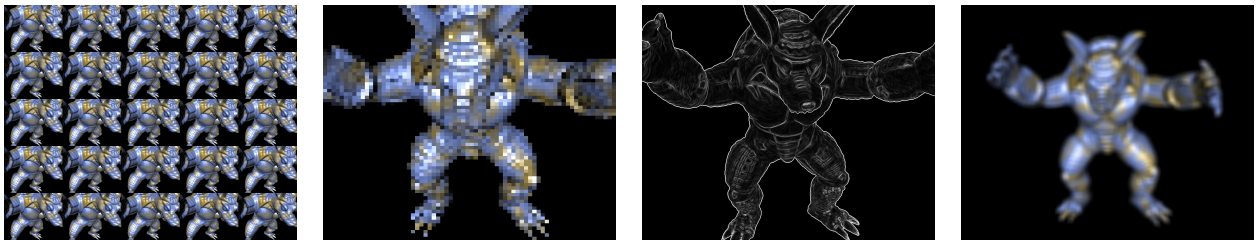
Modélisation et Synthèse d'Image
TP8 : FBO & Post-processing



Introduction

Dans ce TP, nous allons voir comment dessiner directement dans une texture pour faire du post-traitement et créer des effets spéciaux.

La base de code fournie utilise 2 passes de rendu. La première dessine un objet dans une texture (avec un premier shader). La seconde dessine un carré de la taille de l'écran (comme on l'a fait pour les fractales) sur lequel la texture est plaquée. Pour la récupérer, allez sur la page du cours :
http://romain.vergne.free.fr/blog/?page_id=228.

Pour l'utiliser, suivez la démarche habituelle :

- extrayez l'archive du TP dans le dossier de votre répertoire personnel dédié aux TP de 3D (`unzip TP8.zip -d ~/TP3D/`)
- accédez au dossier du TP (`cd ~/TP3D/TP8/`)
- créez un lien symbolique vers le dossier `external` (`ln -s ../external/`)
- créez un lien symbolique vers le dossier `models` (`ln -s ../models/`)
- créez un lien symbolique vers le dossier `textures` (`ln -s ../textures/`)
- créez un dossier pour la compilation (`mkdir build`)
- accédez à ce dossier (`cd build`)
- lancez `cmake` (`cmake ..`)
- lancez la compilation (`make`)
- exécutez (`./polytech_ricm4_tp8`)

Vous devriez avoir la hiérarchie de fichiers ci-contre.

```
TP3D
├── external
│   └── ...
├── models
│   └── ...
├── textures
│   └── ...
├── TP1
│   └── ...
├── TP2
│   └── ...
├── TP3
│   └── ...
├── TP3 bis
│   └── ...
├── TP4
│   └── ...
├── TP5
│   └── ...
├── TP6
│   └── ...
└── TP8
    ├── external [-> ../external/]
    ├── models [-> ../models/]
    ├── textures [-> ../textures/]
    ├── build
    │   └── ...
    ├── shader
    │   └── ...
    ├── src
    │   └── ...
    └── CMakeLists.txt
```

Observation du code

Observez attentivement le code fourni. 2 VAOs sont créés : un pour gérer la géométrie de l'objet, l'autre pour la géométrie du carré plaqué sur l'écran. On dispose aussi de 2 shaders (firs- et second-pass). Le premier est utilisé pour l'affichage de l'objet, avec les transformations classique (multiplication par les matrices MVP) et un Phong shading de base. Le second est utilisé lors de l'affichage du carré. La texture contenant le premier rendu lui est envoyé pour faire du post-traitement. A noter que par défaut, aucun post-traitement n'est réalisé. On ne visualise que la texture créée.

2 textures sont créées et attachées à un FBO (FrameBuffer Object - l'objet utilisé dans OpenGL pour que l'on puisse dessiner dans des textures). On va donc dessiner dans la texture ayant l'ID textureColorID (la seconde est utilisée seulement pour qu'OpenGL puisse faire le test de profondeur dans notre cas).

Dans la boucle de rendu, il y a maintenant 2 passes. Pour la première, le code est identique au TP sur les maillages indicés. Pour la seconde, le code est identique au TP sur les fractales. La seule différence est que l'on active le FBO pour la première : cela signifie qu'on dessine dans les textures qui lui sont attachées. Puis on désactive le FBO pour la seconde et on envoie au shader la texture dans laquelle on vient de dessiner.

Vous modifierez principalement le code de second-pass.frag durant ce TP pour faire du post-traitment sur la texture contenant le rendu. Les exercices qui suivent sont indépendant. Vous pouvez les faire dans le désordre.

Pixel-art

En jouant avec les coordonnées de texture dans second-pass.frag, essayez d'obtenir un effet pixélisé sur le rendu. Fonctions GLSL utiles : `floor` (partie entière d'un nombre/vecteur réel).

Multi-écran

Toujours avec les coordonnées de texture, obtenez un effet de répétition (voir image de la première page). Fonctions GLSL utiles : `fract` (partie fractionaire d'un nombre/vecteur réel) et/ou `mod` (modulo d'un réel/vecteur)

Discontinuités de couleur

Pour détecter les contours de l'objet, on peut simplement calculer un gradient, puis d'afficher sa norme. Pour calculer un gradient, il suffit de calculer 2 différences de couleurs : pixel de droite - pixel de gauche, puis pixel du haut - pixel du bas. Pour aller cherche un pixel voisin, il faut se déplacer d'un pixel pour la coordonnée. La taille d'un pixel dans l'espace des coordonnées de texture est égal à `vec2 ps = 1./textureSize(texSampler,0);`. Pour accéder à un pixel voisn (par exemple celui de droite, il suffit donc de faire `texture(texSampler,texcoord+vec2(1,0)*ps)`. Enfin, la norme d'un vecteur se calcule avec la fonction `length`.

Flou

Un flou est une simple moyenne des couleur sur un voisinage autour du pixel traité. Il faut donc faire une double boucle dans le shader. Les fonctions nécessaires sont uniquement celles décrites pour les discontinuités.

Variantes

Flou de profondeur

Adaptez le flou en fonction de la profondeur des points de la scènes depuis la caméra. Note : la profondeur est stockée dans le canal alpha de la texture.

Flou directionnel

Le flou directionnel est souvent utilisé pour reproduire du motion blur lorsque la caméra bouge. Pour cela, il suffit de flouter les pixel seulement dans une certaine direction. Il n'y a donc qu'une seule boucle ici.

Combinaison d'effets

Vous pouvez combiner les effets, en ajoutant les contours aux couleurs floutées ou en modifiant les coordonnées pour obtenir différent rendus. Essayez d'obtenir votre propres effets. Regardez la spécification GLSL pour regarder les fonctions mathématiques dont vous disposez. Vous pouvez aussi animer ces effets en envoyant une variable au shader et en l'utilisant.