Université de Limoges

master ISICG limoges

Informatique Graphique
Moteurs 3D temps réels
TP3 : Effets de Post-Processing

G.Gilet

19 octobre 2017

1 Utiliser un Frame Buffer Object(FBO)

Utiliser un FBO consiste à définir un objet OpenGL qui va être utilisé comme sortie du pipeline graphique. Le pipeline n'ecrit alors plus sur le Frame Buffer final mais directement dans une texture liée au FBO.
Sous GobLim, l'utilisation des FBO passe par la classe GPUFBO qui permet de définir un FBO et de l'utiliser simplement. Cette classe est considérée comme étant une ressource GPU. Commencez par déclarer une variable GPUFBO dans votre moteur et récuperez (lors de l'initialisation) un pointeur vers cette variable en utilisant le gestionnaire de ressource de la scène (cela vous permettra d'acceder à l'interface de votre ressource dans le menu). Pour cela, utilisez la commande :
$monFbo = scene->getResource<GPUFBO>("LeNomDeMonFBO")$
Appelez maintenant la fonction **createTexture2DAttachment(1024,1024)** de cette classe pour créer un FBO de résolution 1024 par 1024.
Dans la fonction **render** de votre moteur, activez ce FBO avant d'envoyer la géométrie, puis désactivez le. Finalement, affichez votre FBO en utilisant la fonction **display** de la classe GPUFBO. Constatez maintenant grâce à l'interface que votre FBO est bien rempli et essayez d'en changer la résolution

2 Un premier Effet

Nous allons maintenant créer un effet de flou. Dans le repertoire Effects de votre projet, vous trouverez une classe Blur, héritant de **EffectGL**. Cette classe initialise un vertex shader permettant d'afficher un quad couvrant l'ecran et un fragment shader basique assignant comme couleur finale à un pixel la couleur du texel correspondant dans le FBO donné en entrée.
Dans votre moteur, créez et declarez cet effet (en utilisant le gestionnaire de ressource de la scène) et appliquez le, dans votre fonction **render**, en lui transmettant le FBO défini précedemment (laissez pour l'instant la sortie out vide afin d'afficher directement le resultat de votre effet).

Modifiez le fragment shader de l'effet afin de créer un effet de flou de 8 par 8. Il faut pour cela assigner comme couleur finale au pixel la moyenne des 64 texels voisins.

3 Un flou amélioré

Nous avons ici appliqué un flou de manière brute, ce qui implique dans notre cas de faire 64 appels à la fonction texture pour chaque pixel. Il est possible d'optimiser cela en séparant notre flou en deux passes. Le principe est d'effectuer en premier un flou horizontal et d'ecrire le resultat dans un FBO intermediaire, puis d'effectuer un flou vertical à partir du FBO précédent. Cela ne fera plus que 16 appels a la fonction texture au lieu de 64.
Modifiez votre effet pour réaliser ce flou séparable. Il vous faudra pour cela créer un nouveau shader et utiliser la fonction **useProgramStage** pour changer au vol de fragment shader au sein du pipeline de l'effet.
Améliorez le flou obtenu en utilisant un flou de type gaussien. Le principe est de ne plus faire la simple moyenne des pixels voisins, mais de pondérer chaque pixel par une fonction gaussienne. Finalement, passez la taille du flou en paramètre a vos shaders et exposez ce paramètre dans l'interface de votre effet (en implémentant le fonction **displayInterface()**).

4 Un effet de Bloom

L'effet bloom consiste a afficher les objets avec un halo lumineux autour, afin de refléter les artefacts produits par les caméras. Afin de réaliser cet effet, effectuez le rendu de votre scène dans un FBO. A l'aide d'un shader, copiez ensuite dans un premier FBO les pixels dont l'intensité dépasse une valeur seuil (a définir en paramètre). Commencez par flouter ce buffer (en utilisant votre effet flou précédent) et combinez l'image d'entrée avec ce premier FBO.
Afin d'obtenir de meilleurs résultats, le principe est d'effectuer cette méthode plusieurs fois, avec des FBO de taille de plus en plus petite (en divisant la taille par 2 par exemple) et de recombiner lors de la passe finale tous les FBO ensemble.