TP OpenGL 5
Introduction aux textures

Licence Informatique 3ème année

Année 2021-2022

1 Introduction

L'utilisation de textures est devenue incontournable en synthèse d'image. Tous les domaines d'application de la synthèse 3D y recourent : les jeux, le cinéma, les effets spéciaux, les films d'animation, les publicités, la réalité virtuelle etc. ...

Le principe de l'application de texture est très simple : il consiste à « plaquer » une image sur un modèle en trois dimensions. Par exemple on peut plaquer l'image d'une tapisserie sur le mur d'une scène 3D ou bien celle d'un tableau. Mais on peut également photographier la façade d'une maison et l'utiliser pour représenter cette façade en 3D par une seule facette rectangulaire sur laquelle on aura collé l'image de la façade. L'intérêt est que la modélisation est grandement simplifiée, que la façade a l'air « réelle » et que l'animation est plus rapide qu'en ayant modélisé cette façade avec un grand nombre de triangles.

Bien que le principe des textures soit simple, leur fonctionnement est complexe et même leur utilisation en OpenGL est délicate. Bien que dans le cas général ce sont des textures 2D qui sont utilisées, il est possible d'utiliser également des textures 1D ou 3D. Il est nécessaire dans tous les cas de faire correspondre les points de la texture (appelés texels) à la surface sur laquelle elle sera appliquée. Il faut également choisir comment la texture se comportera avec la couleur existante de la surface. Et enfin, il est nécessaire de définir comment se comportera la texture lorsque celle-ci sera plus grande que la surface ou plus petite que la surface. Ces étapes seront obligatoires mais il existe également de nombreux autres paramètres qui doivent ou peuvent être réglés pour l'utilisation des textures.

Dans ce TP, nous nous restreindrons aux textures 2D et à seulement une partie des paramétrages et des utilisations possible des textures. Le but est d'acquérir simplement les bases de l'utilisation du placage de texture en OpenGL.

2 Mise en route

Vous allez récupérer un programme de base n'utilisant pas de texture et le modifier étape par étape pour incorporer des textures :

- Récupérez les fichier `TP6.zip` qui est associé à cet enoncé et décompressez le dans le dossier consacré à vos TP d'informatique graphique ; Vous obtenez alors un dossier nommé `TP6` qui contient les sources que vous allez utiliser ;
- compilez et exécutez le programme :
  - la touche espace active ou désactive l'animation ;
  - un glisser à l'aide du bouton gauche de la souris permet de zoomer ;
  - les touches échappement et 'q' quittent l'application ;
- ouvrez le fichier `types.h` et étudiez son contenu : vous remarquerez une structure Image qui sera utilisée pour stocker l'image de la texture ; une fonction `charger_textures` vide pour l'instant a été prévue dans le fichier `texture.c`.

La première chose à faire est d'activer le mode placage de texture :

- dans la fonction InitGL, après l'appel à charger_textures, ajouter la commande **glEnable(GL_TEXTURE_2D)**. Rien de particulier ne se produit suite à cette activation, puisque aucune texture n'a été utilisée à ce stade.

2.1 Chargement d'une première image

La première image que nous allons utiliser pour créer une texture est un damier. Au lieu de charger cette image à partir d'un fichier, nous allons la créer ; ceci est effectué par la fonction `fabriquer_damier`

qui se trouve dans le fichier `damier.c`.

Cette fonction prend pour paramètre un pointeur vers une structure image et la taille du damier que vous voulez créer. **Attention : Les textures doivent toujours avoir une taille qui est une puissance de 2**.

Dans le fichier `texture.c` :

— déclarez une variable globale image1 de type Image (une variable locale ne convient pas car l'image ne doit pas disparaître en sortant de la fonction) ;

— fabriquer ensuite un damier de taille 64 dans image1, en appelant la fonction `fabriquer_damier` dans `charger_textures` [1].

— compilez et vérifiez que l'application fonctionne toujours.

