Université de Limoges

master ISICG limoges

Informatique Graphique
Moteurs 3D temps réels
TP3 : Textures

G.Gilet

6 octobre 2017

1 Textures en OpenGL - Rappels

Utiliser une texture en OpenGL revient a définir la couleur de chaque pixel en fonction d'un tableau de données (une image). Il faut pour cela connaître pour chaque fragment ses **coordonnées de textures** qui servent à établir une relation entre un élément de la surface et un élément de l'image. Heureusement, les modèles fournis dans ces TP possèdent déjà des coordonnées de texture.
Afin qu'OpenGL puisse utiliser cette texture, il faut au préalable créer un identifiant de texture (pour réserver l'espace en mémoire graphique) et charger la texture en mémoire.

L'accès aux textures en GLSL se fait via une variable (uniform) de type **sampler*d** (pour une texture 2D,**sampler2D**). Cette variable symbolise dans notre shader l'adresse mémoire de notre texture. La fonction *GLSL* **vec4 texture(sampler2D tex, vec2 texCoord)** renvoie la couleur de la texture identifiée par *tex* aux coordonnées *texCoord*.

2 Les textures avec GobLim

Sous GobLim, la création et la gestion des textures passe par la classe **GPUTexture2D**. Le constructeur de cette classe permet le chargement d'une texture, dont le nom de fichier est passé en paramètre directement sur le GPU. Afin de lier la variable GLSL avec la texture, elle aussi présente sur le GPU, il faut lui transmettre l'adresse mémoire de la texture (celle ci peut se récupérer en CPU grâce à la méthode *getHandle()* de la classe **GPUTexture2D**).

3 Exercices

— Afin de vérifier que notre modèle géométrique d'entrée possède bien des coordonnées de textures, modifiez vos shaders afin d'assigner à chaque fragment la couleur des coordonnées de textures du modèle. Dans GobLim, les coordonnées de textures sont de type *vec3* et définies comme un attribut de sommet transmis sur le canal 3. Créez donc dans le VS une variable *in* de

type *vec3* correspondant à l'attribut 3 et assignez cette variable comme couleur pour chaque sommet.

— Affichez maintenant notre bunny avec les informations de texture contenue dans la texture *Bunny1.png*. Créez pour cela un nouveau **Material** et modifiez son constructeur afin de rajouter une **GPUTexture** dans ses paramètres. Afin d'afficher le modèle avec une texture, il faudra ensuite :
  — Récupérer un pointeur (de type **GPUSampler**) sur une variable de type *sampler2D* déclarée dans le *fragment shader* qui matérialisera une unité de texture
  — Transmettre à ce pointeur la valeur de l'adresse mémoire de la texture.

  Utilisez enfin cette texture pour assigner une couleur différente pour chaque fragment en fonction des coordonnées de textures interpolées, grâce à la fonction GLSL **vec4 texture(sampler2D Tex, vec2 texCoord)**
— Chargez la deuxième texture *Bunny2.png*. La coordonnée w de cette texture contient cette fois une information de transparence. Mélangez cette texture (*col2*) avec la texture précédente (*col1*) en effectuant pour chaque fragment : $col = col2.w * col2.xyz + (1.0 - col2.w) * col1.xyz$
— Si ce n'est pas déjà fait, réintroduisez un éclairage de Phong dans votre *fragment shader*, afin d'avoir un modèle à la fois illuminé et texturé. Les couleurs diffuse et ambiante de votre modèle sont cette fois directement lues dans la/les textures.

4 Normal Mapping

L'idée du *normal mapping* est de remplacer la normale à la surface des objets afin d'introduire une illusion de relief. Le principe est d'utiliser pour le calcul d'illumination de Phong, non pas la normale à la surface mais une normale stockée sous la forme d'un triplet RGB dans une texture (la *carte de normales*). Il existe plusieurs logiciels et plugins pour créer automatiquement des cartes de normales à partir de textures ou de modèles 3D. Ces normales peuvent être exprimées directement dans le **repère de l'objet** (ces cartes de normales sont utilisables uniquement avec l'objet pour lequel elles ont été conçues) ou dans un **repère local à la surface**. Elles peuvent dans ce dernier cas être réutilisées pour différents objets.

