TP1: Réflexions par Stencil buffer

Loïc Simon

December 12, 2014

L'objectif de ce TP est de détailler l'utilisation du stencil buffer. Deux applications sont envisagées:

- la création d'un trou dans un objet
- et la création d'un miroir plan et parfait.

Cette séance sera aussi l'occasion de préparer la séance suivante, ou le stencil buffer sera utilisé pour implanter la technique de "shadow volume" pour réaliser des ombres portées.

Pour réaliser les exercices suivants, vous devez cloner le répertoire git dédié. Voici la liste de commandes à réaliser en début de TP:

```
# lien vers resources tierses
cd
ln -s /home/public/simonl/Synthese/local_install .
# recuperation du repertoire de travail
cd ~/Bureau/ # ou tout autre repertoire de travail
git clone http://www.ecole.ensicaen.fr/~simonl/gits/CGILabs.git/
cd CGILabs
# test compilation et lancement
mkdir build
cd build
cmake ..
make
./Mirror/mirror
```

Au cours de l'avancement n'hésitez à pas utiliser git pour sauvegarder votre code. Voici une séquence typique de commandes:

```
git add mirror.cpp
git commit -m"petite avancée sur l'exo 1"
...
git commit -a -m"Exo 1= piece of cake"
git tag exo1
...
```

Dans ce TP vous travaillerez essentiellement sur le fichier `mirror.cpp` et de façon plus accessoire sur le fichier `mirror.f.glsl`. Dans son état original, l'exécutable créé, affiche une scène composée d'un singe, une théière, un mur, un plancher et un miroir plan. Il est possible de déplacer la caméra avec les flèches du clavier. Utilisez cette fonctionnalité pour observer la scène.

D'autre part, dans le code, des sections sont réservées pour les réponses aux exercices. Elles sont du types:

```
if(controls.exercise==i)
{
    /*!todo Instructions pour l'exercice i*/
}
```

Une telle section doit être remplie pour l'exercice `i` en suivant les instructions en commentaire. Certaines sections sont identiques pour plusieurs exercices, et certains exercices peuvent être concernés par plusieurs sections. *Respectez bien cette structure !*

Au départ le code est vide pour tous les exercices a part l'exercice 0 (qui correspond à la scène de base). Mais lorsque vous aurez répondu, vous pourrez naviguer entre les différents exercices en appuyant sur la touche 'e'. Vous pouvez aussi accéder a l'exercice 'i' en utilisant le clavier numérique.

1 Utilisation d'un Stencil Buffer

1.1 Qu'est ce qu'un stencil buffer

Le mot "stencil" en anglais signifie pochoir. Un stencil buffer est un buffer disponible en lecture et en écriture, qui permet de réaliser la même fonction qu'un pochoir, à savoir de dessiner uniquement une partie des pixels. Un exemple simpliste d'application de cette idée permet de simuler le rendu d'un objet troué (cf exercice 1).

Ce buffer ressemble au buffer de profondeur (le Z-buffer). C'est une zone mémoire, qui contient une donnée par pixel, appelée la valeur stencil du pixel. Cette donnée est un entier non signé. Nous supposons dans la suite que sa taille est de 8 bits par pixel (cela dépend de la carte graphique).

Rappelons le fonctionnement des tests de profondeur : avec ce mécanisme, la valeur d'un pixel qui normalement est une couleur devient un couple (c, d) où c'est une couleur et d un réel correspondant à la profondeur du pixel. Chaque pixel a une valeur courante (c, d). La couleur c est stockée dans le buffer des couleurs et d se trouve dans le Z-buffer. Lorsqu'on calcule une nouvelle valeur (c', d') pour un pixel, on compare d à d' (test de profondeur). Suivant le résultat, soit on remplace la valeur courante du pixel par sa nouvelle valeur (c' , d'), soit on conserve l'ancienne valeur.