2.2 Création de la première texture

À chaque texture sera assigné un numéro, qui permettra de les identifier. Pour les stocker :

— créez, dans le fichier `texture.c`, une variable globale `texture` de type tableau de `GLuint` qui sera de taille 1 pour l'instant car une seule texture va être créée.

Dans la fonction `charger_textures`, suite à la création du damier, la création de la texture nécessite plusieurs étapes en commençant par la création d'objets de texture permettant de stocker toutes les informations nécessaires à chaque texture :

1. La création des numéros des objets de texture s'effectue par un appel à

   glGenTextures(GLsizei n, GLuint *numeros_textures)

   le premier paramètre indique le nombre de numéros de texture à créer et le second le tableau dans lequel ils seront stockés.

2. L'activation de la texture est effectuée par un appel à

   glBindTexture(GL_TEXTURE_2D, GLuint numéro_texture)

   le premier paramètre indique que c'est une texture 2D et le second le numéro de la texture à sélectionner. Ce numéro est celui retourné par la fonction `glGenTextures` qui aura été stocké, à ce stade, dans la variable `texture[0]`. Suite à cet appel, la texture sélectionnées devient la texture courante et toutes les instructions suivantes s'appliquant sur une texture ou nécessitant une texture, l'utiliseront, jusqu'à ce qu'une nouvelle texture soit sélectionnée ;

3. Le réglage des paramètres de la texture est effectué grâce à la fonction **glTexParameter*** qui prend 3 paramètres : `GL_TEXTURE_2D` pour spécifier le type de texture, puis le paramètre à fixer et enfin la valeur du paramètre :

   glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S, GL_REPEAT) ;
   glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T, GL_REPEAT) ;
   glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,
   GL_NEAREST) ;
   glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,
   GL_NEAREST) ;

   Les 2 premiers appels spécifient la manière dont la texture est répétée si celle-ci est plaquée de manière à être plus petite que la surface 3D (voir figure 1). Les 2 autres spécifient la manière dont les couleurs seront filtrées lorsque plusieurs texels (les pixels de l'image composant la texture) correspondront à un seul pixel écran et vice-versa (ce point ne sera pas davantage détaillé ici).

4. Avant d'associer notre image de damier à l'objet texture il faut spécifier la manière dont les pixels sont stockés dans notre image par :

   glPixelStorei(GL_UNPACK_ALIGNMENT, 1) ;

   Sans entrer dans les détails, cette instruction spécifie que les valeurs des pixels de chaque ligne de l'image représentant la texture sont alignés sur 1 octet ;

5. L'association de l'image à l'objet texture est obtenue par la fonction

```
glTexImage2D(GL_TEXTURE_2D ,GLint niveau, GL_RGBA,
             GLsizei largeur, GLsizei hauteur, GLint bordure, GL_RGBA,
             GL_UNSIGNED_BYTE, const GLvoid *pixels_image)
```

1. Cette fonction est appelée dans la fonction `InitGL` du fichier `cube.c`

Explication des paramètres :

- `GL_TEXTURE_2D` : type de texture ;
- `GLint niveau` : niveau de résolution de la texture, le **niveau 0** est la résolution maximale c'est-à-dire celle de l'image, et les niveaux supérieurs représentent la même image à des résolutions de plus en plus petites (utilisés pour le mipmapping, qui ne sera pas vu ici) ;
- `GL_RGBA` : représentation des texels de la **texture** en mémoire sous forme d'un quadruplet rouge, vert, bleu et transparence ;
- `GLsizei largeur` : largeur de l'image (en nombre de pixels) ;
- `GLsizei hauteur` : hauteur de l'image (en nombre de pixels) ;
- `GLint bordure` : largeur de la bordure de la texture (on supposera toujours **0**) ;
- `GL_RGBA` : représentation des pixels de l'**image** sous forme d'un quadruplet rouge, vert, bleu et transparence ;
- `GL_UNSIGNED_BYTE` : type de données de chacune des composantes des pixels (rouge, vert, ...) - ici un octet par canal de couleur ;
- `const GLvoid *pixels_image` : pointeur vers les pixels de l'image - ici le membre `donnees` de l'image considérée.

