# Synthèse d'images : TD 2

Textures

L'objet de ce TD est de fabriquer une petite application interactive permettant de se déplacer dans un paysage 3D, tout en découvrant des techniques propres à l'utilisation de textures. Un squelette de code comportant l'initialisation de la visualisation, le code de chargement de texture à partir de fichiers TIFF, et la gestion de l'interaction (flèches, pavé numérique), sont fournis. On prendra tout au long de l'exercice comme échelle indicative 1 unité opengl = 1 mètre.

## Exercice 2.1 *Manipulations élémentaires*

Dans cette partie, nous allons dessiner le sol du paysage. Une texture d'herbe `grass.tiff` est appliquée à un quad dessiné en $z = 0$ pou représenter le sol. L'application de la texture nécessite l'activation d'un état OpenGL, grâce à par un appel à la fonction `glEnable(GL_TEXTURE_2D)`.

**1)** Repérez dans le code l'emplacement de la création de la texture, de chargement OpenGL de la texture courante, et de la définition de la façon d'apposer la texture sur le quad. Modifiez la taille du quad de sorte de créer l'illusion d'un sol plat. Quel problème esthétique constatez-vous?

**2)** La texture `grass.tiff` a été conçue pour être répétable (juxtaposition lisse). Modifiez les coordonnées de textures pour permettre la répétition de la texture par carreaux de 3 ou 4 mètres, en prenant des coordonnées de texture sortant de l'intervalle $[0, 1]$. Comment passe-t-on des paramètres à OpenGL au sujet des textures, notamment pour permettre la répétition?

## Exercice 2.2 *Filtrage*

Pour améliorer la qualité des textures une fois plaquée, ou plus généralement la qualité de l'image rendue, OpenGL permet de paramétrer plusieurs opérations.

**1)** Quel problème esthétique constatez-vous sur votre sol plaqué? Affectez la touche 'f' pour changer la méthode d'interpolation (`GL_LINEAR` ou `GL_NEAREST`) utilisée. Observez et comprenez la différence. A quelle étape du rendu cette opération de filtrage intervient-elle ?

**2)** Dans de très nombreux cas, la qualité des textures joue un rôle déterminant dans la qualité du rendu de la scène finale. Or, plus la taille de ces textures va être élevée, plus coûteux va être leur accès. D'autre part, une texture plaquée sur un polygone lointain n'a pas besoin d'une résolution importante. La technique de *mipmapping* permet de prendre ce facteur en compte en définissant plusieurs versions d'une même texture à des résolutions différentes. OpenGL déterminera automatiquement la meilleure résolution de texture à appliquer. Modifiez la fonction `initTexture()` pour spécifier vous même les niveaux de détail, en utilisant pour le niveau X la texture `mipmapX.tiff` à la place de la texture d'herbe. En plus de `GL_LINEAR`, `GL_NEAREST`, affectez à la touche 'f' les autres modes de filtrage propres au mipmapping et observez les différences: `GL_NEAREST_MIPMAP_LINEAR`, `GL_NEAREST_MIPMAP_NEAREST`, `GL_LINEAR_MIPMAP_LINEAR`, `GL_LINEAR_MIPMAP_NEAREST`.

Quel est le paramètre permettant la meilleure qualité de transition et de rendu?

**3)** Pour obtenir l'effet de la question précédente sur votre texture d'herbe, vous pourriez créer des versions filtrées avec un logiciel de dessin. Heureusement, GLU fournit une fonction pour créer et envoyer à OpenGL une famille de mipmaps à partir de la texture haute résolution. Remplacez vos appels à `glTexture2D` par un appel à `gluBuild2DMipmaps`.

**Exercice 2.3** *Plusieurs textures*
Le TP n'a jusqu'à présent utilisé qu'une seule texture. Il est bien sûr possible avec OpenGL d'en charger plusieurs, chacune pouvant retenir ses propres paramètres de filtrage, bordure, etc. Pour spécifier quelle texture est active pour le rendu et l'affectation de paramètres, OpenGL propose de créer un identifiant par texture manipulée avec `glGenTextures`, rendu actif par `glBindTexture()`.

**1)** Réécrivez la fonction `initTexture()` pour qu'elle charge une image OpenGL à partir d'un fichier, dans une entrée des tableaux de `Texture` proposés. Chargez une deuxième texture `crate.tiff` et changez la texture de rendu du sol parmi les 2 textures, à l'appui d'une touche.

**2)** Appliquez la nouvelle texture chargée aux 5 facettes visibles d'un cube posé au sol, de manière à ce que l'orientation soit correcte. *Assurez-vous que les paramètres de texture modifiés par la touche 'f' s'appliquent à chaque texture utilisée.*

**Exercice 2.4** *Billboards et alpha-blending*

La technique du *billboarding* consiste à remplacer des objets de géométrie complexe par un ou plusieurs imposteurs plats et semi-transparents texturés avec une image de l'objet. Cette technique est largement utilisée dans les jeux vidéos pour rendre des arbres rapidement.

**1)** Plaquez `tree.tiff` sur un quad vertical visible. Ouvrez également ce fichier dans un viewer d'images. Obtenez-vous l'effet recherché?

**2)** La semi-transparence est encodée dans une quatrième coordonnée `alpha`, qui s'ajoute aux trois couleurs encodées pour un texel, et définit son opacité. Sa prise en compte est possible nativement dans OpenGL mais nécessite d'activer explicitement le *blending* et de positionner les états OpenGL permettant de contrôler le mélange des couleurs de la texture appliquée avec celles déjà présentes dans les images. Documentez-vous et activez ces fonctionalités. Notez l'utilisation indispensable de `GLUT_RGBA` à l'initialisation du buffer de rendu.

**3)** Dessinez un deuxième arbre derrière le premier. D'où vient le problème de rendu constaté? Proposez une solution simple pour que l'affichage soit correct.

**Exercice 2.5** *Multitexturing*
OpenGL permet d'utiliser plusieurs textures pour le rendu d'un même polygone. Ceci permet de réaliser un grand nombre d'effets visuels saisissants à nombre de primitives géométriques constant. Pour cela, OpenGL définit des *unités de texture*, dont chacune peut prendre et adresser une texture, avec son propre état (filtrage, contrôle de bordure, etc). Toute texture affectée à une unité active peut alors être combinée avec les autres pour réaliser un effet, dont quelques-un explorés ici.

**1)** Documentez-vous sur l'activation et le fonctionnement des unités de texture, en particulier la fonction `glActiveTexture` qui définit l'unité de texture courante, `glMultiTexCoord2f`.

**2)** La génération de textures irrégulières est possible en combinant 2 textures (ou plus) à des échelles différentes et en mélangeant leur couleurs. Rajoutez à votre sol la texture répétitive semi-transparente `flower.tiff` à l'aide du multitexturing. Pour varier les effets, jouez avec les modes de `glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, ...)`.

**3)** Le multitexturing peut permettre de simuler des éclairages statiques ou dynamiques en combinant une texture supplémentaire de lumière (technique du lightmap). Utilisez `lightmap.tiff` pour simuler l'éclairage ponctuel du sol à un endroit de votre choix. Affectez des touches permettant de déplacer la source de lumière sur le plan du sol. Pour le déplacement de la lumière, utilisez une simple translation `glTranslatef` de la matrice des textures (en mode `glMatrixMode(GL_TEXTURE)`).