En introduisant le stencil, on ajoute aux données du pixel une donnée supplémentaire dite valeur stencil du pixel. La valeur stencil courante d'un pixel se trouve dans le buffer stencil. Il existe aussi un test stencil : il consiste à

comparer une valeur stencil à une valeur de référence. Si le test échoue, le buffer de couleur et le Z-buffer resteront inchangés. S'il réussit, ces deux buffers peuvent être l'objet d'une mise à jour, suivant les méthodes habituelles. Dans tous les cas, on peut demander la modification du stencil suivant une méthode prédéfinie, parmi 6 méthodes proposées.

Nous verrons dans ce TP les applications de ce mécanisme. Sans plus tarder, nous allons voir les étapes nécessaires à l'utilisation de ce buffer.

1.2 Utiliser un stencil buffer: api OpenGL et GLFW3

Voici la liste des fonctions utiles pour gérer un buffer de stencil:

- Réserver la mémoire nécessaire:

```
glfwWindowHint(GLFW_STENCIL_BITS, 8);
```

- autoriser/interdire le test de stencil

```
glEnable(GL_STENCIL_TEST);
glDisable(GL_STENCIL_TEST);
```

- réinitialiser a une valeur donnée

```
glClearStencil( 0x0 );
glClear(GL_STENCIL_BUFFER_BIT);
```

- Choisir la fonction de test

```
glStencilFunc(function, ref, mask);
```

  Dans cette instruction, `function` peut être choisie parmi les options suivantes: `GL_NEVER`, `GL_ALWAYS`, `GL_LESS`, `GL_LEQUAL`, `GL_EQUAL`, `GL_GEQUAL`, `GL_GREATER`, `GL_NOTEQUAL`. Attention le test est effectué entre `ref` et le valeur de stencil du pixel et non l'inverse (cela a une importance pour les test d'inégalité!). La variable `mask` est un masque binaire permettant de choisir quels bits de la valeur de stencil seront utilisés dans le test.

- Choisir les actions a prendre pour mettre a jour la valeur du stencil suivant 3 configurations

```
glStencilOp(ifStencilFails, ifStencilPassesButZfails, ifBothPass);
```

  Les trois configurations correspondent au cas où:

  - le test de stencil échoue (`ifStencilFails`),
  - le test de stencil réussi mais le test de profondeur échoue (`ifStencilPassesButZfails`)
  - les deux tests réussissent (`ifBothPass`)

Les actions a prendre sont indiquées par le contenu de chacune des variables `ifStencilFails`, `ifStencilPassesButZfails`, `ifBothPass` . Elles peuvent être choisies parmi: `GL_KEEP`, `GL_ZERO`, `GL_INCR`, `GL_DECR`, `GL_REPLACE`, `GL_INVERT`. L'action `GL_REPLACE`, remplace la valeur courante par celle utilisée dans le test `ref`.

NB: Les actions définies précédemment constituent l'unique moyen d'écrire dans le buffer de stencil. Dans certains cas, on veut écrire dans le stencil, mais l'écriture dans le buffer de couleur ne doit pas dépendre de la valeur de stencil. Dans cette situation, il faut quand même activer le test de stencil mais utiliser un test trivial (e.g `GL_ALWAYS`).

- Contrôler quels bits seront mis à jours dans les 3 cas:

  ```
  glStencilMask (maskUint)
  ```

  NB: Cette dernière fonction est à comparer aux fonctions similaire pour les buffers de couleurs et de profondeur:

  ```
  glColorMask (redBoolean, greenBoolean, blueBoolean, alphaBoolean)
  glDepthMask (depthBoolean)
  ```

  Ces dernières fonctions seront utiles pour les exercices.

1.3 Exercice 1: Mur troué

Dans la fonction `draw`, remplissez la section délimitée par `if(controls.exercise==1)`, pour dessiner la même scène que celle de base, mais ou le miroir sert a créer un trou dans le mur. Pour cela vous suivrez les instructions en commentaire en début de la section concernée.