6. Enfin il faut spécifier la fonction d'application de la texture avec

   glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE)

   La fonction est placée à GL_REPLACE qui indique à la texture de remplacer la couleur de la surface 3D.

Arrivés à ce niveau, **compilez l'application** pour vérifier qu'il n'y a pas d'erreur. Si vous exécutez le programme, vous remarquerez que les couleurs des faces du cubes ont disparu mais que la texture n'apparaît pas : c'est parce qu'il y a à présent un texture activée (*cf* l'appel la fonction `glBindTexture`), mais aucune information sur la manière de la positionner sur l'objet.

2.3 Positionnement de la texture

Pour positionner la texture sur une face du cube il faut faire coïncider les coins de la texture avec les coins de cette face. Pour cela, la fonction **glTexCoord2f(GLfloat coord1,GLfloat coord2)** doit être appelée avant chaque appel à `glVertex3f(...)`.

Le repère d'une texture, dont les axes sont notés Os et Ot (ou parfois u et v), est toujours tel que le **coin bas gauche est l'origine et donc a pour coordonnées (0,0), le coin bas droit (1,0), le coin haut gauche (0,1) et le coin haut droit (1,1)** (voir le centre de la figure 1).

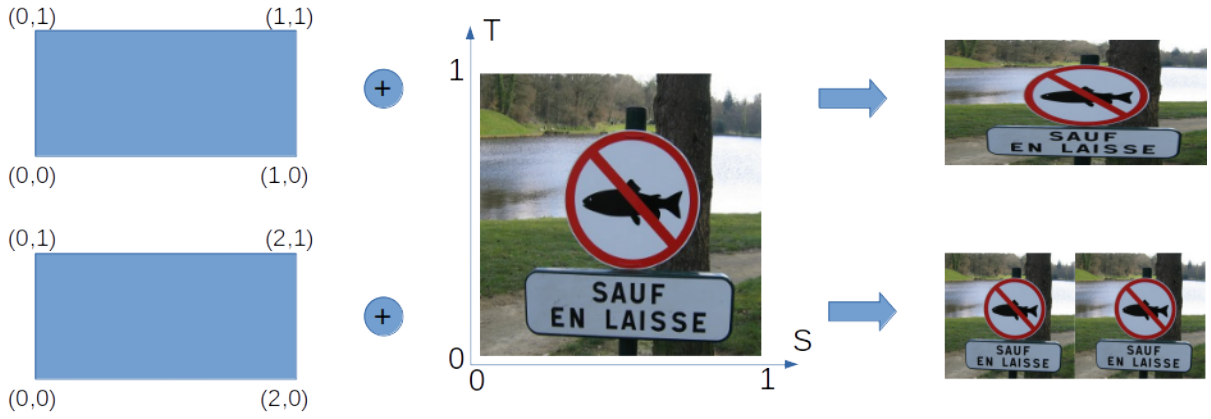

FIGURE 1 – Aperçu de l'effet de l'utilisation des coordonnées de texture. La texture est définie dans son repère $[0,1] \times [0,1]$ (au centre). A gauche en haut, on fait coïncider les coins de la facette sur laquelle appliquer la texture avec les coins de l'image. L'image est donc déformée pour prendre toute la place sur la facette (en haut à droite). An bas à gauche, la coordonnée s des point de droite de la facette ont pour coordonnées 2.0. On a fixé précédemment les mode d'application à `GL_REPEAT` via la fonction `glTexParameteri()`, ce qui signifie qu'on souhaite que la texture emplisse ici 2 fois la facette horizontalement (en bas à droite).

Ainsi le dessin de la face avant du cube dans la fonction d'affichage devient :

```
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f); /* bas gauche */
glTexCoord2f(1.0f, 0.0f); glVertex3f( 1.0f, -1.0f, 1.0f); /* bas droit */
```

```
glTexCoord2f(1.0f, 1.0f); glVertex3f( 1.0f, 1.0f, 1.0f); /* haut droit */
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f); /* haut gauche */
```

— De même, faites coïncider les coins de la texture avec les coins des 5 autres faces.
— compilez et testez ... ça marche si vous n'avez pas fait d'erreur !