4.1 Normal Mapping espace Objet

Dans un premier temps, nous allons travailler avec une carte de normales définie dans le repère de l'objet. Il s'agit de la texture $Bunny\_nm.png$. Cette texture contient pour chaque texel une valeur rgb qui peut être interpretée comme une normale. Chargez cette texture et remplacez la normale traditionnelle de l'objet par la normale issue de cette texture. Attention, les valeurs de chaque texel de la texture sont entre 0 et 1 (il est difficile de stocker des valeurs rgb négatives), alors que les valeurs d'une normale peuvent être négative. Il faut donc étendre la valeur du texel pour obtenir une valeur entre -1 et 1.

4.2 Normal Mapping espace surface

Nous allons considérer dans cet exercice des cartes de normales définies dans un repère local à la surface. Ce repère (orthonormé) est défini en chaque point d'une surface et est composé de la **normale** N à la surface, d'un vecteur **tangent** T, et du vecteur **bi-tangent** B. Ces deux derniers vecteurs sont des vecteurs orthogonaux au plan tangent à la surface et sont généralement définis en fonction des variations des coordonnées de textures de la surface. Ce repère est composé sur ses axes X et Y des vecteurs tangents à la surface, et sur son axe Z de la normale à la surface.

4.3 Transformation en repère local à la surface

Les vecteurs tangents à la surface d'un modèle en chaque sommet sont déjà calculées et transmise au pipeline comme attribut défini sur le canal 4. Nous devons calculer pour chaque sommet un repère orthonormé tangent à la surface et exprimer la direction de la lumière et de la caméra dans le repère **local** a chaque sommet. Pour cela,

- Récupérez la variable *Tangent* (de type *vec3*) correspondant au vecteur tangent à la surface en chaque sommet (canal d'attribut 4).
- Le repère que nous voulons créer est exprimé par les vecteurs T, B et N. Calculez B comme étant le produit vectoriel (fonction **cross**) des vecteurs N et T et définissez une matrice (de type **mat3**) composée des vecteurs T, B et N.
- Cette matrice est une matrice de changement de repère, qui exprime la transformation du repère local au sommet vers le repère du modèle. Afin de vous en convaincre et de bien comprendre le mécanisme, voici quelques questions :
  - Dans ce repère TBN local à la surface, quelle est l'expression de la normale ?
  - Pourquoi la carte de normales est majoritairement bleue ?
  - Quelle est la transformation du vecteur précédent par la matrice TBN ?
  - Pouvez vous déduire à quoi correspond cette matrice TBN ?
- La multiplication des vecteurs contenant la direction de la lumière et de la caméra (exprimés dans le repère de l'objet) par l'inverse (ou ici, la transposée) de cette matrice donnera l'expression de ces vecteurs dans le repère local à chaque sommet. Commencez par exprimer ces vecteurs de direction dans l'espace local et réalisez ensuite l'éclairage de Phong en utilisant le vecteur de la normale exprimé dans le repère local. Votre illumination doit à ce point être correcte et identique à celle de l'exercice précédent.

4.4 Du normal mapping

Nous pouvons maintenant charger la texture contenant les normales. Chargez (en plus des textures précédentes) en mémoire graphique la texture de carte de normales (*Bunny_N.png*).

Dans le *fragment shader*, il s'agit maintenant de récupérer la valeur de normale stockée dans la texture (la carte de normales). Les champs R,G et B de la texture correspondent ici aux coordonnées X, Y et Z d'un vecteur exprimé dans le repère TBN local à la surface (la valeur selon l'axe N correspond ici au champ de couleur B, ce qui explique la couleur bleutée des cartes de normales). Toutefois, une

texture stocke des valeurs de R, G et B comprises entre 0 et 1 alors que les valeurs des coordonnées X, Y et Z du vecteur normal peuvent être comprises entre -1 et 1. Il faudra donc convertir les valeurs stockées en normales utilisables. Utilisez cette valeur finale de normale pour effectuer les calcul de l'éclairage de Phong. Vous pouvez également expérimenter avec d'autres textures et cartes de normales (la texture "bricks" par exemple).