Q1. Expliquez ce que vous avez fait pour répondre a cet exercice. Votre réponse doit comporter des étapes conceptuelles et les instructions OpenGL permettant d'achever ces étapes. N'hésitez pas a illustrer vos propos avec des dessins.

2 Création d'un miroir parfait plan

Dans cette section, vous allez suivre des exercices courts, décrivant les différentes étapes nécessaires pour simuler la réflexion de la scène dans un miroir plan. Chaque exercice s'attache a une étape atomique de cette technique.

2.1 Exercice 2: réflexions

Dans la fonction `drawReflectionOfAllObjects` dessinez la réflexion de tous les objets (autres que le miroir) par rapport au plan du miroir (i.e. le plan $y = 0$ dans le repère objet du miroir). Dans la suite on fera la distinction entre objets **réels** et les objets **virtuels** qui correspondent a leur reflets.

Q2. Expliquez comment vous avez réalisé la symétrie requise, du point de vue mathématique et du point de vue des instructions OpenGL associées.

2.2 Exercice 3: quels objets doivent être reflétés ?

Le miroir ne reflète que la moitié de l'espace 3D. Donc certains objets ne doivent pas être reflétés. Dans la pipeline fixe, un plan de clipping pouvait être introduit via la fonction `glClipPlane`. Cette fonctionnalité devenant obsolète dans la pipeline programmable, nous allons l'implanter dans le fragment shader via la directive `discard`.

Dans cet exercice, vous devrez à la fois éditer le fichier C++ et le fragment shader `mirror.f.glsl`.

Q3. Expliquez le rôle d'un plan de clipping en général.

Q4. Dans notre cas, où le plan doit-il être placé pour éliminer les bons objets? Dans quelle orientation?

Q5. Expliquez la stratégie qui a été choisie pour définir le plan de clipping, notamment dans quel repère?

2.3 Exercice 4: Éliminer les objets réels dans le champs du miroir

Le miroir doit masquer les objets réels qui sont derrière lui. Ceci peut se réaliser aisément en dessinant le miroir dans le z-buffer uniquement. Suivez les instructions.

Q6. Expliquez pourquoi il faut dessiner les objets virtuels avant de dessiner le miroir dans le z-buffer.

2.4 Exercice 5: Éliminer les objets virtuels hors du champs du miroir

Le miroir a un champs de vue limité. Donc les objets virtuels doivent être découpés pour éliminer les parties qui sortent du champs du miroir. Vous réaliserez cette opération en utilisant le stencil buffer.

Q7. Expliquez comment est construit le champs de vue du miroir.

Q8. Pourquoi ne pas avoir utilisé la même technique (basée sur un test de stencil) dans l'exercice 4 pour décider quelles parties des objets réels doivent être dessinées?

2.5 Exercice 6: Dernières petites touches

Q9. Placez la caméra derrière le mur. Vous devriez observer les reflets du singe au lieu du mur, ce qui est évidemment une anomalie. Ce problème est du au fait que le z-buffer est erroné. Expliquez pourquoi et proposez une correction que vous implanterez.

Q10. Il reste encore une incohérence dans le dessin des objets virtuels. Saurez vous la repérer?

2.6 Exercice 7: Utilisation de plans de clipping

Q11. Expliquez une alternative à l'utilisation du stencil buffer, basée sur l'utilisation de plusieurs plans de clipping. Illustrez par des dessins.

Si vous êtes rapide, **en bonus** vous pouvez implanter cette technique.

2.7 Exercice 8: Miroirs parallèles

Q12. Proposer une extension de cette suite d'exercice pour gérer des miroirs se reflétant l'un dans l'autre (toujours en utilisant le stencil buffer).

Si vous êtes rapide, **en bonus** vous pouvez implanter cette technique.

3 Compte-rendu du TP

Bon courage!