2.4 Chargement et création d'une deuxième texture

Cette fois-ci la texture va être lue à partir d'une image bmp **en 24bits et de taille en puissance de 2** (64x128, 256x128, 256x256, ...).

1. Pour se faire, vous utiliserez la fonction `LireBmp` qui se trouve dans le fichier `bmp.c`. Celle-ci prend en paramètre le nom du fichier à lire, un pointeur vers l'image à créer et une valeur de 0 à 255 spécifiant le coefficient alpha qui sera associé à cette image.
2. modifiez votre code pour charger une deuxième texture après celle du damier, à partir du fichier `opengl.bmp` : attention vous aurez besoin de 2 numéros d'objets de texture et d'une deuxième image ! Surtout laissez **1 seul appel à glGenTextures** en spécifiant le nombre de numéros dont vous avez besoin.
3. au final, si vous arrivez à lire OpenGL sur toutes les faces, c'est que vous avez a priori bien fait correspondre les coins des textures et des faces du cube ... sinon corrigez.

Questions

1. Dans la fonction d'affichage positionnez l'appel
   glBindTexture(GL_TEXTURE_2D, texture[0]) ;
   juste avant `glBegin(GL_QUADS)`. Que ce passe-t-il ?
   Maintenant, mettez-le juste après. Que ce passe-t-il et qu'en déduisez-vous ?
2. Modifiez votre code pour afficher Opengl sur 3 faces et le damier sur 3 autres.
3. Ajouter une troisième et dernière texture lue dans l'un des fichiers `panneau01.bmp` ou `panneau02.bmp`.
4. Modifiez à nouveau votre code pour afficher le damier, OpenGL et le panneau sur 2 faces chacun.

2.5 Répéter ou resserrer des textures

Jusqu'à présent, nous avons positionné les textures de manière à ce qu'elles coïncident exactement avec les faces du cube. Mais il est possible d'assigner des coordonnées de textures en dehors de (0,1) à l'aide de la fonction `glTexCoord2f` :

— Choisissez une face du cube sur laquelle est plaquée l'image du panneau et modifiez les coordonnées de placage de la texture en remplaçant les 1.0 par des 2.0. Regardez ce qui se passe et essayez différentes valeurs en largeur et en hauteur.
— Ensuite, modifiez la mise en place du paramètre `GL_TEXTURE_WRAP_S` de la texture du panneau (dans l'appel à la fonction `glTexParameteri`) de sorte que la constante `GL_CLAMP` lui soit affectée à la place de `GL_REPEAT`. Regardez ce qui se passe alors ...
  Essayez de même avec le paramètre `GL_TEXTURE_WRAP_T`.
  Qu'en déduisez-vous ?

3 Exercice d'application

1. Reprenez le code de la fonction cylindre que vous avez écrite au TP 3 et ajoutez y ce qu'il faut pour que l'image du damier soit appliquée en une seule fois sur toute sa surface. Vous remplacerez le dessin du cube par le cylindre ;
2. complétez votre application pour que l'appui sur la touche `c` permette de passer l'affichage du cube vers le cylindre ou du cylindre vers le cube, selon l'objet actuellement affiché ;
3. modifiez votre application de telle sorte que l'appui sur la touche *flêche gauche* permette de changer la texture du cylindre en prenant la texture précédente dans la liste des textures chargées, tandis que l'appui sur la touche *flêche droite* change pour la texture suivante. Le passage d'une texture à l'autre sera géré de manière cyclique. Il ne sera appliqué que si le cylindre est affiché.

4 Exercice optionnel

Reprendre l'exercice précédent avec un modèle de cône, aligné sur l'axe $0z$ et dont l'extrémité se trouvera à l'origine. L'utilisation de la touche `c` affichera cycliquement le cube, le cylindre et le cône. Le changement de texture s'appliquera également au cône et cette dernière sera appliquée de telle sorte que le milieu de l'image se trouve sur la pointe du cône.