.1 Compilation

Question 1 Compilez le code, assurez-vous de voir un maillage texturé, vous pouvez bouger la caméra à la souris.

Question 2 Observez les fichiers du projet (or dossier external) afin de comprendre le fonctionnement du projet. Vous devez comprendre l'ensemble du code.

3 Frame Buffers Objects

On souhaite maintenant créer un FBO. Cet objet permet de ne pas afficher à l'écran mais à l'intérieur d'une autre *frame*. Une frame est composée pour chaque fragment : d'une couleur, d'une profondeur et d'un stencil (~ profondeur mais avec plus de contrôle). C'est un premier pas vers le GPGPU. Les FBO permettent notamment d'effectuer du post-processing sur la scène mais aussi de créer des "miroirs", d'afficher une autre image calculée dans la scène (ex. télévision) ou de créer un affichage de debug.

3.1 À l'initialisation

Pour créer le FBO et le stocker dans une texture, il faut :

- Créer un framebuffer : `glGenFramebuffers(...)`
- Utiliser ce framebuffer : `glBindFramebuffer(...)`
- Créer un buffer de texture : `glGenTextures(...)`
- Utiliser la texture : `glBindTexture(...)`
- Créer la texture vide : `glTexImage2D(...)` (utiliser des octets non signés)
- Configurer la texture :

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

- Attacher la texture au FBO : `glFramebufferTexture2D(...)`

Il faut ensuite créer un buffer de rendu contenant les informations de profondeur et de stencil:

- Créer un buffer de rendu: `glGenRenderbuffers(...)`
- Utiliser ce renderbuffer : `glBindRenderbuffer(...)`
- Allouer la place à ce buffer : `glRenderbufferStorage(...)`
- Lier le framebuffer courant et le renderbuffer (profondeur et stencil) :

```
glFramebufferRenderBuffer(...)
```

Il faut ensuite tester le bon déroulement `glCheckFramebufferStatus(GL_FRAMEBUFFER)` et remettre le framebuffer courant à l’affichage `glBindFramebuffer(GL_FRAMEBUFFER, 0)`

3.2 À l’affichage

Pour créer le FBO avec le bon contenu, il faut effectuer le rendu normalement, seulement il faut utiliser le framebuffer voulu. Pensez à nettoyer l’image (`glClear` et `glClearColor`). Dans certains cas, il peut être nécessaire de préciser à OpenGL les informations sur la fenêtre `glViewport(...)`

Question 3 Implémentez un FBO. Une fonction de création d’image à partir du FBO courant est proposée dans `glhelper.h`

Question 4 Utilisez la texture générée par le FBO pour créer un effet miroir sur un quad.

4 Transform Feedbacks

Les transform Feedbacks permettent de modifier les vertices contenus dans un buffer. Seul le vertex shader (et dans certains cas le geometry shader) sont utilisés. Les transform feedbacks ne servent pas à faire de l’affichage. Il est cependant possible d’utiliser les buffers modifiés pour de l’affichage. Les vertices ne sont pas obligatoirement représentatifs d’une géométrie et les TF sont un moyen de faire de GPGPU.

Pour effectuer un TF:

4.1 À l’initialisation

- Créer les shaders
- Créer un programme et lier les shaders
- ▲ IL FAUT PRÉCISER LES ÉLÉMENTS DE SORTIE AVANT DE LINKER LE PROGRAMME.

```
glTransformFeedbackVaryings(...)
```

- Linker le programme
- Créer et utiliser les VAO, VBO et EBO normalement (voir `mesh.cpp` pour plus d’information). Il faut créer un/des VBO pour le stocker l’information issue du TF.

4.2 À l’affichage

- Désactiver l’affichage : `glEnable(GL_RASTERIZER_DISCARD)`
- Préciser le programme de TF
- Préciser le VAO
- Préciser le VBO
- Préciser le buffer d’écriture du TF : `glBindBufferBase(...)`
- Faire la transformation :

```
glBeginTransformFeedback(...);  
glDrawArrays(...);  
glEndTransformFeedback();
```

- Attendre le buffer `glFlush(...)`
- Réactiver l’affichage `glDisable(GL_RASTERIZER_DISCARD)`

Attention, il faut décrire de nouveau le fonctionnement des buffers après utilisation des TF !

Pour lire le contenu d’un buffer, vous pouvez utiliser : `glGetBufferSubData(...)`. Dans le cas où un geometry shader est utilisé, le nombre de vertices peut changer. Il est alors nécessaire d’utiliser un *query objects*. Plus d’information sont disponibles : <https://open.gl/feedback>.

Question 5 Observer le code de tf, et comprendre pourquoi le maillage se déplace.